Ignition User Manual

Welcome to the New Home for Ignition Documentation!

Welcome to Ignition by Inductive Automation, a true Industrial Application Platform. It is the next generation of accessible, scalable, and data-centric HMI/SCADA software. Ignition was designed from the ground up to be approachable and easy to get started with, but highly flexible and capable of scaling up to the largest projects.

In this manual, you will find reference information along with step-by-step instructions. The final goal of the content and information provided here is to help you gain understanding in regards to Ignition concepts and accomplish tasks within Ignition so that you can be productive and effective while working on your own projects.

The table of content is organized so you can easily find answers to your questions on all the Ignition features, modules, and scripting functions. The Feature section is closely tied with the Inductive University so you can have conceptual understanding from this manual and a visual fast walk-through of the feature in the videos. The Modules section provides information relating to each Module. And finally, in the Appendix, you can easily search to find the specific function you are looking for.

Some of the things you will learn in this manual are: how to install the software, connect to PLCs and databases, create your own projects, create tags, launch clients, set alarms, and generate reports.

Let's get started!

Answers to your Technical Questions

Inductive University

Inductive University, offered by Inductive Automation® SCADA software, is a free online learning platform designed to help you master Ignition. You can watch over 500 training videos, test your knowledge, and participate in our new credential program.

Note the main section of this User Manual is organized in a way to match the **Courses** in the Inductive University. When you need information on any given feature, you can read about the feature here in the User Manual, then go to the same section in the Inductive University Library course list to watch the related video.

When there is a corresponding video for a feature description, you will see a video link as follows:

INDUCTIVE UNIVERSITY

Ignition Video

Watch the Video

Support Team

Support homepage, for one-on-one help from our support team, go to the Support homepage and submit a Ticket. One of our Support Engineers will follow up with you quickly.

You can reach us during business hours 8am-5pm PST at 1-800-266-7798. Support charges may apply. 24-hour support is also available, at an additional fee.

E-mail support is available at support@inductiveautomation.com.

Online Help

You can access this Online Help from the Ignition Gateway and the Designer. To access online Help from the Gateway Webpage, log into the Configure section and select **System > User Manual**. To access it from the Designer, click the **F1** key or select **Help > Help** from the menubar.

Other Resources

- Ignition How-Tos
- White Papers and Articles
- Webinars on Demand
- Tip Sheets

Blogs

Steve Hechtman's Blog, Inductive Automation President

Travis Cox's Blog, Inductive Automation Director of Sales Engineering

Third-Party Tutorials

Technical Resources, for some additional information such as links to sites that contain information about standards, SQL tutorials, and more.

These site are not affiliated with Inductive Automation but we feel they offer excellent advice and information.

Knowledge Base

Knowledge Base, search and view all the ar ticles for troubleshooting, known problems, and workarounds.

Community User Forum

Forum, get help from our passionate community of Ignition users.

The User Forum is one of the most effective ways to get help. The forum is always available, and is actively patrolled by Inductive Automation staff and many knowledgeable users. Chances are you will find your question already answered in an existing post, but if not you can be assured that yours will receive a quick reply.

Comments, Feedbacks, or Ideas?

We encourage your feedback on both the Ignition Software and Ignition User Manual.

Feedback on the User Manual

We encourage you to give us feedback on the Ignition wiki User Manual as this will help us improve the documentation over time. Your feedback can be as small as finding typos or requesting new information on some concepts or how to do something. Now writing Comments and giving feedback is easy! Just use the comment box at the bottom of each page. You can use the keyboard shortcut **m** to jump to the comment box.

Write Comments for the User Community

The comment area at the bottom of each page is mainly for our user community to share and post their questions, answers, tips, and comments related to the information on that specific page. The comments are not monitored by Inductive Automation technical writer or Support team. Therefore, if you have a question that needs an immediate answer, please contact us via the Support homepage.

Give us your ideas on the Ignition Software

How can we improve Ignition? Our goal is to provide you a powerful yet easy-to-use SCADA/HMI software. Let us hear from you on new features and ideas you have so we can continue to improve Ignition.

Ignition By Feature

This **Ignition Manual** section shows all of the documentation for Ignition in a format that is listed by feature or section of the platform. It provides reference and concept information along with step-by-step instructions on how to do just about everything in Ignition.

This section is organized in a similar structure as the Inductive University. In this way, you can have a conceptual understanding from the information provided here, and a fast visual walk-through of the feature by watching the videos. When you need information on any given feature, find and read about the feature here in the User Manual, then go to the Inductive University Library to see it in action. On those pages you will see a link in the upper right corner that looks like this ->

IU INDUCTIVE UNIVERSIT

Video Name

Watch the Video

New to Ignition?

Lets get started by getting Ignition running on your computer!

Our free trial lets you evaluate the full version of the software
without requiring any license.

Download and Install Ignition

- On Windows
- On Linux
- On Mac OS X

The Basics in Ignition

Read these topics and watch the Quick Tour...

- **What is Ignition?**
- **Quick Tour**
- **Architecture and Modules**

A Simple Workflow in Ignition

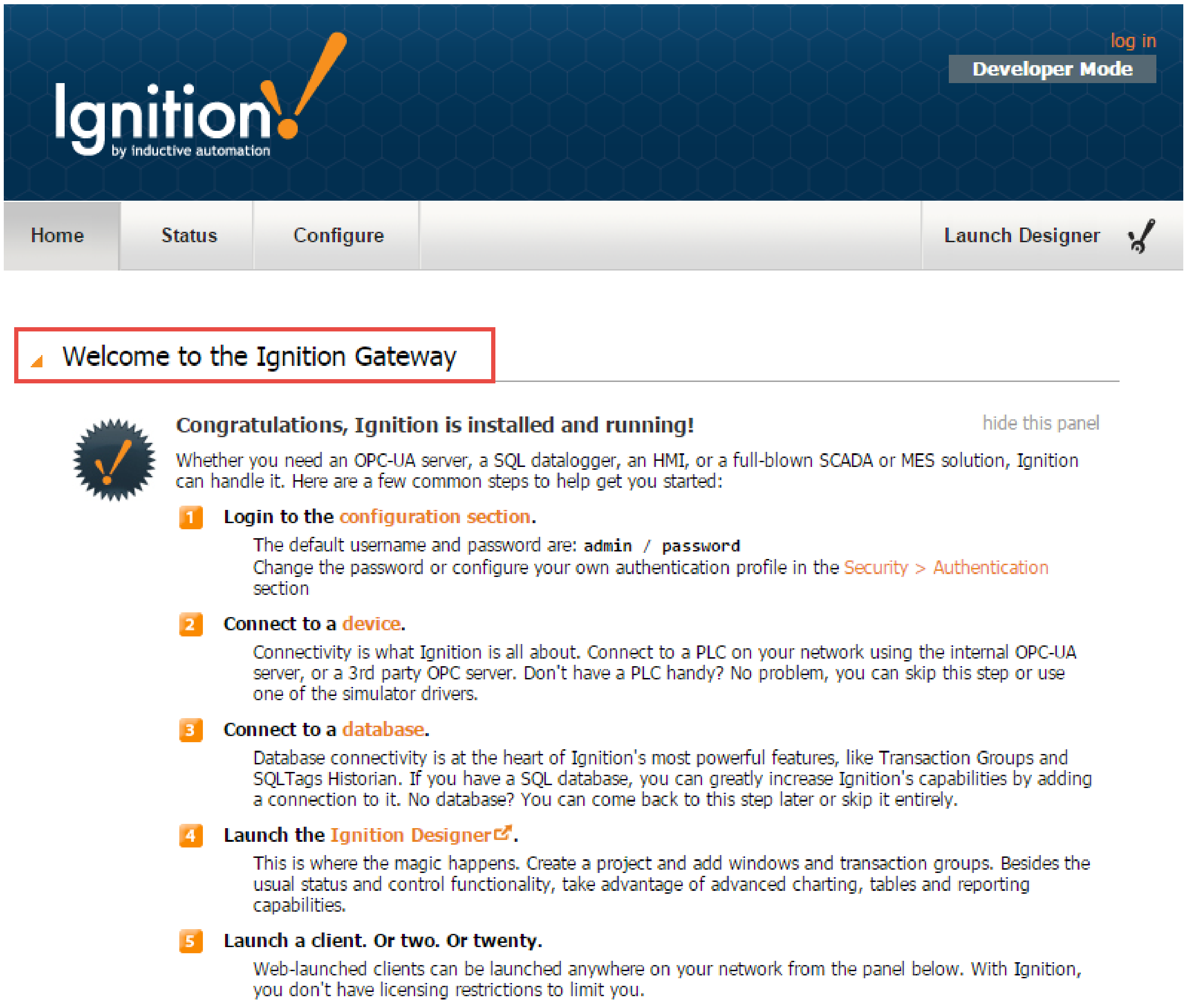
Follow a step-by-step walk-through...

To get started with Ignition, its helps to have a general idea about the major steps involved in designing your SCADA/HMI projects. Although, there are many creative and innovative ways of using Ignition to do almost anything you want, but most users will go through the steps provided here.

Launching the Gateway

The Ignition Gateway is a web server. When it is running, you access it through a web browser. For example, if you are logged into the computer that you installed Ignition on, open up a web browser and go to the address:
http://localhost:8088

and it will launch Gateway and bring up the Gateway Homepage as pictured here.



Welcome to the Ignition Gateway

Congratulations, Ignition is installed and running!

Whether you need an OPC-UA server, a SQL datalogger, an HMI, or a full-blown SCADA or MES solution, Ignition can handle it. Here are a few common steps to help get you started:

1. **Login to the configuration section.**
   The default username and password are: `admin / password`
   Change the password or configure your own authentication profile in the Security > Authentication section
2. **Connect to a device.**
   Connectivity is what Ignition is all about. Connect to a PLC on your network using the internal OPC-UA server, or a 3rd party OPC server. Don't have a PLC handy? No problem, you can skip this step or use one of the simulator drivers.
3. **Connect to a database.**
   Database connectivity is at the heart of Ignition's most powerful features, like Transaction Groups and SQLTags Historian. If you have a SQL database, you can greatly increase Ignition's capabilities by adding a connection to it. No database? You can come back to this step later or skip it entirely.
4. **Launch the Ignition Designer.**
   This is where the magic happens. Create a project and add windows and transaction groups. Besides the usual status and control functionality, take advantage of advanced charting, tables and reporting capabilities.
5. **Launch a client. Or two. Or twenty.**
   Web-launched clients can be launched anywhere on your network from the panel below. With Ignition, you don't have licensing restrictions to limit you.

The first time you go to the Gateway Homepage, it shows you 5 common steps to help you get started. You can follow along with these steps and /or with the step-by-step walk through shown on this page, follow the same basic workflow.

Connecting to a Device

See the following links for the device connection you are interested in:

- Allen Bradley Ethernet
- Modbus
- Siemens
- UDP and TCP Driver
- Third Party OPC Servers

Connecting to a Database

After you install your database, you can connect it to Ignition. To get detailed descriptions for a number of different database connections, see Conn ecting to Database. The common steps in connecting to all databases is as follows:

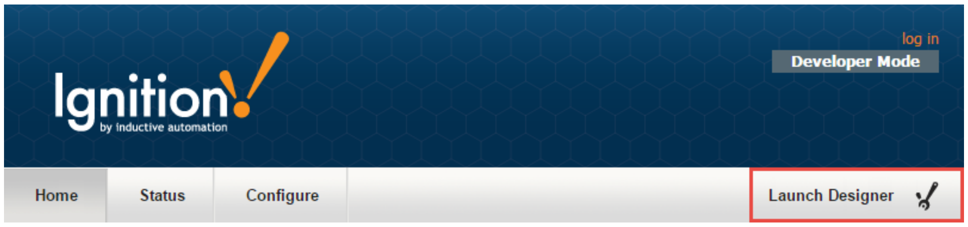
1. **Add a Database Connection**
   Once you are in the Gateway **Configure** section of the Gateway's web interface, use the menu on the left to go to the **Databases > Connections** section. On at the **Database Connections** page, click on the **Create new Database Connection...** link at the bottom of the table.

2. **Pick 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.

3. **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.

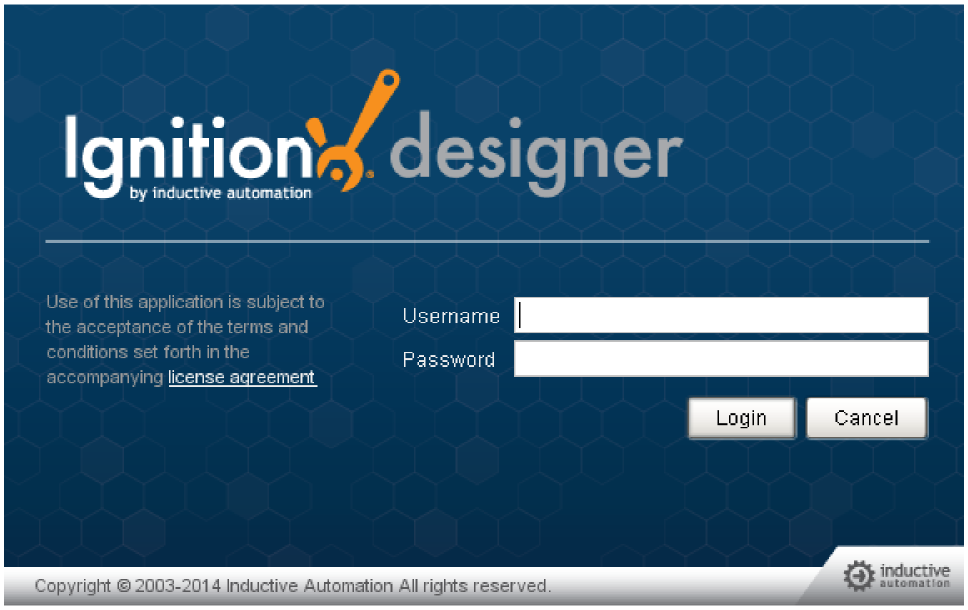
Launching the Designer

1. Click **Launch Designer** on the upper-right of any Gateway page.



   Your web browser asks you if you want to Keep or Discard a **designer.jnlp**file.

2. Click on the **designer.jnlp** file to start the Designer.
   The Designer starts up and the login window is displayed.



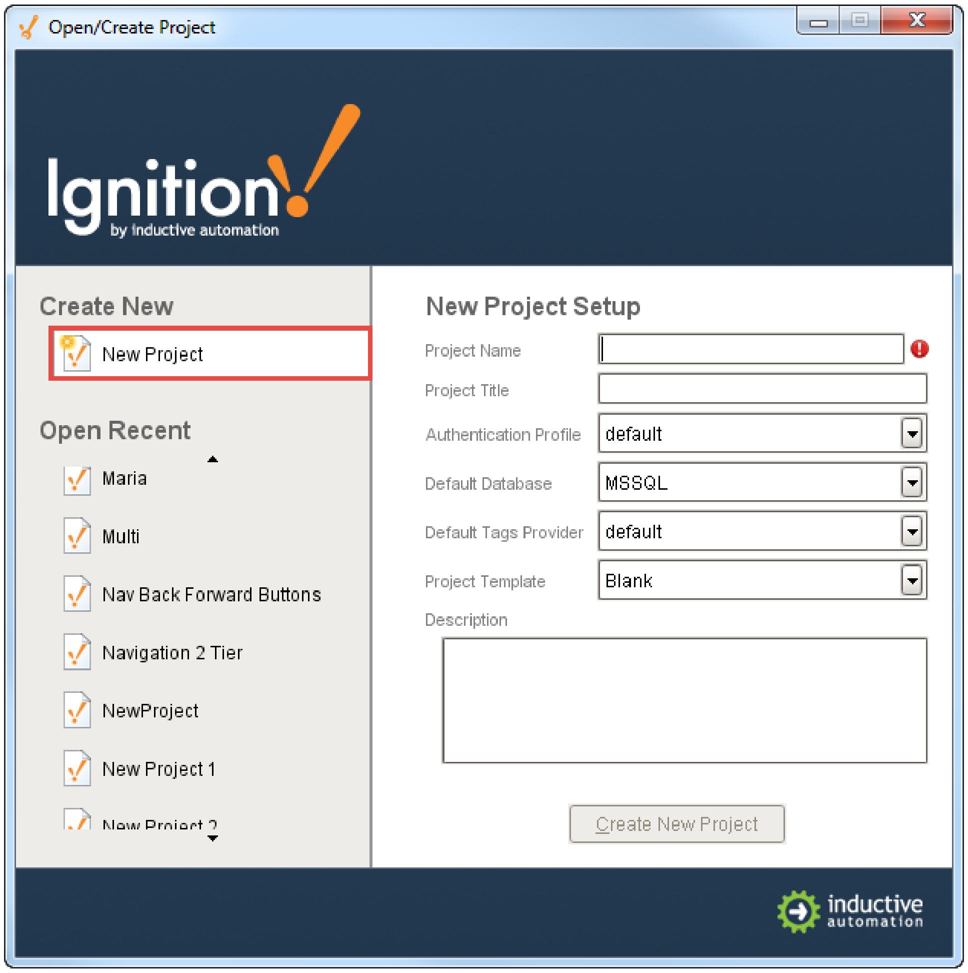
Use of this application is subject to the acceptance of the terms and conditions set forth in the accompanying license agreement

Copyright © 2003-2014 Inductive Automation All rights reserved.

3. Enter the default username and password for the Designer. is the same as for the Gateway Configuration section, enter
   Username: **admin**
   Password: **password**

   Note:
   The default **Username** is set in the **Configure** section, under **Security > Users, Roles**, and going to the **manage users** link for the **User Source**named **default**. Additionally this **Username** must be one of the roles defined in the **Configure** section, under **Configuration > Gateway Settings**, in **Designer Role(s).**

4. Click on **Login**.
The **Open/Create Project** window is displayed giving you two options, to create a new project or open an existing one. As we don't have any existing projects yet, we will create a new project.



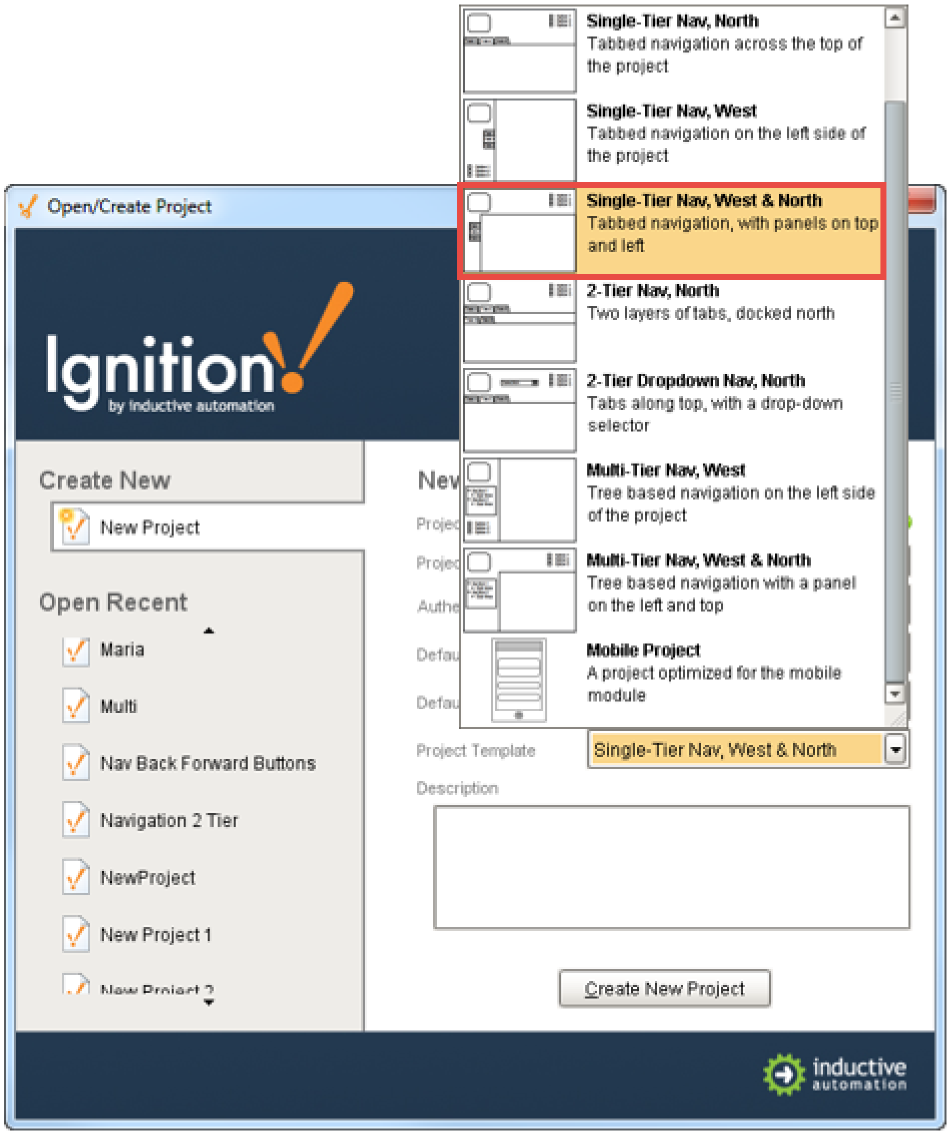
New Project Setup

Project Name	
Project Title	
Authentication Profile	default
Default Database	MSSQL
Default Tags Provider	default
Project Template	Blank
Description	

5. In **Project Name**, enter a name for your new project (no spaces!).

6. For **Project Template**, select a template from the drop-down list, for example, you can select the **Single-Tier Nav, West & North** template.

   Rather than starting from a blank project, the templates let you start from one of the eight skeleton projects that already have navigation windows setup. These projects are completely customizable and are designed to help you get started faster.



Single-Tier Nav, North
Tabbed navigation across the top of the project

Single-Tier Nav, West
Tabbed navigation on the left side of the project

Single-Tier Nav, West & North
Tabbed navigation, with panels on top and left

2-Tier Nav, North
Two layers of tabs, docked north

2-Tier Dropdown Nav, North
Tabs along top, with a drop-down selector

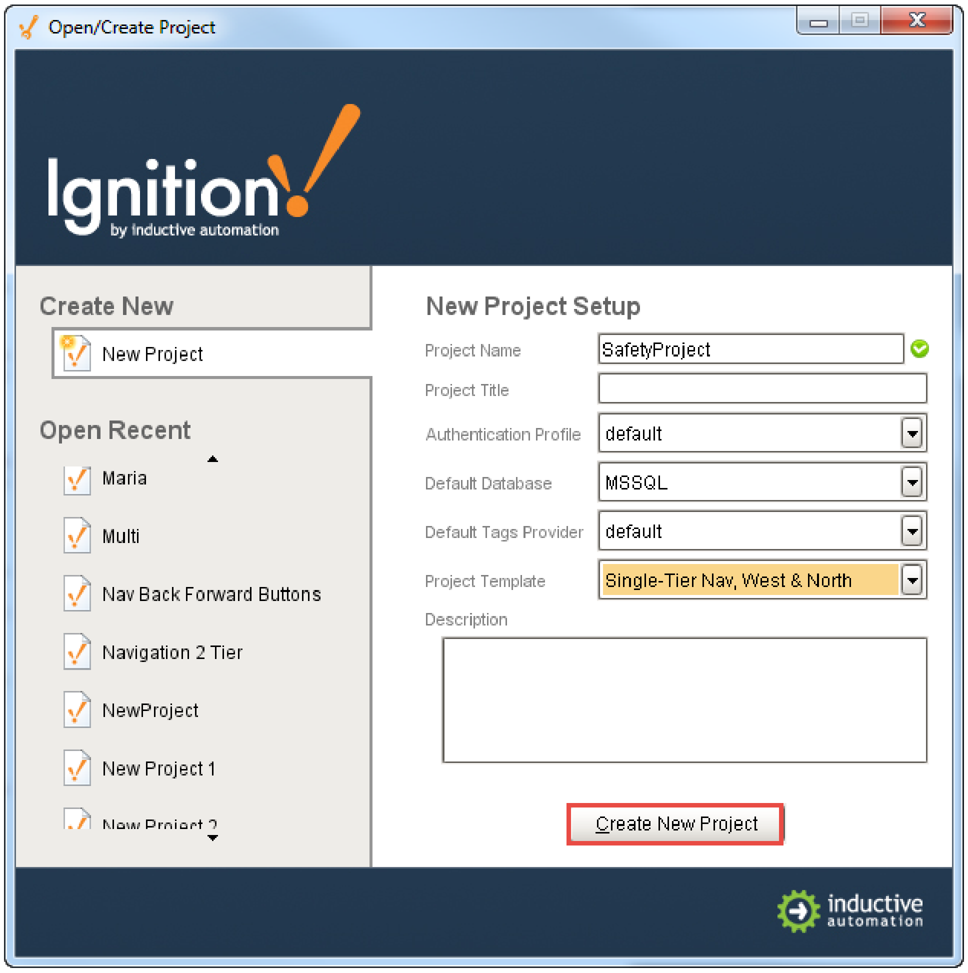
Multi-Tier Nav, West
Tree based navigation on the left side of the project

Multi-Tier Nav, West & North
Tree based navigation with a panel on the left and top

Mobile Project
A project optimized for the mobile module

Project Template: Single-Tier Nav, West & North

7. Click on **Create New Project**.
The Designer is now launched and you can work on your project!



Open/Create Project

Create New

- New Project

Open Recent

- Maria
- Multi
- Nav Back Forward Buttons
- Navigation 2 Tier
- NewProject
- New Project 1
- New Project 2

New Project Setup

Project Name	SafetyProject
Project Title	
Authentication Profile	default
Default Database	MSSQL
Default Tags Provider	default
Project Template	Single-Tier Nav, West & North
Description	

Once in the Designer, your can create or open a project using the Designer's **File > Open** menu. The **Open/Create Project** window is displayed and you can follow the same steps as above to open existing projects or create a new one.

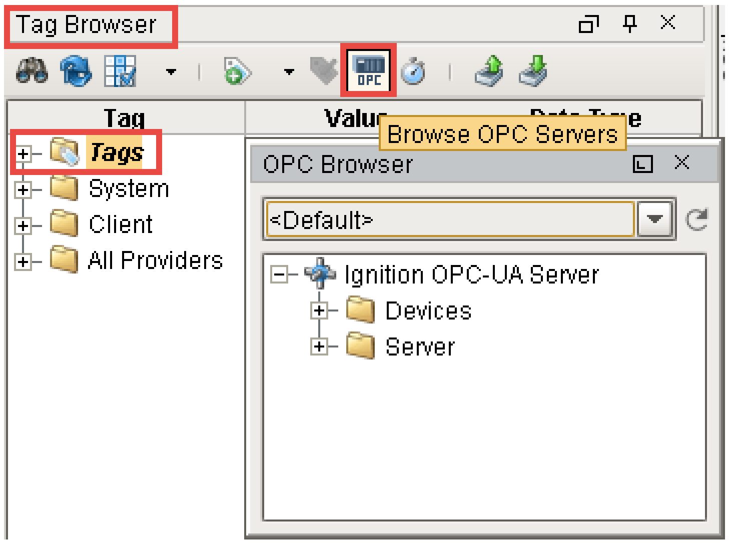
Creating Tags

Now that you are in the Designer, a good first step is to create some Tags. You'll use these tags for realtime status and control and to store history with the Tags Historian. Tags are all configured in the Tag Browser panel.

To create tags by dragging them from OPC

If you have created a device in the earlier step, the easiest way to create some Tags is by dragging the tags from the OPC Browser.

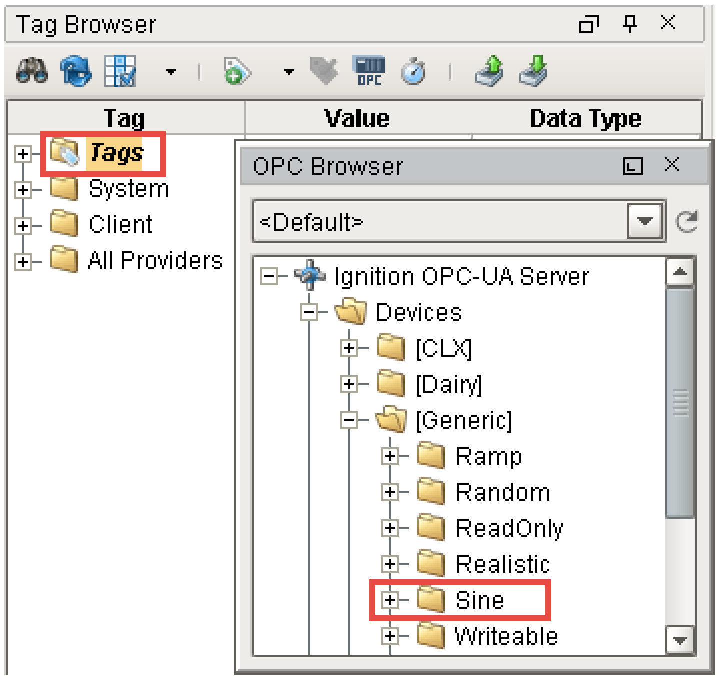
1. In the **Tag Browser**, select the **Tags** folder and click the **Browse OPC Servers** icon.



   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 the devices in it you created earlier. Browse the devices and find some tags that you're interested in.

2. From the **OPC Browser**, highlight the tags (for example, as in the picture below, folders inside the **Sine** folder) and drag them to 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. As soon as you drag the tags into Tag Browser, you can see their values being updated in the Tag Browser,

   That's it! You now have some Tags. You can see their values come in and start updating automatically. By default, they update at a rate of 1 per second.

Launching a Client

Now that we've created a project, lets launch a Client to see it in action.

1. From the Designer toolbar, select **File > Save and Publish** to save your project.

2. Go back to the Gateway **Home** page.
   Your project will appear in the **Launch Projects** panel with a big **Launch** button to its right.

3. Click on the **Launch** button to start up a Client.

4. Login to the Client using
   Username: **admin**
   Password: **password**

   By default, a new project uses the same user source as the Gateway, so the **`admin/password`** credentials will work.

5. After you logged in, you can see your Client window displaying your project.

6. Go back into the Designer, make a change to the window, and click **File > Save and Publish**.
   Your Client will show a notification saying that there are updates to the project.

7. Click on the notification and the Client will update itself.

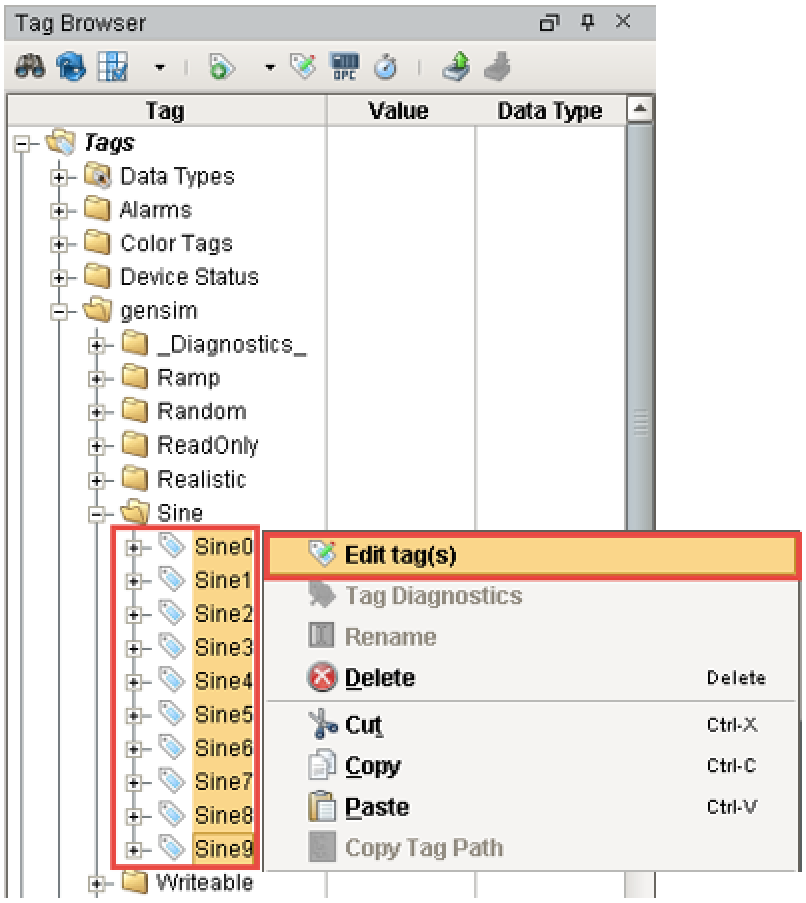
> You can launch as many Clients as you want! If you've got other computers on the same network as the Gateway computer try launching on them too. Make sure that your Gateway computer doesn't have a Firewall enabled, otherwise, it is allowing traffic on port 8088 - the default port for the Ignition web server.

Adding History to Tags

Do the following steps to log history data for your tags:

1. Go to the Designer, from **Tag Browser**, select one or more tags.
   For example, select all the **Sine** tags in the Sine folder.

2. Right-click on the selected tags, and then select **Edit tag(s)**.
   The Tag Editor window is displayed. Here, you can edit the tag and change the name, data type, scaling options, metadata, permissions, history, and alarming.

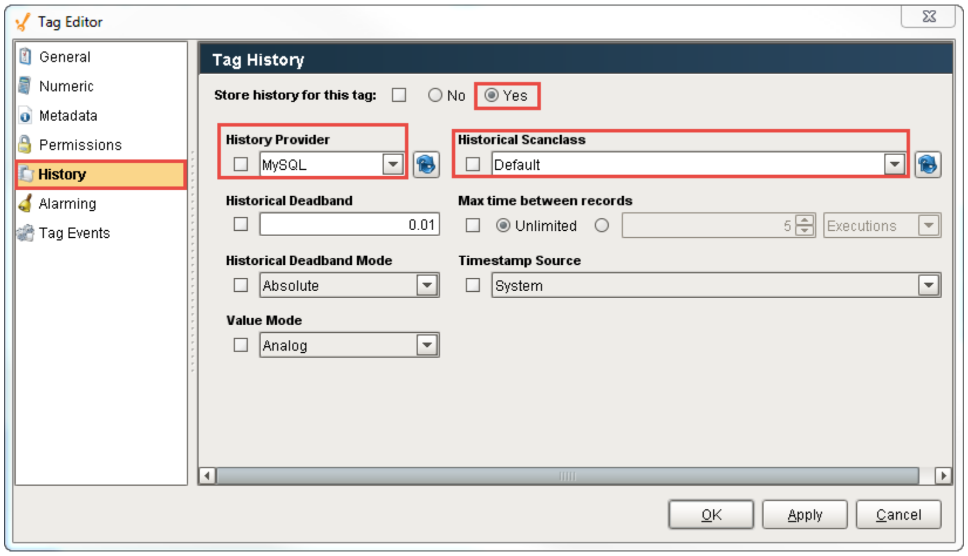


3. Click on **History** on the left side of **Tag Editor**.

4. Click **Yes** for **Store history for this tag**.

5. Choose a database from the **History Provider** dropdown.
   For example, choose **mysql** as the database you want to log to.

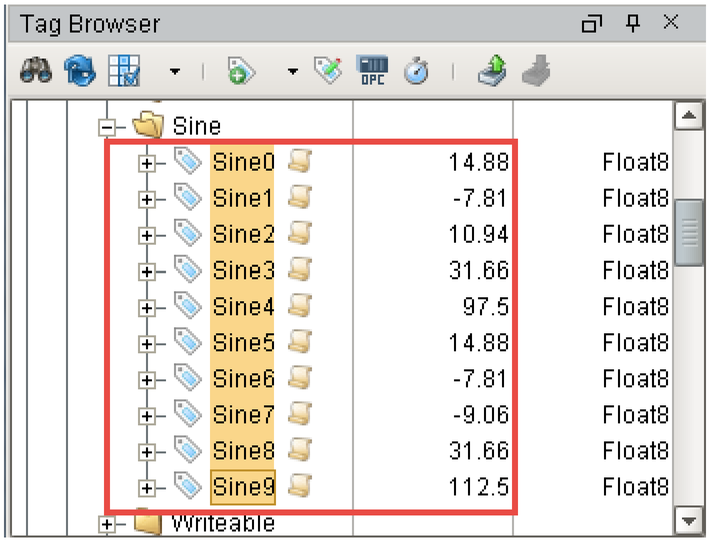
6. Choose **Default** for **Historical Scanclass**.
   This sets the rate at which data is logged. The default is set to 1 second which means data is logged every second.



Store history for this tag: No [✓ Yes]

History Provider: MySQL
Historical Scanclass: Default
Historical Deadband: 0.01
Max time between records: [✓ Unlimited] 5 Executions
Historical Deadband Mode: Absolute
Timestamp Source: System
Value Mode: Analog

7. Click **OK**.
   Now look in the **Tag Browser**, to the right of each **Sine** tag that is storing history, an icon appears letting you know it is setup.



Sine

Sine0	14.88	Float8
Sine1	-7.81	Float8
Sine2	10.94	Float8
Sine3	31.66	Float8
Sine4	97.5	Float8
Sine5	14.88	Float8
Sine6	-7.81	Float8
Sine7	-9.06	Float8
Sine8	31.66	Float8
Sine9	112.5	Float8

Writeable

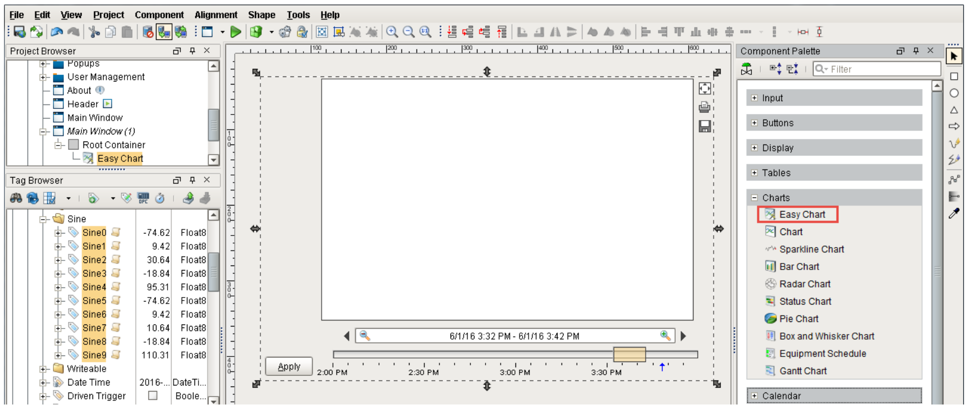
   If you were to look in your database, you can see all the tables and data Ignition has created for you.

Creating a Tag History Graph

You can see the tag history data, by creating a graph. Your graph can include subplots, axes, and a moving average pen. You can view the runtime data in realtime or historical mode.

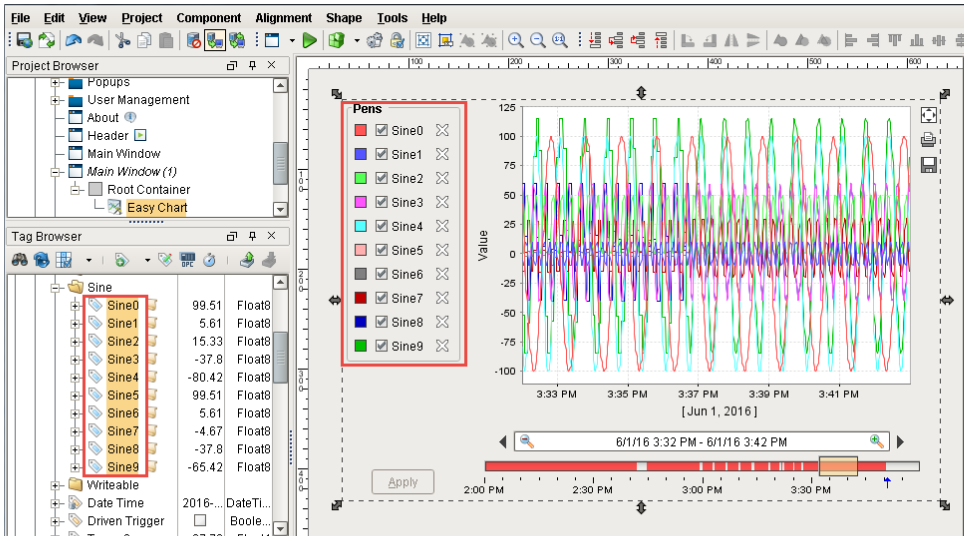
To create a tag history graph

1. Open your Main Window, from the Charts tab of the Component Palette, drag in the Easy Chart component.



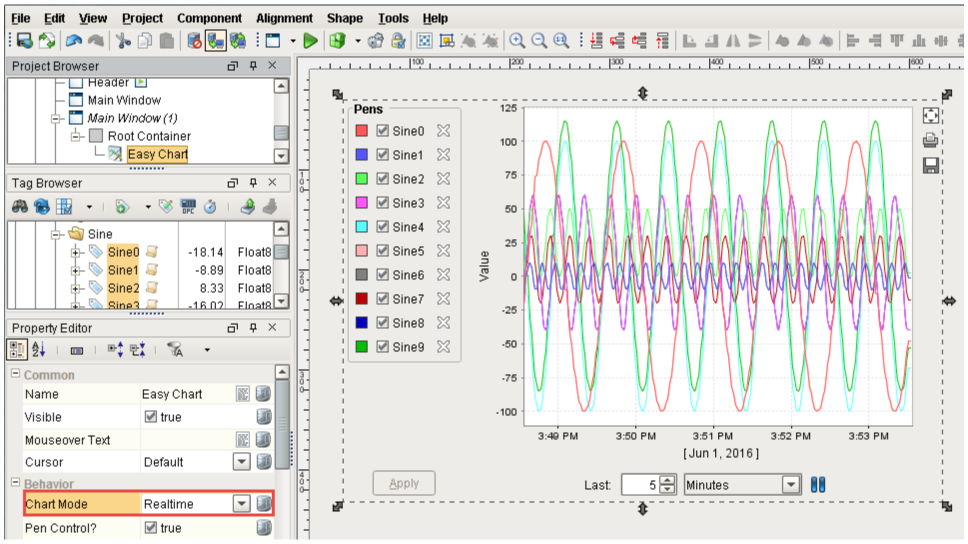
We can add the tags to the graph the same way we created the status and control windows, that is, through drag and drop.

2. Drag all the Sine tags from Tags Browser to the graph.



The tags are displayed on your Main Window but because you have just started logging data, not much data will show up.

3. To show more data, set the Chart Mode property of Easy Chart to Realtime.



Common	
Name	Easy Chart
Visible	✓ true
Mouseover Text	
Cursor	Default
Behavior	
Chart Mode	Realtime
Pen Control?	✓ true

In this section ...

Getting Started

Ignition, a powerful HMI/SCADA/MES software, uses web technology, databases, and OPC-UA to offer a wide variety of configurations based on the customer’s system and plant needs.

In this section ...

Introducing Ignition

This section introduces you to Ignition and fully explains what it can do, what its basic work flow is, its modular architecture, and its the different modules are for your use.

In this section ...

What is Ignition?

Ignition is a software platform for creating custom human machine interface (HMI), supervisory control and data acquisition (SCADA), and manufacturing execution system (MES) applications.

Ignition, designed from ground up, is approachable and easy to get started with while it is flexible and capable of scaling up to the largest projects.

Ignition is installed as server software and it is:

- **Web-based**
  Ignition is installed and deployed using web technologies.
- **Web-managed**
  Ignition's platform is managed using web.
- **Web-launched Designer and Clients**
  Ignition's Designer tool and network clients are launched using web.

On this page...

INDUCTIVE UNIVERSITY

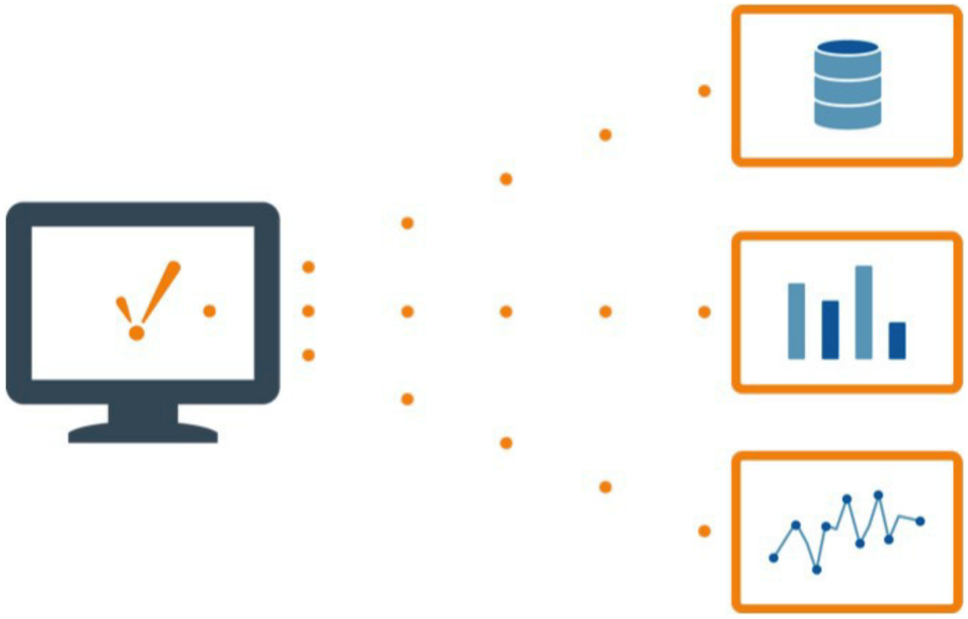
What is Ignition?

Watch the Video

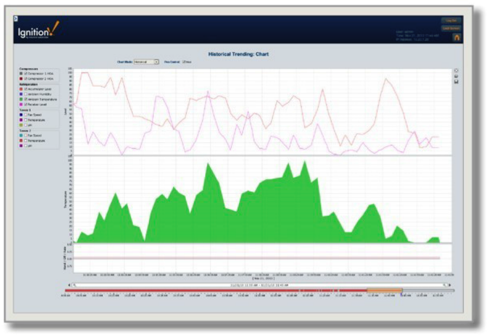
What Can Ignition Do?

Ignition's unified architecture provides the following functionalities:

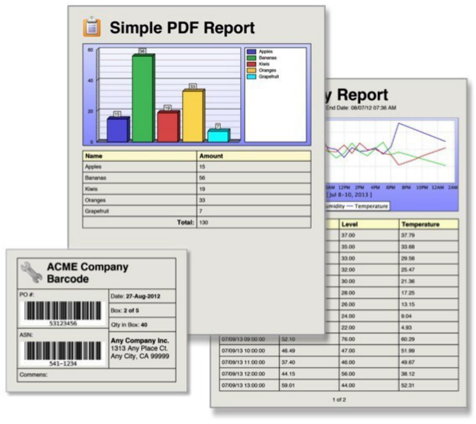
- **Web-based HMI and SCADA applications deployment**
  Using web technologies, you can access PLCs and connect to SQL databases on any platform. Ignition provides the common HMI/SCADA capabilities, but in a lot better way!



- **High performance historian**
  You can use any SQL database with Ignition's high-performance historian. The Historian includes compression algorithms.



- **Reporting Engine**
  You can create dynamic PDF reports using the reporting engine.



Simple PDF Report

Name	Amount
Apples	15
Bananas	56
Kiwis	19
Oranges	33
Grapefruit	7
Total:	130

ACME Company Barcode

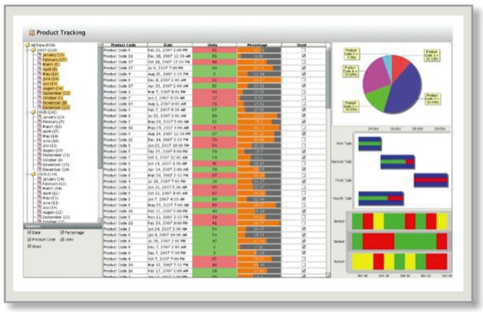
PO #: 53123456	Date: **27-Aug-2012**
	Box: **2 of 5**
	Qty in Box: **40**
ASN: 541-1234	**Any Company Inc.** 1313 Any Place Ct. Any City, CA 99999
Commens:	

y Report
End Date: 08/07/12 07:36 AM
[Jul 8-10, 2013]
umidity — Temperature

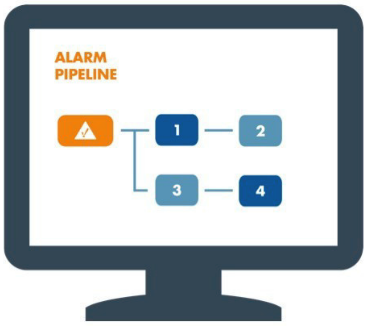
		Level	Temperature
		37.00	37.79
		35.00	33.68
		33.00	29.58
		32.00	25.47
		30.00	21.36
		28.00	17.25
		26.00	13.15
		24.00	9.04
		22.00	4.93
07/09/13 09:00:00	52.10	76.00	60.29
07/09/13 10:00:00	46.49	47.00	51.99
07/09/13 11:00:00	37.40	46.00	49.67
07/09/13 12:00:00	44.15	56.00	38.12
07/09/13 13:00:00	59.01	44.00	52.31

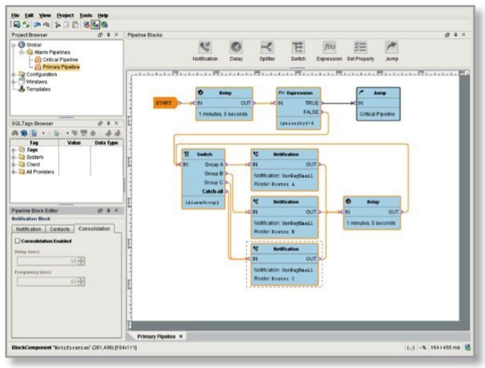
1 of 2

- **Database applications**
  You can build database applications using Ignition. You can create POS, CRM, and inventory tracking systems. You can bring in data from any SQL database in your plant. And you can convert Microsoft Access programs to Ignition allowing multiple people to view the data.

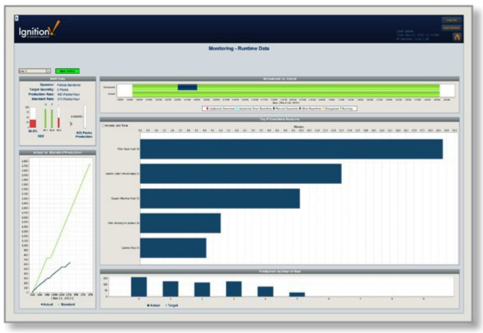


- **Alarming**
  You can use Ignition's state-of-the-art alarming engine to send out notifications via voice, SMS, or email.



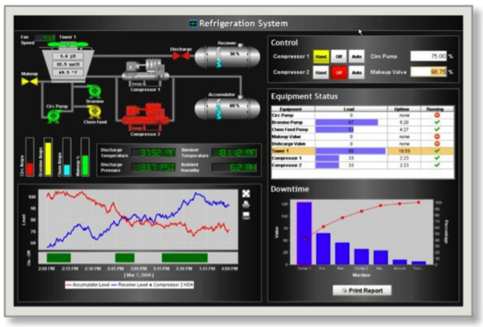


- **MES applications**
  You can host a variety of MES applications including OEE/Downtime, SPC (quality control), Recipe Management, Instrument Interface, and Track & Trace.



- **Mix and match**

  You can mix and match all of these features in any way you want!

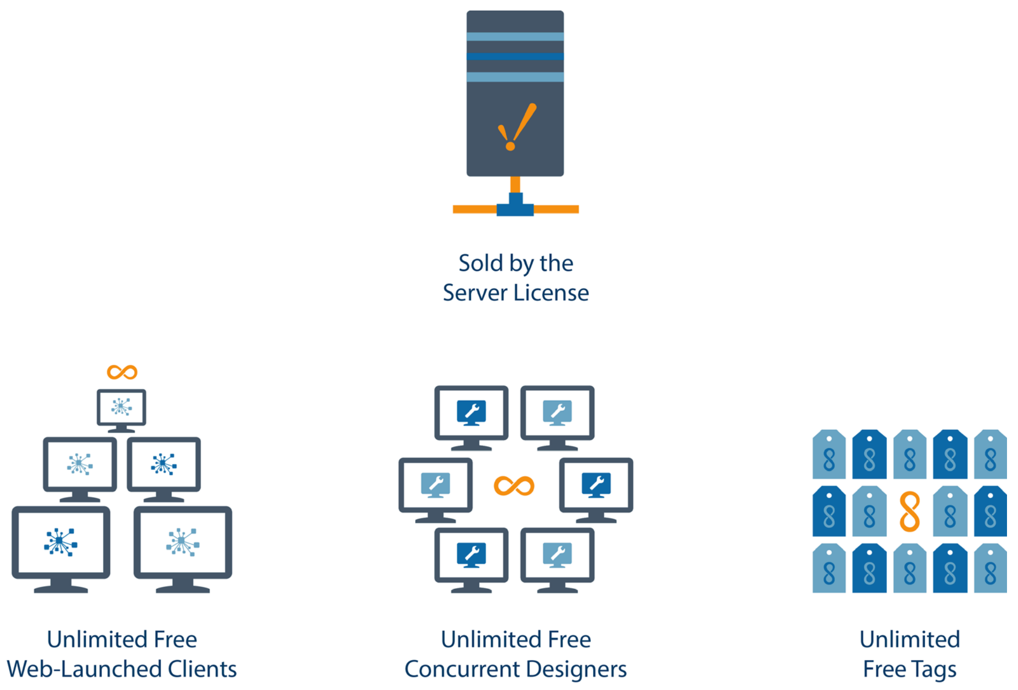


What makes Ignition Special?

Here are the key points that set Ignition apart from other HMI/SCADA software.

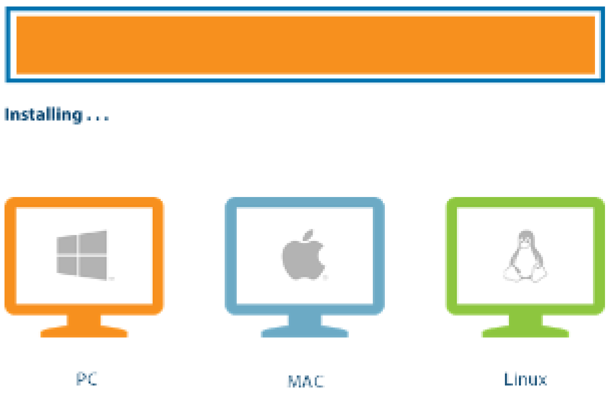
1. Hassle-free Licensing

Ignition is sold by server license and you can add unlimited number of clients, screens, tags, connections, and devices.


Sold by the Server License
Unlimited Free Web-Launched Clients
Unlimited Free Concurrent Designers
Unlimited Free Tags

2. Easy Installation & Deployment

You can install Ignition in less than 3 minutes on any operating system (Windows, Mac, Linux), all you need is a web browser and Java.



3. Rapid design & development environment

You can use the built-in scripting engine, customize projects, and integrate business processes.

4. Flexible & scalable modular software platform

You can add any HMI, SCADA, MES module to your system and they work seamlessly together from the same platform.

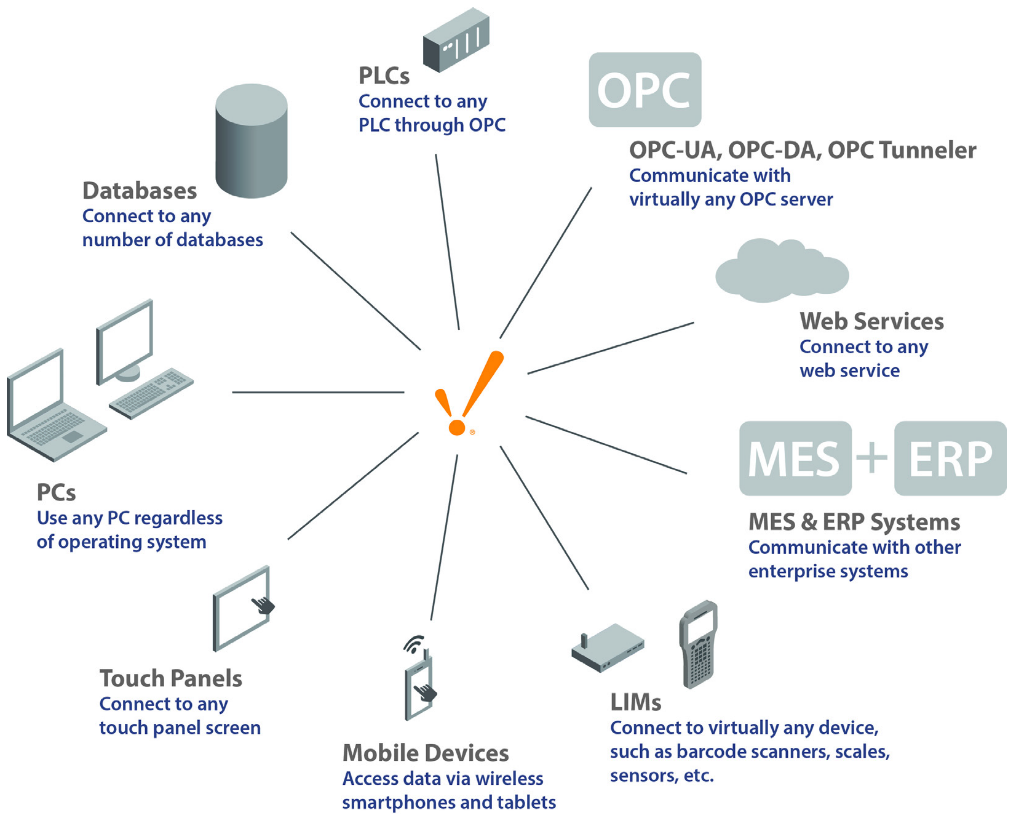
5. Easy to get going

Once you connect to a device and some SQL tags, you are ready to start
creating your first project and to launch your clients.

Ignition as a Communication Hub

You can use Ignition as an effective communication hub in your network. This central location is possible because of the following features in Ignition:

- OPC-based and communicates with virtually any OPC server
- SQL-based (JDBC) and connects to any number of Databases
- Built-in PLC drives and connects to any PLC (Allen-Bradley, Siemens, Modbus) through OPC
- Supports any web services and Launches anywhere
- Connects with other enterprise systems (ERP)
- Connects to devices such as barcode scanners, scales, and sensors
- Runs on any computer regardless of operating system (OS)
- Connects to any mobile device such as wireless smartphones and tablets
- Connects to any touch panel screen

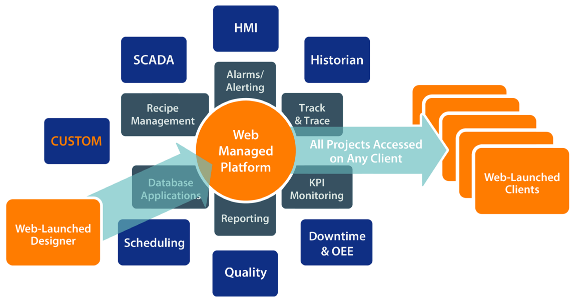

PLCs
Connect to any PLC through OPC
OPC
OPC-UA, OPC-DA, OPC Tunneler
Communicate with virtually any OPC server
Databases
Connect to any number of databases
Web Services
Connect to any web service
PCs
Use any PC regardless of operating system
MES + ERP
MES & ERP Systems
Communicate with other enterprise systems
Touch Panels
Connect to any touch panel screen
Mobile Devices
Access data via wireless smartphones and tablets
LIMs
Connect to virtually any device, such as barcode scanners, scales, sensors, etc.

Next ...

- Quick Tour
- Understanding Ignition's Basic Flow

Understanding Ignition's Basic Flow

Ignition is installed as server software and it is web-based. The basic flow in Ignition is easy, first you launch the Designer from the Gateway webpage, then you configure your project in the Designer and save it back to the Gateway/Server. Once the project is saved on the server, you will then launch the Client, there are no limits to the number of Clients you can launch. When a change is made and saved in the Designer, the Clients are all automatically updated.



Understanding Ignition's Basic Flow

Watch the Video

Next ...

- About Ignition's Modular Architecture

About Ignition's Modular Architecture

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 providing additional capabilities.

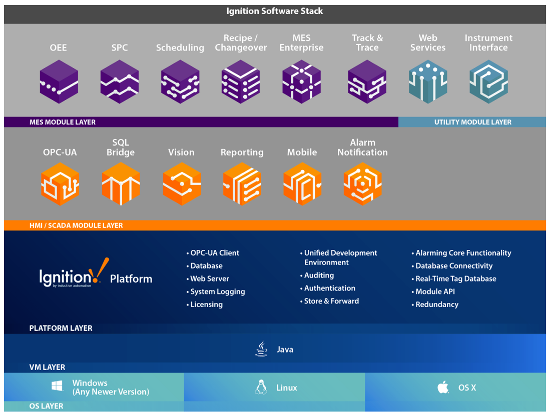
Most of the main features of Ignition are actually provided by different modules such as the Vision and SQL Bridge modules.

Ignition Software Stack

Looking at the Ignition software stack below, you can see that the HMI/SCADA/MES module layers are built on the Ignition platform.

Here are the different software layers in the Ignitions modular architecture:

- **OS Layer – the Operating System Layer**
  Provides basic computing resources such as the file system and access to the network.
- **VM Layer – the Java Virtual Machine Layer**
  Ignition is built on Java and so it runs the same on any operating system.
- **Platform Layer**
  Provides all the basic functionalities such as connecting to devices and databases, licensing, launching clients, managing all Ignition modules over the web, and more.
- **HMI/SCADA Module Layer**
  Provides the core modules that enable real-time and historical data access, trends, and control.
- **MES Module Layer**
  Seamlessly integrates OEE, Downtime, Track & Trace, and other modules.



Next ...

- About Ignition's Modules

INDUCTIVE UNIVERSIT

About Ignition's Modular Architecture

Watch the Video

Steve Hechtman Blog

Click the link below to see the blog:

Why your SCADA system should be more like your smart phone

About Ignition's Modules

What are Modules?

Modules are applications that are built on the Ignition platform, they are similar to applications for a smartphone. The modules are seamlessly integrated onto Ignition's architecture providing you the flexibility to choose them based on your system and plant requirements.

About Ignition's Modules

Watch the Video

HMI/SCADA Modules

The following are some of Ignition's modules. They provide the core functionalities for building your HMI and SCADA system:

- **OPC-UA Module**
  Lets you connect to PLCs via OPC-UA.
- **SQL Bridge Module**
  Lets you track and log historical data.
- **Vision Module**
  Lets you view real-time data via charts, graphs, and more.
- **Reporting Module**
  Lets you build dynamic PDF reports.
- **Mobile Module**
  Lets you access your entire Ignition system via mobile.
- **Alarm Notification Module**
  Lets you configure how, why, and when to send alarm notifications.
- **Voice Notification Module**
  Lets you use phone call alarm notifications.
- **SMS Notification Module**
  Lets you use text message alarm notifications.

MES Modules

The following Ignition modules provide the functionalities for supervising and tracking work on the plant floor:

- **Downtime OEE Module**
  Lets you improve overall equipment effectiveness (OEE) and offers plug-and-play downtime tracking and analysis.
- **SPC Module**
  Lets you add statistical process control (SPC) charting capabilities.
- **Recipe and Changeover Module**
  Lets you build, manage, and monitor recipes.
- **Instrument Interface Module**
  Lets you capture and process data from instruments and devices.
- **Track and Trace Module**
  Lets you see the entire manufacturing process in one unified interface.

For more information on the MES modules, see the MES Module section of the documentation.

Ignition's System Architectures

Ignition being a modern software and system, it combines web servers, databases, and OPC-UA. This combination allows a wide variety of powerful configurations based on the customer’s system and plant needs.

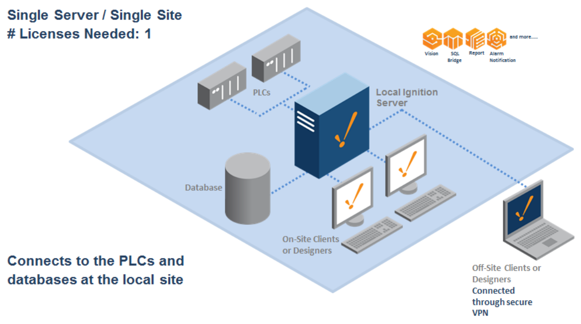
In this section ...

Standard Architecture

This standard server-based architecture provides you a scalable and centrally managed SCADA system with unlimited connections to PLCs, SQL databases, and web-launched clients. You can pick and choose any modules, such as the Vision, SQL Bridge, Reporting, Symbol Factory, and Alarm Notification modules.

Standard Architecture

Watch the Video

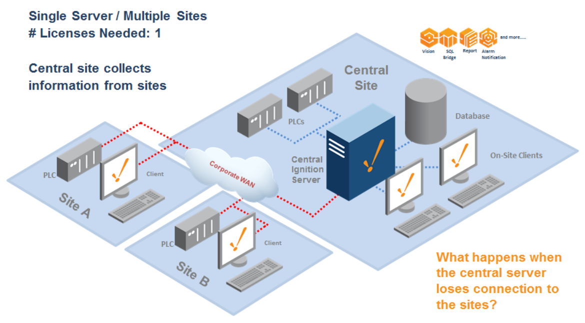

Single Server / Single Site
# Licenses Needed: 1
Vision
SQL Bridge
Report
Alarm Notification
and more.....
PLCs
Local Ignition Server
Database
On-Site Clients or Designers
Connects to the PLCs and databases at the local site
Off-Site Clients or Designers
Connected through secure VPN

Next ...

- Standard Architecture - Multiple Sites

Standard Architecture - Multiple Sites

Using the Standard Architecture you can have a single-server at the central site with connection to PLCs and clients at a number of other sites. Therefore, you have access to all PLC and runtime data from all sites connected to the central Ignition server. The one drawback with this architecture is that if you loose connection to a remote site, all data from that site is lost.



Single Server / Multiple Sites
# Licenses Needed: 1
Central site collects information from sites
Central Site
PLCs
Database
Central Ignition Server
On-Site Clients
Corporate WAN
PLC
Client
Site A
Site B
What happens when the central server loses connection to the sites?

Standard Architecture – Multiple Sites

Watch the Video

Next ...

- Hub and Spoke - Reliable Remote Data Logging

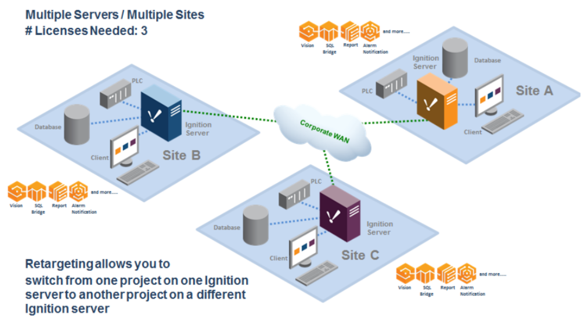
Wide Area SCADA

The problem of loosing data at the remote sites when there is only one Ignition server at the central site is handled by using the Wide Area SCADA architecture. In this architecture, independent Ignition Gateways, for example **The Works** are installed in each facility to provide local control and SCADA functionality. Corporate wide area network (WAN) plus client re-targeting provides a secure and seamless access across multiple facilities. Re-targeting allows you to switch from one project on one Ignition server to another project on a different Ignition server.

When The Works is installed in each facility, the following modules are available at each independent location: the SQL Bridge, Vision, Reporting, Alarm Notification, and Symbol Factory, plus a license for unlimited clients and tags. Of course you can pick and choose which modules you want at each site depending on the needs of that particular site.

Wide Area SCADA

Watch the Video

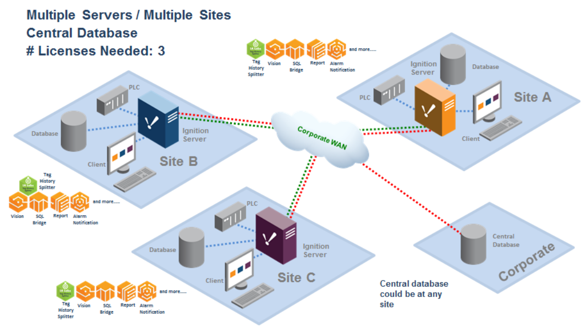


Next ...

- Wide Area SCADA - Central Database

Wide Area SCADA - Central Database

Multiple independent Ignition Gateways hosted at different locations and connected by wide area network (WAN) allow you can log data both locally and centrally into one database. The **Tag History Splitter** mod ule allows you to login to two places at the same time so you can have access to the data at the other sites and log all the data to the central database.



Wide Area SCADA – Central Database

Watch the Video

Next ...

- Hub and Spoke - Reliable Remote Date Logging

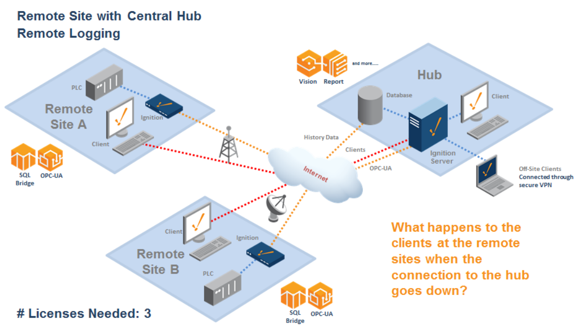
Hub and Spoke - Reliable Remote Data Logging

Because connections between a database and remote PLCs are usually unreliable, with the hub and spoke architecture, the SQL Bridge module in Ignition is installed on embedded data logging computers. This module uses the store-and-forward technique to ensure that data is never lost. The system may still be accessed through the OPC-UA stand alone architecture, or database-based tags may be used to communicate current status. By doing this, it is possible to use Tag Historian and Alarming, both of which utilize the store-and-forward system to avoid data loss. This method is ideal for historical data.

INDUCTIVE UNIVERSITY

Hub and Spoke Architecture – Reliable Remote Data Logging

Watch the Video

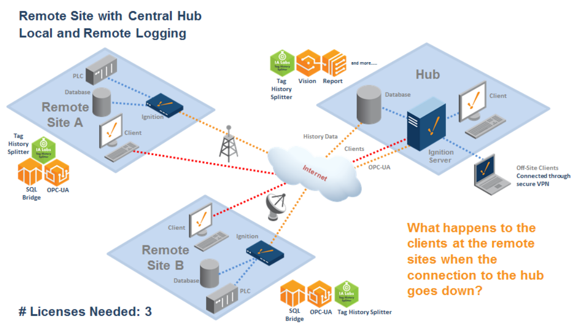


Next ...

- Hub and Spoke - Local and Remote Logging

Hub and Spoke - Local and Remote Logging

Combining the Hub and Spoke architecture with the Tag History Splitter module allows you to simultaneously log data locally and remotely for redundant data. The one problem here is that if the connection between the central and remote site is lost, the clients at the remote site will loose access to the central server, but data is not lost as there are remote and central databases.



INDUCTIVE UNIVERSITY

Hub and Spoke – Local and Remote Logging

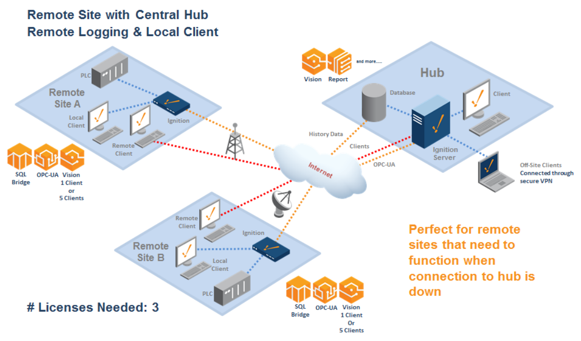
Watch the Video

Next ...

- Hub and Spoke - With Vision Module

Hub and Spoke - With Vision Module

Typically the Hub and Spoke architecture doesn't include the **Vision** Module at the remote sites. You can add the Vision Module in local client fallbacks to each remote site so that you can keep the project visible when you lose communication to the hub. **Vision Limited** licenses help lower costs on remote machines.



Hub and Spoke – With Vision Module

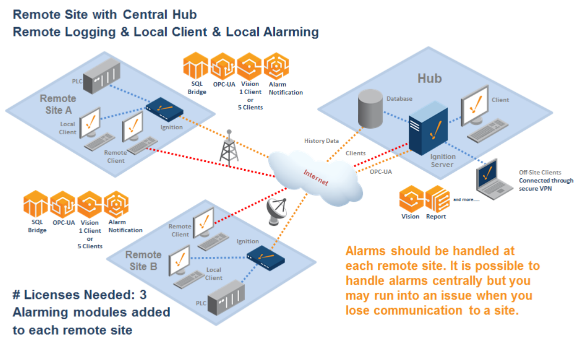
Watch the Video

Next ...

- Hub and Spoke - With Alarming

Hub and Spoke - With Alarming

In addition to the Vision module, you can add other modules to the remote sites. For example, by adding the **Alarming** Module to a remote site, you can add alarming capabilities to that location. It is best to handle alarms at each remote site rather than centrally to avoid any problems in case the connection to the central location is lost.


Remote Site with Central Hub
Remote Logging & Local Client & Local Alarming
PLC
Remote Site A
Local Client
Remote Client
Ignition
SQL Bridge
OPC-UA
Vision 1 Client or 5 Clients
Alarm Notification
Hub
Database
Client
Ignition Server
History Data
Clients
OPC-UA
Internet
Off-Site Clients Connected through secure VPN
Vision
Report
and more.....
SQL Bridge
OPC-UA
Vision 1 Client or 5 Clients
Alarm Notification
Remote Client
Ignition
Remote Site B
Local Client
PLC
# Licenses Needed: 3
Alarming modules added to each remote site
Alarms should be handled at each remote site. It is possible to handle alarms centrally but you may run into an issue when you lose communication to a site.

Hub and Spoke – With Alarming

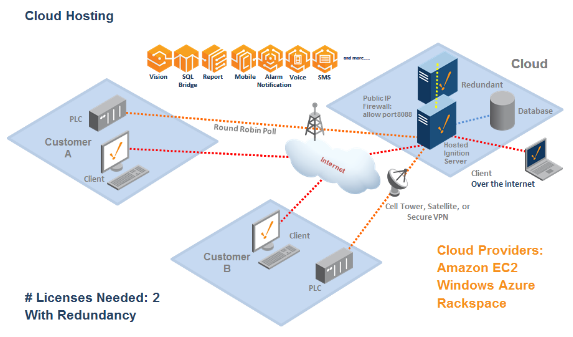
Watch the Video

Next ...

- Hosted Ignition Server

Hosted Ignition Server

Ignition server can be hosted in the Cloud. Cloud providers are: Amazon EC2, Windows Azure, or Rackspace. You can have two Ignition servers with redundancy plus the central database in the Cloud. The server can connect to the PLCs at the remote or customer sites via Round Robin Poll, Cell Tower, Satellite, or Secure VPN. The clients are connected to the Ignition server via Internet.



INDUCTIVE UNIVERSITY

Hosted Ignition Server

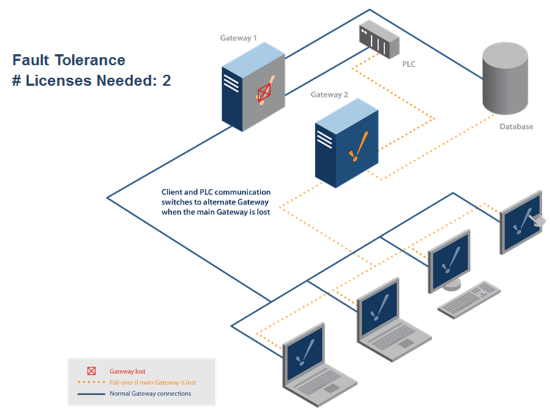
Watch the Video

Next ...

- Mission Critical - Gateway Redundancy

Mission Critical - Gateway Redundancy

Using redundancy, two Ignition installations can be linked together so that when the master fails, the backup takes over and continues executing. All of the clients connected are redirected to the backup machine, and historical data continues to log.


Fault Tolerance
# Licenses Needed: 2
Gateway 1
PLC
Gateway 2
Database
Client and PLC communication switches to alternate Gateway when the main Gateway is lost
Gateway lost
Fail-over if main Gateway is lost
Normal Gateway connections

Mission Critical Architecture – Gateway Redundancy

Watch the Video

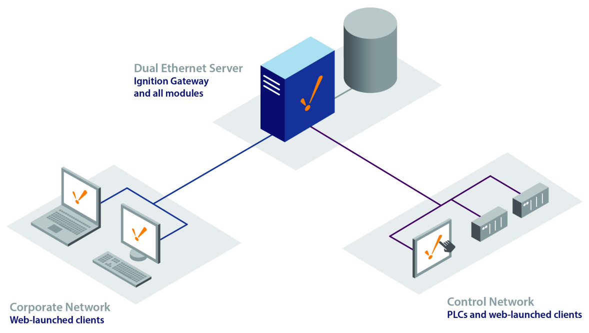
Next ...

- Bridging Corporate and Control Networks

Bridging Corporate and Control Networks

The Ignition Gateway supports dual-NIC servers, and can act as a bridge between multiple networks. Since clients talk to databases and PLCs through the Gateway, clients can be launched from both a corporate network and an isolated control network and provide full access to both.

Network security and easy access without compromise



Bridging Corporate and Control Networks

Watch the Video

Next ...

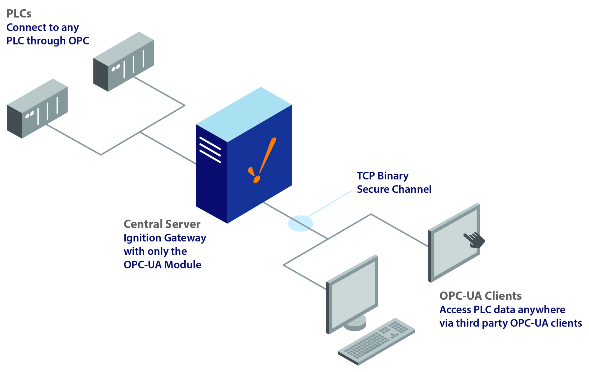
- OPC-UA Stand Alone Architecture

OPC-UA Stand Alone Architecture

OPC-UA is a network-based specification, and is ideal for collecting data from remote locations. Ignition can operate as an OPC I/O server with just the OPC-UA Module and UA drivers installed. Other OPC-UA client software can then access the PLC data through Ignition.

This method only exposes data, however, and the Client must then record it if historical data is desired. If the connection goes down, data will not be available. This method offers the lowest cost, and is suited for situations where the data is not highly critical or historical, for example, remote realtime monitoring.

OPC-UA Licenses are Free



INDUCTIVE UNIVERSITY

OPC-UA Stand Alone Architecture

Watch the Video

Installing and Upgrading Ignition

Installing Ignition

You can download and install Ignition in just a few minutes! It's very simple:

1. Download Ignition from your web browser
2. Run the installer on any any Windows, Mac, or Linux operating system that can run Java 8
3. The Gateway Webpage opens automatically after the installer finishes so you can get started immediately!

See the links shown below for detailed installation instructions on Windows, Linux, and Mac OS X operating systems.

Upgrading Ignition

The Ignition installer also doubles as an upgrader. Simply download the executable installer from our website for your operating system. See the links below for upgrading instructions specific to Windows, Linux, and Mac OS X operating systems.

Regardless of which Operating System you are using, upgrading has a simple set of steps:

1. Download Ignition from your web browser
2. Run the installer on the computer that has an older version of Ignition (don't forget to backup you work)
3. The installer automatically upgrades all of your projects and opens the Gateway Webpage!

For a complete guide to protect your projects when upgrading, see the Best Practices when Upgrading page.

Installing Ignition on Windows

Before you install Ignition, make sure you meet the minimum system requirements:

- Java 8+ (server), Java 6+ (client)
- Dual-Core Processor (32 or 64 bit)
- 4GB RAM
- 10GB free HD space

Also, ensure you have Administrator privileges to run the Ignition installer.

On this page ...

Installing Ignition on Windows

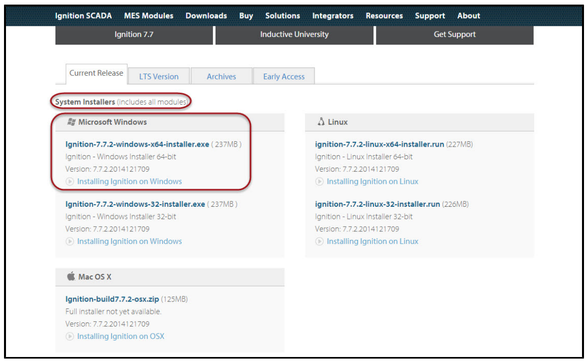
Watch the Video

To download and install Ignition on Windows

1. Go to the Inductive Automation website at www.inductiveautomation.com

2. Select **Download > Current Release**.

3. On the Ignition Downloads page, scroll down to **System Installers** and select the **ignition-X.X.X-windows-x64-installer.exe** installer.
   If your computer runs on a 32-bit Windows, click the ignition-X.X.X-windows-32-installer.exe installer.



System Installers (includes all modules)

Microsoft Windows

Ignition-7.7.2-windows-x64-installer.exe (237MB)
Ignition - Windows Installer 64-bit
Version: 7.7.2.2014121709
Installing Ignition on Windows

Ignition-7.7.2-windows-32-installer.exe (237MB)
Ignition - Windows Installer 32-bit
Version: 7.7.2.2014121709
Installing Ignition on Windows

Linux

ignition-7.7.2-linux-x64-installer.run (227MB)
Ignition - Linux Installer 64-bit
Version: 7.7.2.2014121709
Installing Ignition on Linux

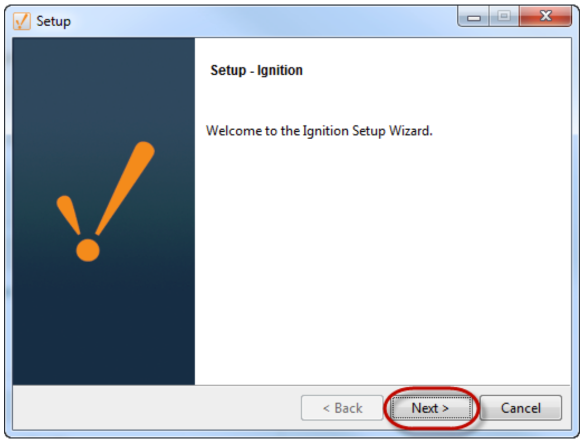
ignition-7.7.2-linux-32-installer.run (226MB)
Ignition - Linux Installer 32-bit
Version: 7.7.2.2014121709
Installing Ignition on Linux

Mac OS X

Ignition-build7.7.2-osx.zip (125MB)
Full installer not yet available.
Version: 7.7.2.2014121709
Installing Ignition on OSX

   Note: An error window is displayed if you try to install 64-bit Ignition on a 32-bit Windows system. You are allowed to install 32-bit Ignition on a 64-bit system (however, this is not recommended).

4. After the installer downloads, click on the **.exe** executable file to start installation.
   The Setup window welcomes you to the Ignition Setup Wizard.

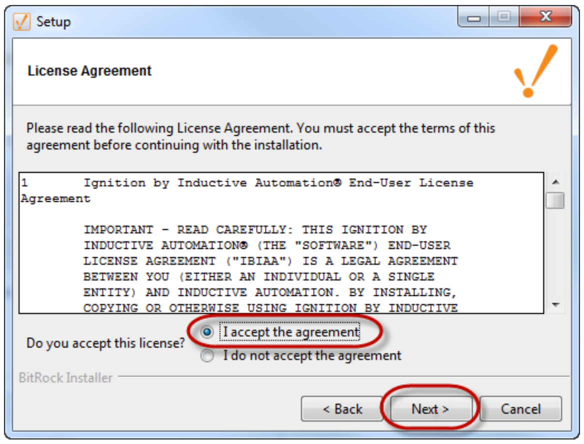


Setup - Ignition

Welcome to the Ignition Setup Wizard.

5. Click **Next**.

6. Click the **I accept the agreement** button and click **Next**.



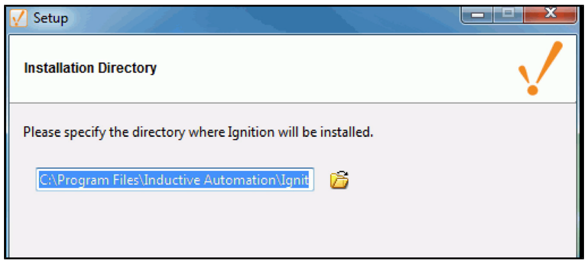
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

```
1       Ignition by Inductive Automation® End-User License
Agreement

        IMPORTANT - READ CAREFULLY: THIS IGNITION BY
        INDUCTIVE AUTOMATION® (THE "SOFTWARE") END-USER
        LICENSE AGREEMENT ("IBIAA") IS A LEGAL AGREEMENT
        BETWEEN YOU (EITHER AN INDIVIDUAL OR A SINGLE
        ENTITY) AND INDUCTIVE AUTOMATION. BY INSTALLING,
        COPYING OR OTHERWISE USING IGNITION BY INDUCTIVE
```

Do you accept this license? [✓ I accept the agreement] I do not accept the agreement

The Installation Directory window is displayed. By default, Ignition is installed under your Program Files directory. This is usually a good choice.

7. Make a note of the default installation directory, **Program Files/Inductive Automation/Ignition**, accept the default Installation Directory, and click **Next**.

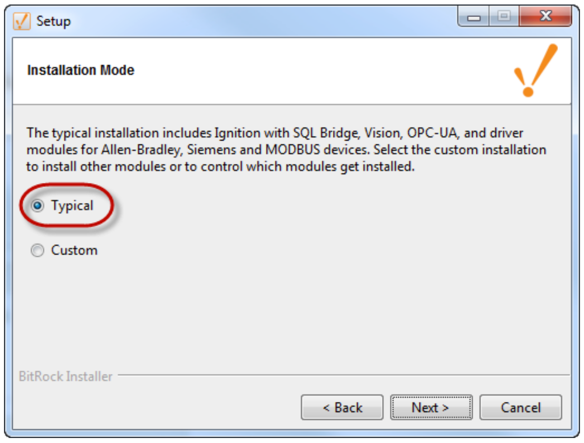


Please specify the directory where Ignition will be installed.

C:\Program Files\Inductive Automation\Ignit

The **Installation Mode** window is displayed.

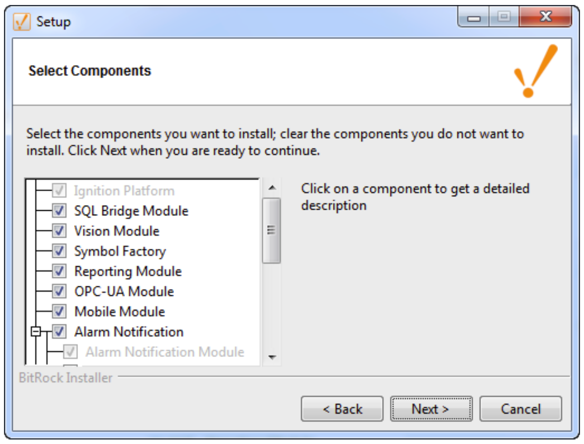
8. Select either the **Typical** or **Custom** mode.
The **Typical** installation includes Ignition with all of the HMI/SCADA modules. The only modules that are not installed by default are the MES modules.
The **Custom** installation lets you control and individually select the modules that you want installed.



The typical installation includes Ignition with SQL Bridge, Vision, OPC-UA, and driver modules for Allen-Bradley, Siemens and MODBUS devices. Select the custom installation to install other modules or to control which modules get installed.

[✓ Typical] Custom

9. Click **Next**.
The Setup window is displayed.
If you selected **Custom** mode, in the Setup window you can select the check box next to the modules you want to install as part of the Ignition installation. To see a brief description of the module, click on the module name.


Setup

Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

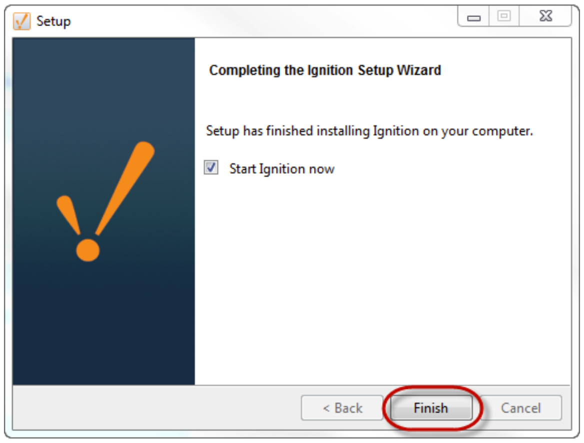
- ✓ Ignition Platform
- ✓ SQL Bridge Module
- ✓ Vision Module
- ✓ Symbol Factory
- ✓ Reporting Module
- ✓ OPC-UA Module
- ✓ Mobile Module
- ✓ Alarm Notification
  - ✓ Alarm Notification Module

Click on a component to get a detailed description

Setup Wizard shows Ignition is now ready to install.
If you decide to change your selections, click **Back** to go back and make your changes.

10. Click **Next.**

11. After Setup Wizard shows Ignition installation is finished, make sure that the check box for **Start Ignition now** is selected, then click **Finish**.

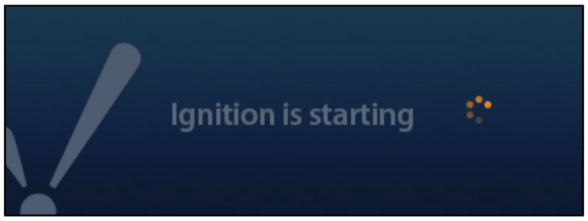

Setup

Completing the Ignition Setup Wizard

Setup has finished installing Ignition on your computer.

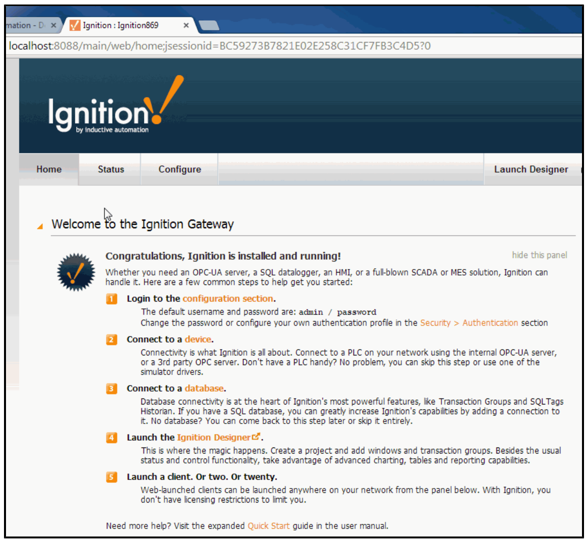
✓ Start Ignition now

A splash screen informs you that Ignition is starting. This may take a few minutes to finish.



That's it! Ignition Installation is complete and the Gateway is then launched.

Your web browser opens the Gateway home page at
http: //localhost:8088


mation - D
Ignition : Ignition869
localhost:8088/main/web/home;jsessionid=BC59273B7821E02E258C31CF7FB3C4D5?0
Ignition by inductive automation
Home
Status
Configure
Launch Designer
hide this panel

Welcome to the Ignition Gateway

Congratulations, Ignition is installed and running!

Whether you need an OPC-UA server, a SQL datalogger, an HMI, or a full-blown SCADA or MES solution, Ignition can handle it. Here are a few common steps to help get you started:

1. **Login to the configuration section.**
   The default username and password are: `admin / password`
   Change the password or configure your own authentication profile in the Security > Authentication section
2. **Connect to a device.**
   Connectivity is what Ignition is all about. Connect to a PLC on your network using the internal OPC-UA server, or a 3rd party OPC server. Don't have a PLC handy? No problem, you can skip this step or use one of the simulator drivers.
3. **Connect to a database.**
   Database connectivity is at the heart of Ignition's most powerful features, like Transaction Groups and SQLTags Historian. If you have a SQL database, you can greatly increase Ignition's capabilities by adding a connection to it. No database? You can come back to this step later or skip it entirely.
4. **Launch the Ignition Designer.**
   This is where the magic happens. Create a project and add windows and transaction groups. Besides the usual status and control functionality, take advantage of advanced charting, tables and reporting capabilities.
5. **Launch a client. Or two. Or twenty.**
   Web-launched clients can be launched anywhere on your network from the panel below. With Ignition, you don't have licensing restrictions to limit you.

Need more help? Visit the expanded Quick Start guide in the user manual.

Ignition installs itself as a Window Service (**Start > Control Panel > Administrative Tools > Services > Ignition Gateway**), so it starts automatically when your computer starts up.

Automated Installation on Windows

You can automate Ignition installation across different machines using scripts. Ignition is then installed silently from a command shell without showing any user prompts.

Keep in mind that the installer cannot automatically start the Gateway after a silent installation. You can use the following command to start the Gateway:
`net start ignition`

Command line example:

```
Ignition-7.x.x-windows-x64-installer.exe --mode unattended --prefix "C:\some folder" --unattendedmodeui none

net start ignition
```

Flags:

`-- mode unattended` (ensures that no prompts appear during installation)
`-- prefix "C:\some folder"` (optional flag; if a value is set, then Ignition is installed in the specified folder, otherwise Ignition is installed in a default location under C:\Program Files)
`-- unattendedmodeui none | minimal` (the none flag does not display any sort of graphic during installation; the `minimal` flag displays a small progress bar and nothing else)

Uninstalling Ignition on Windows

Before you uninstall Ignition...

Be sure to back up your Gateway and unactivate your Gateway license.

During a new Ignition installation, an uninstaller executable is created.

When Ignition is uninstalled, the settings database and folder in `/data` is backed up to `/data_<current date>`, and modules and user-supplied JDBC jars in `/user-lib` are backed up to `/user-lib_<current date>`. Also, the Ignition service is removed from Windows installations automatically.

To run the uninstaller for Windows,

1. Open the Start menu and navigate to **All Programs > Inductive Automation > Ignition.**

2. Select **Uninstall Ignition**.

   The uninstaller wizard guides you through the uninstallation process.

You can also uninstall Ignition from the Windows **Control Panel > Programs and Features** section.

Next ...

- Activating Over Internet
- Activating Manually
- Upgrading Ignition on Windows

Upgrading Ignition on Windows

Before you upgrade Ignition...

You should backup your Gateway, see Making Gateway Backups.

The Ignition installer also doubles as an upgrader, thus upgrading Ignition is very similar to downloading a new version and installing from scratch. You can see your current Ignition version by looking in the **Status** page of the Gateway.

To upgrade Ignition on Windows

1. Simply download the Windows executable installer from our website, and double-click on the executable installer.
   A window is displayed asking you **Continue with upgrade?**

   Note: Make sure you download the 32-bit installer for a 32-bit Windows installation, and the 64-bit installer for a 64-bit Windows installation. If you attempt to upgrade a 32-bit installation with a 64-bit installer, or vice versa, you will get an error message.

2. Click **Yes** to start the upgrade process.

3. Accept the license agreement.
   The **Upgrade Mode** window is displayed.

4. You choose either:
   No, just upgrade currently installed modules
   The Ignition base installation and the existing modules are upgraded.

5. Or you can choose:
   Yes, let me choose which modules to add or remove
   The Select Components window is displayed and you can then add a module that was not part of the original installation or uninstall/remove a currently installed module.
   You can view a brief description of each module by clicking on the module name.
   Selecting the check box next to a module, installs the module as part of the upgrade.
   Clearing the check box next to a module, uninstalls the module during the upgrade.

6. Click **Next**.
   The **Ready to Install** window is displayed.
   Ignition is ready to be upgraded. At this point, you may click **Back** to change your selections.

7. Click **Next** button to finish the upgrade.
   Once the upgrade process starts, it may take a few minutes to finish. At the end of the upgrade process, the Setup window is displayed letting you know that **Setup has finished installing Ignition on your computer**.

8. Select the check box for **Start Ignition now** and click **Finish.**
   A splash screen displays informing you that the Ignition service is starting. Once the Ignition Gateway starts up, your web browser opens and brings you to the Gateway Homepage.

   You can go back and check the **Status** page of the Gateway to see that a new version of Ignition is now installed.

Upgrading Ignition on Windows

Watch the Video

Tip Sheet

Upgrading from 7.6 or earlier?

Please read this upgrade guide: Ignition 7.7 Upgrade Guide

Automated Upgrading

You can silently upgrade Ignition from a command shell without showing any user prompts. Remember to make a Gateway backup before performing any type of upgrade. Also, the installer cannot automatically start the Gateway after a silent upgrade.

Use the `net start ignition` command as shown below.

Command line example:

```
gwcmd -b C:\backups\mybackup.gwbk
```

```
Ignition-7.x.x-windows-x64-installer.exe --mode unattended --unattendedmodeue none --autoupgrade true
```

```
net start ignition
```

Flags:

`-- mode unattended` (ensures that no prompts appear during installation)
`-- unattendedmodeui none | minimal` (the `none` flag does not display any sort of graphic during installation; the `minimal` flag displays a small progress bar and nothing else)
`-- autoupgrade true` (disables any `Are you sure`? prompts when running the upgrader so that a user can run an automated upgrade without any user interaction)

Next ...

- add the link here

Installing Ignition on Linux

Before you install Ignition, make sure you meet the minimum system requirements:

- Java 8+ (server), Java 6+ (client)
- Dual-Core Processor (32 or 64 bit)
- 4GB RAM
- 10GB free HD space

Also, it is assumed that you are familiar with operating a shell, and you have **root** or **superuser** privileges to run the Ignition installer.

On this page ...

Installing Ignition on Linux

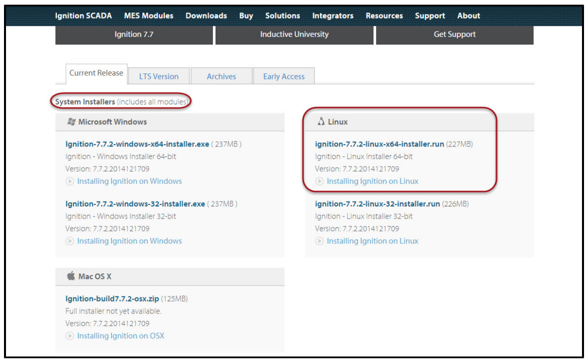
Watch the Video

To download and install Ignition on Linux

1. Go to the Inductive Automation website at www.inductiveautomation.com

2. Select **Download > Current Release**.

3. On the Ignition Downloads page, scroll down to System Installers and select the **ignition-X.X.X-linux-x64-installer.run** installer.
   If your computer runs on a 32- bit Linux, select the ignition-X.X.Xlinux-32-installer.run.

   System Installers (includes all modules)

   Microsoft Windows

   Ignition-7.7.2-windows-x64-installer.exe (237MB)
   Ignition - Windows Installer 64-bit
   Version: 7.7.2.2014121709
   Installing Ignition on Windows

   Ignition-7.7.2-windows-32-installer.exe (237MB)
   Ignition - Windows Installer 32-bit
   Version: 7.7.2.2014121709
   Installing Ignition on Windows

   Linux

   ignition-7.7.2-linux-x64-installer.run (227MB)
   Ignition - Linux Installer 64-bit
   Version: 7.7.2.2014121709
   Installing Ignition on Linux

   ignition-7.7.2-linux-32-installer.run (226MB)
   Ignition - Linux Installer 32-bit
   Version: 7.7.2.2014121709
   Installing Ignition on Linux

   Mac OS X

   Ignition-build7.7.2-osx.zip (125MB)
   Full installer not yet available.
   Version: 7.7.2.2014121709
   Installing Ignition on OSX

   Note: If you choose the wrong installer, an Error window is displayed letting you know the correct installer to use for your machine.
   The installers can run in both graphical or command-line mode, allowing Ignition to install on a headless Linux server. The installer will install files in the following locations:

   `/usr/local/bin/ignition`
   (unless a different installation directory was used) – contains binaries, startup scripts and the uninstall executable

   `/var/lib/ignition/data`
   contains application-generated files, temporary files and the internal database

   `/var/lib/ignition/user-lib`
   contains modules and JDBC jars

   `/var/log/ignition`
   contains the wrapper.log and other log files

   `/etc/ignition`
   contains configuration files. Symbolic links to these files are created in

   `/var/lib/ignition/data`

4. You must run all the commands as root. Prefix everything with `sudo` or first run
   `sudo su`

5. If Java 8 is not already installed, run the following commands to install it:
   ```
   sudo add-apt-repository ppa:webupd8team/java
   sudo apt-get update
   sudo apt-get install oracle-java8-installer
   ```

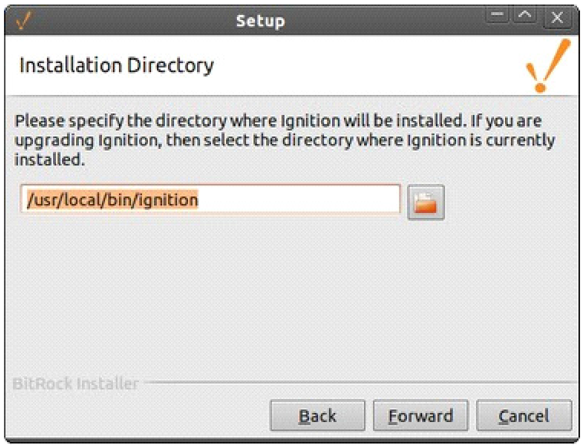
6. Open a command shell and navigate to the installer executable, by running the following command from the command line:
   `sudo chmod +x ignition-x.x.x-linux-xxxinstaller.run`

7. Start the installer executable.
   The installer can run in a **graphical** mode or **command line** mode.
   If you are running the installer in a shell in a graphical environment, the graphical installer opens automatically.
   If you are running the installer in a headless Linux installation or through an SSH shell, the text installer opens automatically.
   To start the text installer in a graphical environment, run the following command:
   `sudo ./ignition-X.X.X-linux-xxx-installer.run -- mode text`

   The installer starts running.

8. Agree to the licensing terms.

9. Ensure the installation directory is the default directory at:
   `/user/local/bin/ignition`

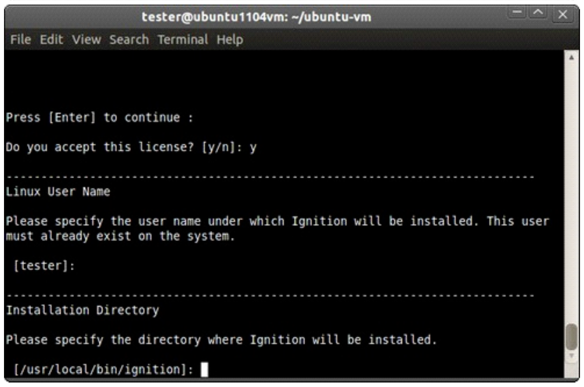


Setup

Installation Directory

Please specify the directory where Ignition will be installed. If you are upgrading Ignition, then select the directory where Ignition is currently installed.

/usr/local/bin/ignition



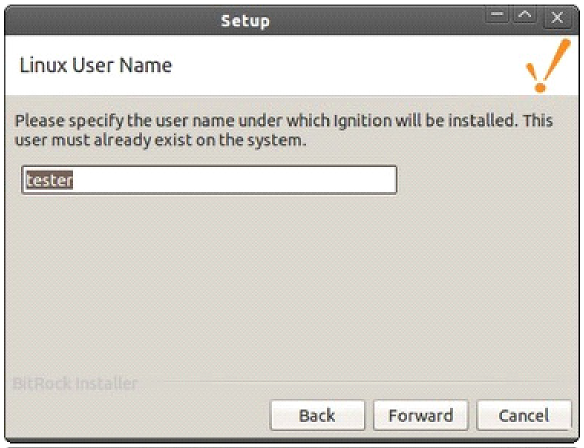
tester@ubuntu1104vm: ~/ubuntu-vm
File Edit View Search Terminal Help

```
Press [Enter] to continue :

Do you accept this license? [y/n]: y

----------------------------------------------------------------------------
Linux User Name

Please specify the user name under which Ignition will be installed. This user
must already exist on the system.

 [tester]:

----------------------------------------------------------------------------
Installation Directory

Please specify the directory where Ignition will be installed.

 [/usr/local/bin/ignition]:
```

10. Specify the user name under which Ignition is to be installed. For Ubuntu installations, the user that invoked sudo is used by default. For other Linux installations, this field is initially blank.

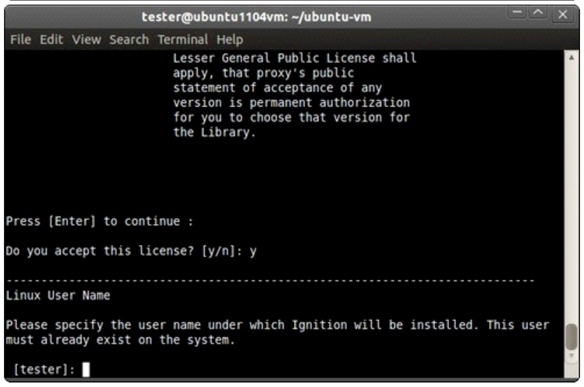
    The Linux installer requires a user name to set the permissions on the folders created by the installer. This user is then able to start and stop Ignition, run the Gateway Control Utility, and the command line interfaces.

    The binaries in the installation folder are still owned by root and cannot be modified without root access. The selected user must already exist on the system before starting the installer.



Please specify the user name under which Ignition will be installed. This user must already exist on the system.

tester

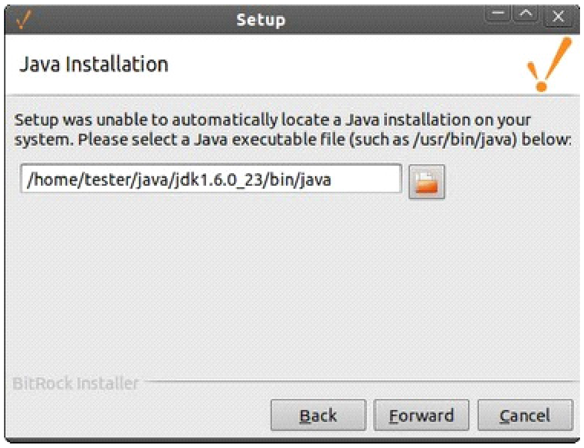

tester@ubuntu1104vm: ~/ubuntu-vm
File Edit View Search Terminal Help

```
                        Lesser General Public License shall
                        apply, that proxy's public
                        statement of acceptance of any
                        version is permanent authorization
                        for you to choose that version for
                        the Library.




Press [Enter] to continue :

Do you accept this license? [y/n]: y

----------------------------------------------------------------------------
Linux User Name

Please specify the user name under which Ignition will be installed. This user
must already exist on the system.

 [tester]: 
```

Normally, the installer is capable of auto-detecting a Java 8 installation that has been installed through APT or some other Linux package management tool. In these cases, the installer uses that Java installation and skips this step.

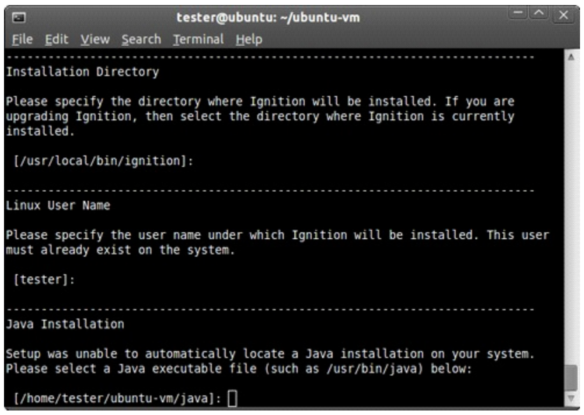
However, if you have installed Java by extracting the Java binaries to a folder and adding them to the system PATH, the installer is unable to find the Java binaries. You must provide the installer with a full path to the Java executable.

11. If the Java Installation window is displayed, type the full path to the Java executable.



Setup was unable to automatically locate a Java installation on your system. Please select a Java executable file (such as /usr/bin/java) below:

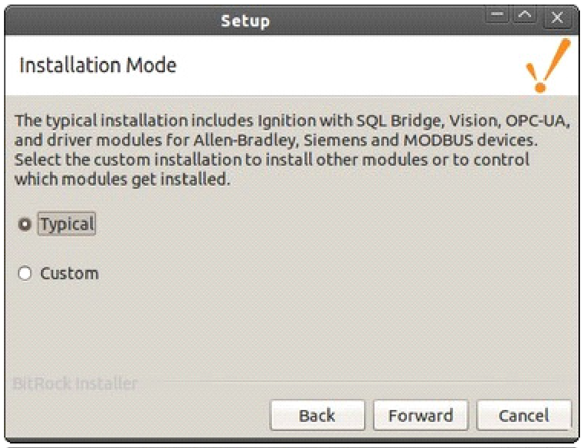
/home/tester/java/jdk1.6.0_23/bin/java


```
-------------------------------------------------------------------------------
Installation Directory

Please specify the directory where Ignition will be installed. If you are
upgrading Ignition, then select the directory where Ignition is currently
installed.

 [/usr/local/bin/ignition]:

-------------------------------------------------------------------------------
Linux User Name

Please specify the user name under which Ignition will be installed. This user
must already exist on the system.

 [tester]:

-------------------------------------------------------------------------------
Java Installation

Setup was unable to automatically locate a Java installation on your system.
Please select a Java executable file (such as /usr/bin/java) below:

 [/home/tester/ubuntu-vm/java]:
```

12. Select either the **Typical** or the **Custom** installation mode.

    The **Typical** mode installs everything that you need to get started.

    The **Custom** mode lets you add optional modules, such as the OEE Downtime module. It also lets you control and select which modules get installed.

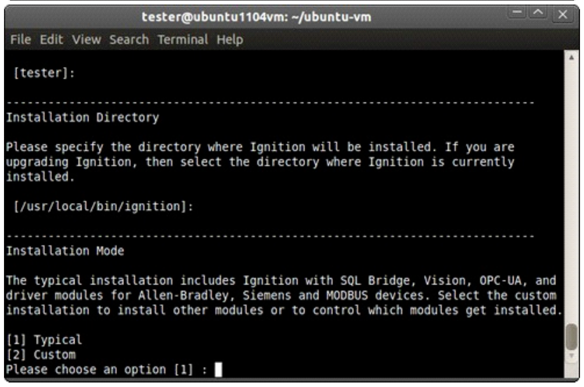


Installation Mode

The typical installation includes Ignition with SQL Bridge, Vision, OPC-UA, and driver modules for Allen-Bradley, Siemens and MODBUS devices. Select the custom installation to install other modules or to control which modules get installed.

[✓ Typical]

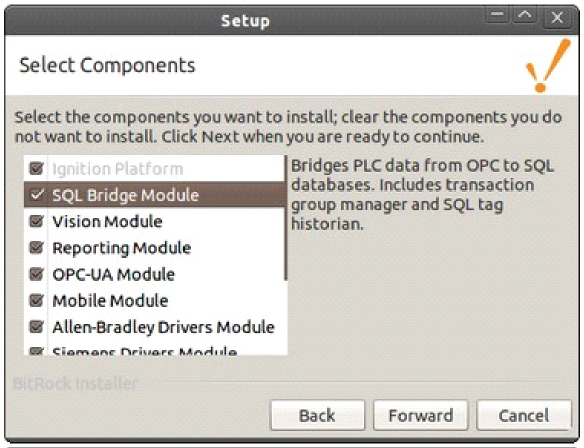
Custom


```
 [tester]:

-------------------------------------------------------------------------------
Installation Directory

Please specify the directory where Ignition will be installed. If you are
upgrading Ignition, then select the directory where Ignition is currently
installed.

 [/usr/local/bin/ignition]:

-------------------------------------------------------------------------------
Installation Mode

The typical installation includes Ignition with SQL Bridge, Vision, OPC-UA, and
driver modules for Allen-Bradley, Siemens and MODBUS devices. Select the custom
installation to install other modules or to control which modules get installed.

[1] Typical
[2] Custom
Please choose an option [1] :
```

If you select **Custom** Mode, the Select Components window is displayed.

• In **Graphical mode** to view a brief description of the module, click on the module name. Selecting the check box next to a module installs the module as part of the Ignition installation. Clearing the check box next to a module prevents the module from being installed.

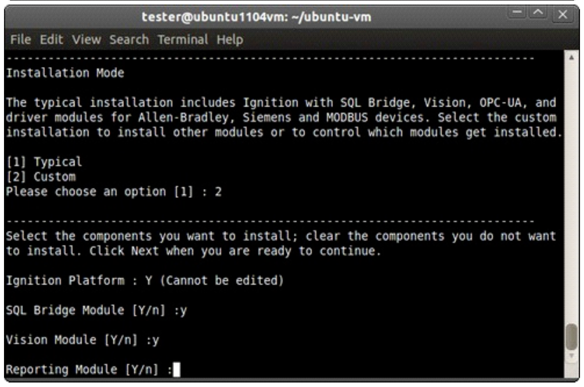
• In **Text mode** a list of all the modules are displayed one at a time. Type y to install the module, and type n to prevent the module from being installed.



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ✓ Ignition Platform
- ✓ SQL Bridge Module
- ✓ Vision Module
- ✓ Reporting Module
- ✓ OPC-UA Module
- ✓ Mobile Module
- ✓ Allen-Bradley Drivers Module
- ✓ Siemens Drivers Module

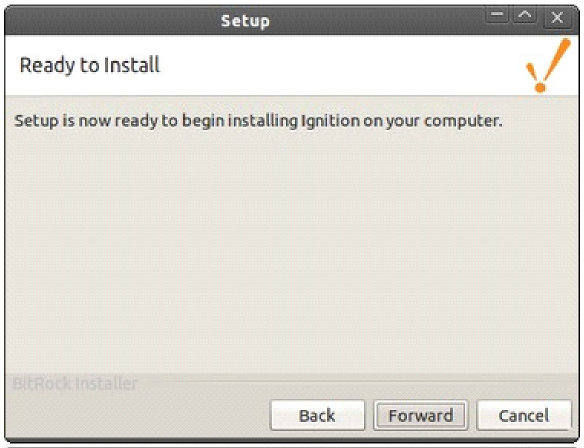
Bridges PLC data from OPC to SQL databases. Includes transaction group manager and SQL tag historian.


tester@ubuntu1104vm: ~/ubuntu-vm
File Edit View Search Terminal Help

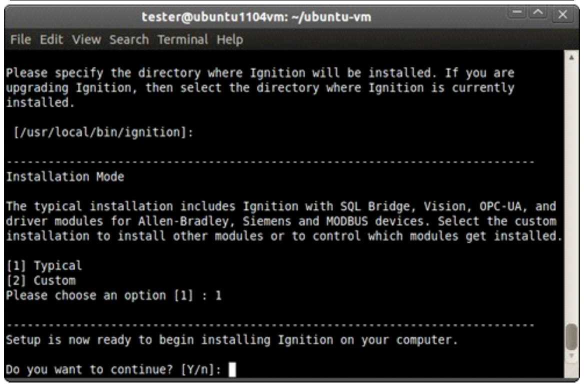
```
----------------------------------------------------------------------------
Installation Mode

The typical installation includes Ignition with SQL Bridge, Vision, OPC-UA, and
driver modules for Allen-Bradley, Siemens and MODBUS devices. Select the custom
installation to install other modules or to control which modules get installed.

[1] Typical
[2] Custom
Please choose an option [1] : 2

----------------------------------------------------------------------------
Select the components you want to install; clear the components you do not want
to install. Click Next when you are ready to continue.

Ignition Platform : Y (Cannot be edited)

SQL Bridge Module [Y/n] :y

Vision Module [Y/n] :y

Reporting Module [Y/n] :
```

Ignition is now ready to be installed. At this point, you can click the Back button (graphical mode) to change your selections. For the text mode, you can only abort the installation at this point by typing n.

13. Click **Forward** (graphical mode) or type y (text mode) to finish the installation.

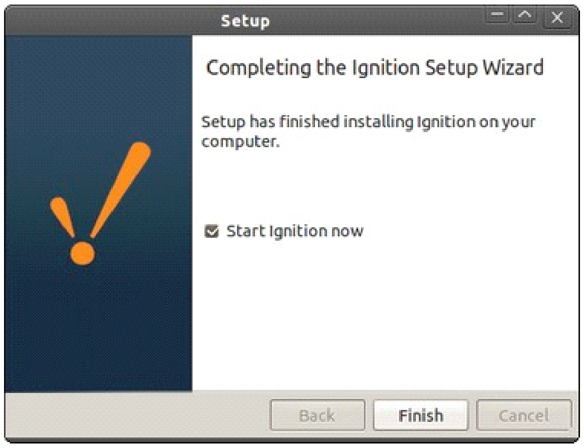


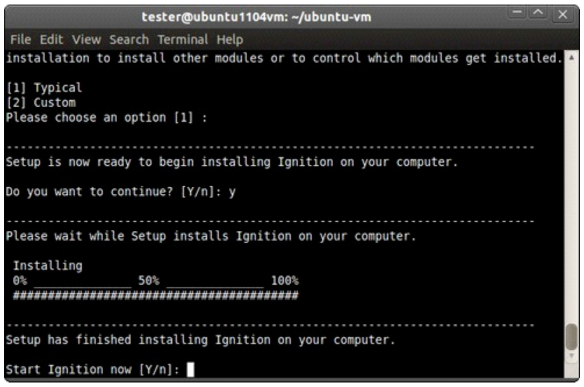
Setup is now ready to begin installing Ignition on your computer.


tester@ubuntu1104vm: ~/ubuntu-vm
File Edit View Search Terminal Help

```
Please specify the directory where Ignition will be installed. If you are
upgrading Ignition, then select the directory where Ignition is currently
installed.

 [/usr/local/bin/ignition]:

----------------------------------------------------------------------------
Installation Mode

The typical installation includes Ignition with SQL Bridge, Vision, OPC-UA, and
driver modules for Allen-Bradley, Siemens and MODBUS devices. Select the custom
installation to install other modules or to control which modules get installed.

[1] Typical
[2] Custom
Please choose an option [1] : 1

----------------------------------------------------------------------------
Setup is now ready to begin installing Ignition on your computer.

Do you want to continue? [Y/n]:
```

14. When the installation is complete, click **Finish** (graphical mode) or type y to start Ignition (text mode).
Ignition is started as a background process.


Setup
Completing the Ignition Setup Wizard
Setup has finished installing Ignition on your computer.
Start Ignition now
Back
Finish
Cancel


tester@ubuntu1104vm: ~/ubuntu-vm
File Edit View Search Terminal Help

```
installation to install other modules or to control which modules get installed.

[1] Typical
[2] Custom
Please choose an option [1] :

----------------------------------------------------------------------------
Setup is now ready to begin installing Ignition on your computer.

Do you want to continue? [Y/n]: y

----------------------------------------------------------------------------
Please wait while Setup installs Ignition on your computer.

 Installing
 0% ______________ 50% ______________ 100%
 #########################################

----------------------------------------------------------------------------
Setup has finished installing Ignition on your computer.

Start Ignition now [Y/n]:
```

Once the Ignition Gateway starts up, your web browser opens and brings you to the Gateway home page at http://localhost:8088.

Stop and Start Ignition

After installation, you can start and stop Ignition with the following commands:

```
/etc/init.d/ignition start
/etc/init.d/ignition stop
```

Ignition as a Service

When installing under Ubuntu, Ignition starts automatically whenever the computer reboots. If you wish to stop this behavior, you need to use the update-rc.d tool to remove the service (uninstalling Ignition also removes the service) as follows:

```
/etc/init.d/ignition stop
update-rc.d -f ignition remove
rm /etc/init.d/ignition
```

When installing under other Linux distributions, use that distribution's method to automatically start a program after reboot.
For example, this command auto-starts Ignition installed in a Fedora 15 system (run as root user):

```
chkconfig --level 2345 ignition on
```

Set the system PATH

For Ubuntu installations, the installation directory is automatically appended to the system PATH. You can start programs like the Gateway Control Utility from the command line without specifying a complete path to the installation directory.

Note that after installation, you need to close and reopen the command shell for the PATH change to take effect. For other Linux installations, you need to manually add `/usr/local/bin/ignition` (or your installation directory) to any script that can set the system PATH (such as `.profile` or `.bashrc`).

Automated Installation on Linux

You can automate Ignition installation across different machines using scripts. Ignition is then installed silently from a command shell without showing any user prompts.

Keep in mind that the installer cannot automatically start the Gateway after a silent installation. Use the `/etc/init.d/ignition start` command as shown in example below.

Command line example

```
sudo ./ignition-7.x.x-linux-x64-installer.run --mode unattended --prefix /somefolder/bin/ignition --
unattendedmodeui none
/etc/init.d/ignition start
```

Flags

`-- mode unattended` (ensures that no prompts appear during installation)
`-- prefix /somefolder/bin/ignition` (optional flag; if a value is set, then Ignition is installed in the specified folder, otherwise Ignition is installed in `/usr/local/bin/ignition` by default)
`-- serviceuser username` (allows a Linux system user to be installed (that is, a user that cannot log in to the OS))
`-- unattendedmodeui none` | minimal (the `none` flag does not display any sort of graphic during installation; the `minimal` flag displays a small progress bar and nothing else)

Installing using Debian Package Management

By using the package management tools you can quickly install Ignition on Ubuntu and other Debian-based systems. Debian package management is a powerful way to quickly install Linux programs and keep them up-to-date with minimal effort.

Note: that you need to be able to run as sudo to be able to install Ignition. Also, the machine where you want to install Ignition must be connected to the Internet so that the installer and updates can be installed automatically. If you are installing Ignition to a non Internet-connected machine, use the downloadable Linux installers instead (see the Download and Install on Linux section at the Top of this page for more details).

The Ignition Debian installers are dependent on an existing Java 6 or later installation through APT. Java 7 is highly recommended over Java 6. Oracle Java Runtime Environment (JRE) version 7 is officially supported by Inductive Automation, although Ignition runs well in most cases using OpenJDK 7.

The installer installs files in the following locations:

`/usr/local/bin/ignition` - contains binaries and startup scripts
`/var/lib/ignition/data` - contains application-generated files, temporary files and the internal database
`/var/lib/ignition/user-lib` - contains modules and JDBC jars
`/var/log/ignition` - contains the wrapper.log and other log files
`/etc/ignition` - contains configuration files. Symbolic links to these files are created in `/var/lib/ignition/data`.

When installing Ignition, you can choose whether to use the graphical package management tools, or install completely from the command-line. With either option, Linux automatically downloads the correct 32-bit or 64-bit installer depending on your installed system architecture.

Note that the instructions below are for an Ubuntu Linux system, but other Debian systems will contain the same tools, although they may be under different menus.

Graphical Installation

1. Download the Inductive Automation public key file from http://archive.inductiveautomation.com/ia.public.key

   or on the command-line, run
   `wget http://archive.inductiveautomation.com/ia.public.key`

2. Within Ubuntu, navigate to **System > Administration > Synaptic Package Manager**.

3. Within Synaptic Package Manager, navigate to **Settings > Repositories**.

4. Navigate to the Authentication tab. Click on **Import Key File** and select the downloaded ia.public.key file.

5. Navigate to the Other Software tab and click on the **Add** button. Add the following text to the text box:
   `dep ://archive.inductiveautomation.com/ignition non-free`

6. If you want to add the Ignition beta repository, click the **Add** button again and add this text to the text box:
   `dep http://archive.inductiveautomation.com/apt ignition-beta non-free`

7. Be sure to uncheck the check boxes next to repositories ending with **Source Code**, as Ignition does not supply source code with the repositories.

8. Click the **Close** button. Within Synaptic Package Manager, click the **Reload** button. You can now type **ignition** into the Quick Filter box and see the latest available Ignition repositories.

9. Right-click on the Ignition repository that you want to install, and select **Mark for Installation**. Then click the **Apply** button at the top. Ignition will be automatically downloaded and started. Navigate to http://localhost:8088 to log into the Ignition Gateway.

Command-line Installation

1. Run
   `wget http://archive.inductiveautomation.com/ia.public.key`

2. Run
   sudo apt-key add `ia.public.key`

3. Copy
   `/etc/apt/sources.list`

   to
   `/etc/apt/sources.list.bak`

4. Edit
   ```
   /etc/apt/sources.list
   ```

   and add the following lines (the ignition-beta line is optional):
   ```
   deb http://archive.inductiveautomation.com/apt ignition non-free
   deb http://archive.inductiveautomation.com/apt ignition-beta non-free
   ```

5. Run
   ```
   $sudo apt-get update
   ```

   to update the download list within the APT utility.

6. Run
   `sudo apt-get install ignition`

   to install the latest stable Ignition version or run

   ```
   sudo apt-get install ignition-beta
   ```

   to install the latest beta Ignition version.

   If you want to run the **Gateway Command Utility** or **gwcmd** right after installation, open a command shell in your home folder and run
   `source .profile`

   This forces your command shell to reload the `.profile` script, which has been updated with a path to the Ignition executable during installation.

Upgrades

During an upgrade, the Ignition internal database, configuration files, and any custom installed modules are not changed. Whenever a new version of Ignition is released, the online Debian repositories also get updated with the latest version.

Your Linux environment will list Ignition in the list of package upgrades when a new version is available. As with the installation, you can choose whether to upgrade using a graphical environment or using the command-line.

Graphical Upgrade

Ignition will appear as an entry under your system's Update Manager under the **Other Updates** section. If you choose to install updates using the Update Manager, Ignition will be upgraded at the same time as other packages.

You can also manually upgrade Ignition using Synaptic Package Manager. Locate **ignition** or **ignition-beta** in the list using the Quick Filter.

Ignition can also be found in the World Wide Web section. Right-click and select **Mark for Upgrade**. Then click **Apply** at the top. The latest version of Ignition will be downloaded and installed.

Command-line Upgrade

To upgrade all installed packages, including Ignition, run
`sudo apt-get upgrade`

To only upgrade Ignition, run
`sudo apt-get install ignition`

or
`sudo apt-get install ignitionbeta`

The APT utility recognizes that Ignition is already installed, and performs the upgrade.

Uninstalling Ignition on Linux

Before you uninstall Ignition...
Be sure to back up your Gateway and unactivate your Gateway license.

During a new Ignition installation, an uninstaller executable is created.

When Ignition is uninstalled, the settings database and folder in `/data` is backed up to `/data_<current date>`, and modules and user-supplied JDBC jars in `/user-lib` are backed up to `/user-lib_<current date>`. Also, the Ignition service is removed from Ubuntu Linux installations automatically.

Linux – Using Downloaded Installer

Ignition 7.3 or later installations on Linux contain an uninstaller executable.

To run the uninstaller,

1. Open a command shell and navigate to `/user/local/bin/ignition` (or your installation folder).

2. As `root` or `sudo`, run
   `./uninstall`

Linux Ignition installations before 7.3 used a zip file that did not form an installation. Since these installations were never created with an installer executable, no uninstaller executable was ever generated. This is true even if using a 7.3 or later installer executable to upgrade an installation from before 7.3. For installations before 7.3, you must manually remove the Ignition folders using the commands below.

```
/etc/init.d/ignition stop
*Ubuntu only* update-rc.d -f ignition remove
rm /etc/init.d/ignition
rm -rf /usr/local/bin/ignition
*Recommended* mv /var/lib/ignition/data /var/lib/ignition/data_<current date>
*Recommended* mv /var/lib/ignition/user-lib /var/lib/ignition/user_lib_<current date>
*Recommended* mv /etc/ignition /etc/ignition_<current date>
rm -rf /var/log/ignition
```

Linux Uninstallation – Using Ubuntu Package Management

Installation using the Ubuntu Package Manager offers the choice to either remove the installation while keeping the configuration files in `/etc`, or to completely remove/purge the installation from the system.

If you choose to remove the installation and then reinstall Ignition, Ubuntu will need to download Ignition again.

To simply remove the installation,

1. Open a command shell.
2. Run
   `sudo apt-get remove ignition` (for stable installations) or
   `sudo apt-get remove ignition-beta` (for beta installations)

To completely remove/purge the installation,

1. Open a command shell.
2. Run
   `sudo apt-get purge ignition` (for stable installations) or
   `sudo apt-get purge ignition-beta` (for beta installations)

Next ...

- Activating Over Internet
- Activating Manually
- Upgrading Ignition on Linux

Upgrading Ignition on Linux

Before you upgrade Ignition...

You should backup your Gateway, see Making Gateway Backups.

Also, you need to be able to run as ROOT (prefix everything with `sudo`, run `sudo su`, or login as `root` user) to perform the upgrade.

Unlike some Linux operations, the Ignition Linux upgrade process is designed to be as simple as possible. Note that while Ignition is currently optimized for Ubuntu Linux, the upgrade process will work for any Linux distribution.

The Ignition installer also doubles as an upgrader, thus upgrading Ignition is very similar to downloading a new version and installing from scratch. You can see your current Ignition version by looking in the **Stat us** page of the Gateway.

INDUCTIVE UNIVERSIT

Upgrading Ignition on Linux

Watch the Video

Tip Sheet

Upgrading from 7.6 or earlier?

Please read this upgrade guide: Igniti on 7.7 Upgrade Guide

To upgrade Ignition on Linux

1. Simply download the Linux executable installer from our website, and double-click on the executable installer.
   Note: Make sure you download the 32-bit installer for a 32-bit Linux installation, and the 64-bit installer for a 64-bit Linux installation. A 32-bit installer does not launch in a 64-bit system and vice versa.
2. After downloading the installer, open a command shell and navigate to the installer executable.
3. Run
   ```
   chmod 744 *
   ```
4. Then run
   ```
   ./ignition_X.X.X-installer.run
   ```
   If you are running the installer in a shell in a graphical environment, the graphical installer opens automatically.
   If you are running the installer in a headless Linux installation, or through a SSH shell, the text installer opens automatically.
   If you want to run the text installer in a graphical environment, add `--mode text` to the end of the above command.

   The installer starts running.
5. Agree to the licensing terms and click **Forward**.
   The Installation Directory window is displayed.
6. Click **Forward** to accept the default installation directory.
   Linux does not have the concept of a central registry like Windows systems do. Therefore, you must provide the location of the existing Ignition installation. By default, this value is `/usr/local/bin/ignition`, unless you have installed Ignition in another location.
7. In the **Question** window, click **Yes** to continue.
   This Question window is displayed when the upgrader was able to locate Ignition in the specified folder. After you click Yes, the Upgrade Mode window is displayed.
8. You choose either:
   No, just upgrade currently installed modules
   The Ignition base installation and the existing modules are upgraded. This is the choice for most scenarios.
9. Or you can choose:
   Yes, let me choose which modules to add or remove
   The Select Components window is displayed and you can then add a module that was not part of the original installation or uninstall/remove a currently installed module.

   In graphical mode, you can view a brief description of each module by clicking on the module name.
   Selecting the check box next to a module, installs the module as part of the upgrade.
   Clearing the check box next to a module, uninstalls the module during the upgrade.

   In text mode, a list of all the modules is displayed, one at a time. Type **y** to install or upgrade the module, and type **n** to uninstall the module.

Ignition is ready to be upgraded. At this point, you can click **Back** (graphical mode) to change your selections, or you can only abort the upgrade at this point (text mode) by typing **n**.

10. To finish the upgrade, click **Next** (graphical mode) or type **y** (text mode).
11. When the upgrade is complete, click Finish (graphical mode) or type y to start Ignition or n to exit (text mode).
    If the box for **Start Ignition now** was selected, Ignition starts as a background process.
12. Use a web browser and go to http://localhost:8088 to confirm that your Gateway is running.
    For instructions on how to stop or start Ignition, see Installing Ignition on Linux.

Automated Upgrading

You can silently upgrad Ignition from a command shell without showing any user prompts. Remember to make a Gateway backup before performing any type of upgrade. Also, the installer cannot automatically start the Gateway after a silent upgrade.

Use the `/etc/init.d/ignition start` command as shown below.

Command line example:

```
gwcmd -b /var/backups/mybackup.gwbk

sudo ./ignition-7.x.x-linux-x64-installer.run --mode unattended --unattendedmodeui none --autoupgrade true


/etc/init.d/ignition start
```

Flags:
`-- mode unattended` (ensures that no prompts appear during installation)
`-- unattendedmodeui none | minimal` (the `none` flag does not display any sort of graphic during installation; the `minimal` flag displays a small progress bar and nothing else, but only if you are working in a graphical system)
`-- autoupgrade true` (runs the installer in upgrade mode)

Next ...

- add the link here

Installing Ignition on Mac OS X

The following feature is new in Ignition version **7.8.0**
Click here to check out the other new features

⚠ 7.8 introduced an OS X installer. Instructions for using the ZIP files can be found on our Installing Ignition on Mac OS X - ZIP file page

INDUCTIVE UNIVERSITY

Installing Ignition on Mac OS X

Watch the Video

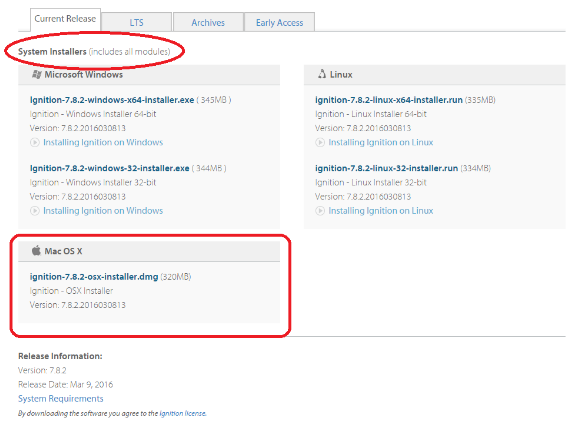
Before you install Ignition, make sure you meet the minimum system requirements:

- Java 8+ (server), Java 6+ (client)
- Dual-Core Processor (32 or 64 bit)
- 4GB RAM
- 10GB free HD space

Additionally, running the installer will require **root** or **administrator** privileges

To download and install on Mac OS X

1. Go to the Inductive Automation website at www.inductiveautomation.com
2. Select **Download > Current Release**.
3. On the Ignition Downloads page, scroll down to **System Installers** which contains all the download zip files.



Current Release | LTS | Archives | Early Access

System Installers (includes all modules)

Microsoft Windows

Ignition-7.8.2-windows-x64-installer.exe (345MB)
Ignition - Windows Installer 64-bit
Version: 7.8.2.2016030813
Installing Ignition on Windows

Ignition-7.8.2-windows-32-installer.exe (344MB)
Ignition - Windows Installer 32-bit
Version: 7.8.2.2016030813
Installing Ignition on Windows

Linux

ignition-7.8.2-linux-x64-installer.run (335MB)
Ignition - Linux Installer 64-bit
Version: 7.8.2.2016030813
Installing Ignition on Linux

ignition-7.8.2-linux-32-installer.run (334MB)
Ignition - Linux Installer 32-bit
Version: 7.8.2.2016030813
Installing Ignition on Linux

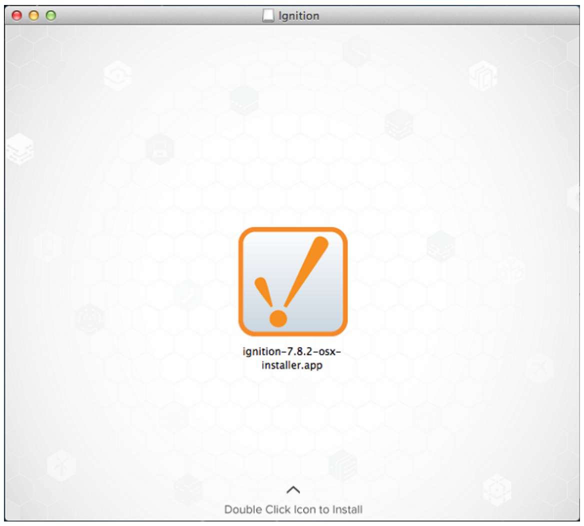
Mac OS X

ignition-7.8.2-osx-installer.dmg (320MB)
Ignition - OSX Installer
Version: 7.8.2.2016030813

Release Information:
Version: 7.8.2
Release Date: Mar 9, 2016
System Requirements
By downloading the software you agree to the Ignition license.

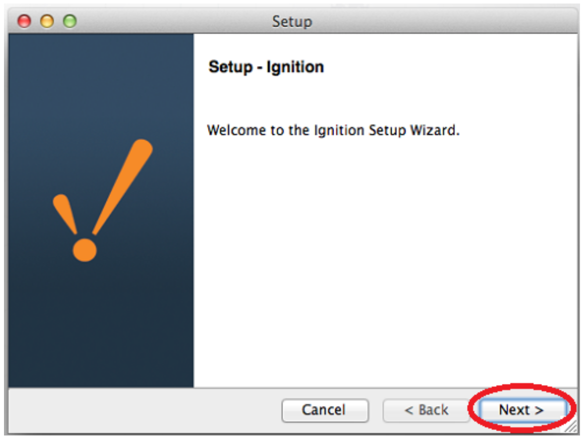
4. Download the **Ignition-x.x.x-osx-installer.dmg** file.
5. Download and install Java 8 if it is not already installed.
   Go to
   http://www.oracle.com/technetwork/java/javase/downloads/index.html

   Click on the download button under JRE. Accept the license agreement and download jre-8uXX-macosx-x64.dmg. Once downloaded, run the installer and follow the instructions to install Java.
6. Double click on the DMG. You'll see a window appear that contains a **ignition-x.x.x-osx-installer.app** file


Ignition
ignition-7.8.2-osx-installer.app
Double Click Icon to Install

7. Double click on the APP file to start the installer. When prompted, enter your password for elevated installation privilege.

8. The Setup window welcomes you to the Ignition Setup Wizard.

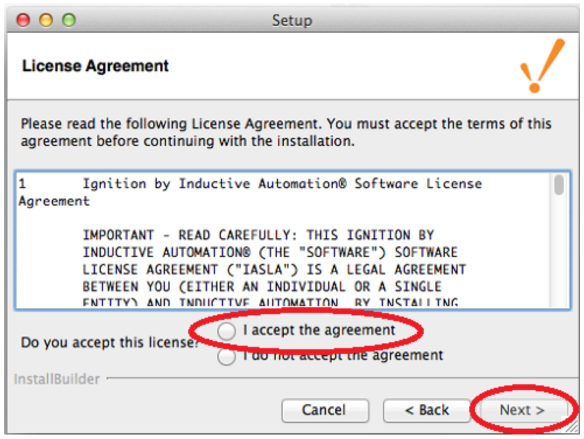


Setup - Ignition

Welcome to the Ignition Setup Wizard.

9. Click **Next**

10. Click the **I accept the agreement** button and click **Next**.



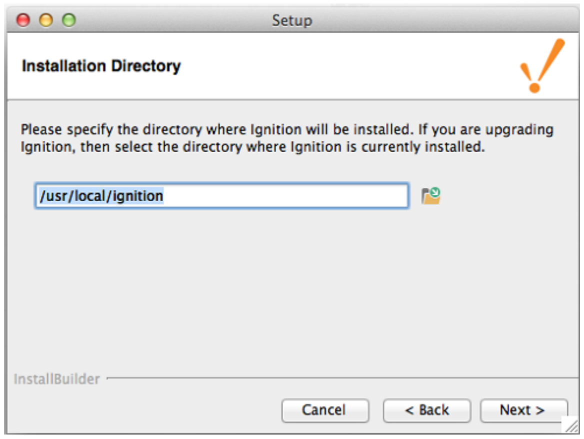
License Agreement

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

```
1       Ignition by Inductive Automation® Software License
Agreement

        IMPORTANT - READ CAREFULLY: THIS IGNITION BY
        INDUCTIVE AUTOMATION® (THE "SOFTWARE") SOFTWARE
        LICENSE AGREEMENT ("IASLA") IS A LEGAL AGREEMENT
        BETWEEN YOU (EITHER AN INDIVIDUAL OR A SINGLE
        ENTITY) AND INDUCTIVE AUTOMATION, BY INSTALLING
```

Do you accept this license? I accept the agreement / I do not accept the agreement

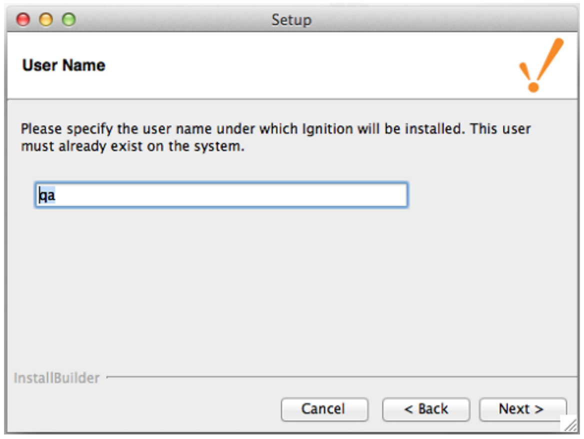
11. The Installation Directory window is displayed. By default, Ignition is installed under /usr/local /ignition. This is usually a good choice.


Installation Directory

Please specify the directory where Ignition will be installed. If you are upgrading Ignition, then select the directory where Ignition is currently installed.

/usr/local/ignition

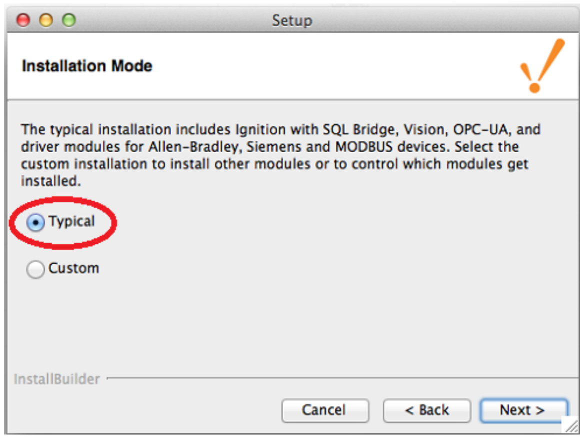
12. Ignition will need a user name to be installed under. The user needs to already exist on the system.


User Name

Please specify the user name under which Ignition will be installed. This user must already exist on the system.

qa

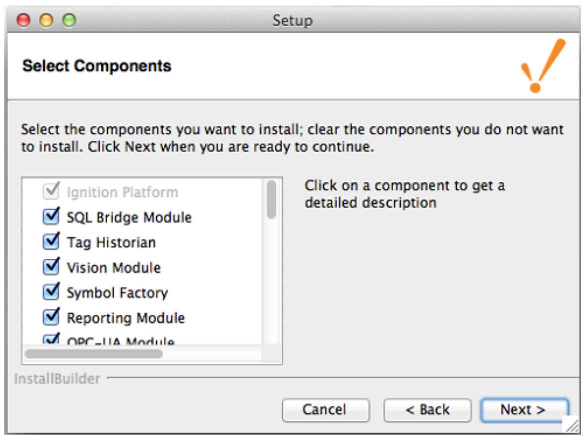
13. Select either the **Typical** or **Custom** mode.
    The **Typical** installation includes Ignition with all of the HMI/SCADA modules.
    The **Custom** installation lets you control and individually select the modules that you want installed.


Installation Mode

The typical installation includes Ignition with SQL Bridge, Vision, OPC-UA, and driver modules for Allen-Bradley, Siemens and MODBUS devices. Select the custom installation to install other modules or to control which modules get installed.

[✓ Typical]
Custom

14. Click **Next**.
    The Setup window is displayed.
    If you selected **Custom** mode, in the Setup window you can select the check box next to the modules you want to install as part of the Ignition installation. To see a brief description of the module, click on the module name.



Select Components

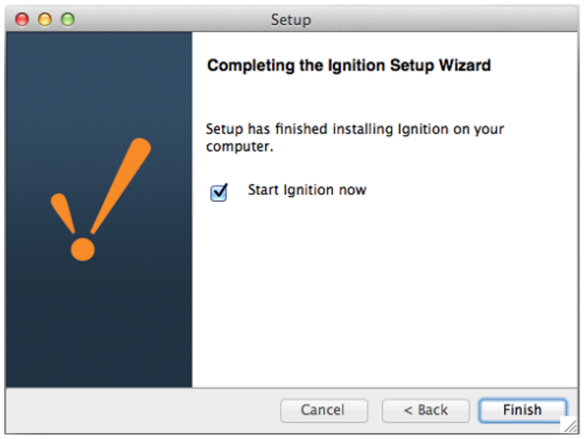
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [✓] Ignition Platform
- [✓] SQL Bridge Module
- [✓] Tag Historian
- [✓] Vision Module
- [✓] Symbol Factory
- [✓] Reporting Module
- [✓] OPC-UA Module

Click on a component to get a detailed description

Setup Wizard shows Ignition is now ready to install.
If you decide to change your selections, click **Back** to go back and make your changes.

15. Click **Next.**
16. After Setup Wizard shows Ignition installation is finished, make sure that the check box for **Start Ignition now** is selected, then click **Finish**.



Completing the Ignition Setup Wizard

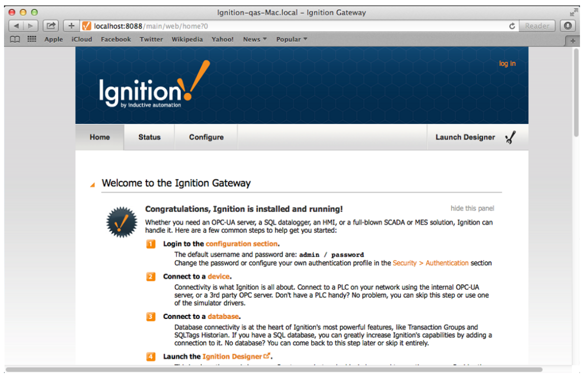
Setup has finished installing Ignition on your computer.

[✓] Start Ignition now

A splash screen informs you that Ignition is starting. This may take a few minutes to finish.

That's it! Ignition Installation is complete and the Gateway is then launched.

You can now open a web browser and go to the Gateway home page at http://localhost:8088.


Ignition-qas-Mac.local – Ignition Gateway
localhost:8088/main/web/home?0
Apple iCloud Facebook Twitter Wikipedia Yahoo! News Popular
Ignition by inductive automation
log in
Home
Status
Configure
Launch Designer
hide this panel

Welcome to the Ignition Gateway

Congratulations, Ignition is installed and running!

Whether you need an OPC-UA server, a SQL datalogger, an HMI, or a full-blown SCADA or MES solution, Ignition can handle it. Here are a few common steps to help get you started:

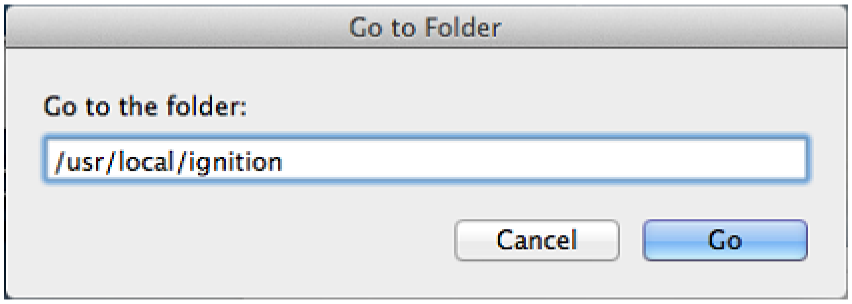
1. **Login to the configuration section.**
   The default username and password are: `admin / password`
   Change the password or configure your own authentication profile in the Security > Authentication section
2. **Connect to a device.**
   Connectivity is what Ignition is all about. Connect to a PLC on your network using the internal OPC-UA server, or a 3rd party OPC server. Don't have a PLC handy? No problem, you can skip this step or use one of the simulator drivers.
3. **Connect to a database.**
   Database connectivity is at the heart of Ignition's most powerful features, like Transaction Groups and SQLTags Historian. If you have a SQL database, you can greatly increase Ignition's capabilities by adding a connection to it. No database? You can come back to this step later or skip it entirely.
4. **Launch the Ignition Designer.**

Uninstalling Ignition on OS X

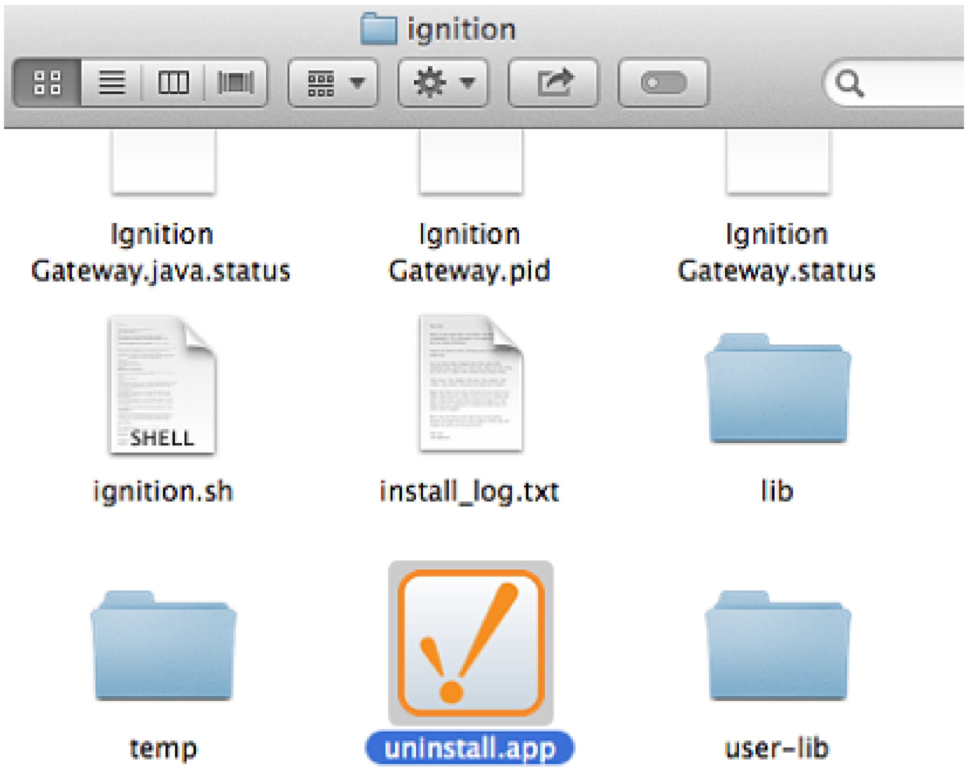
Before you uninstall Ignition...
Be sure to back up your Gateway and unactivate your Gateway license.

During a new Ignition installation, an uninstaller APP is created.

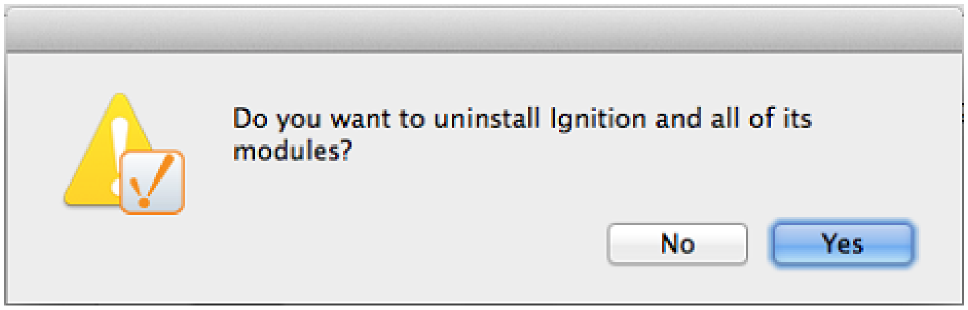
1. Navigate to /usr/local/ignition



2. Run the uninstall.app file



3. The uninstaller wizard guides you through the uninstallation process.



Do you want to uninstall Ignition and all of its modules?

Next ...

- Activating Over Internet
- Activating Manually
- Upgrading Ignition on Mac OS X

Upgrading Ignition on Mac OS X

Before you upgrade Ignition...

You should backup your Gateway, see Making Gateway Backups.

To upgrade Ignition on Mac OS X

1. From Gateway, go to **Configure > Backup / Restore**.
2. Click on **Download Backup** to back up the entire project from the existing installation.
   The `.gwbk` backup file is created in the **Downloads** folder.
3. Shut down Ignition by typing the following from the command-line:

   ```
   /usr/local/bin/ignition stop
   ```

4. Move the existing installation folder somewhere else.
5. Unzip the new installation and place it in the location of the old installation.
6. Start the new installation.

   ```
   /usr/local/bin/ignition start
   ```

7. Go to **Configure > Backup / Restore** and click **Restore** to perform the restoration.

Next ...

- add link here

Upgrading Ignition on Mac OS X

Watch the Video

Tip Sheet

Upgrading from 7.6 or earlier?

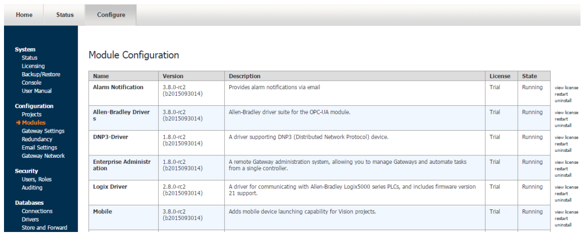
Please read this upgrade guide: Igniti
on 7.7 Upgrade Guide

Installing or Upgrading a Module

All module configuration is done from Gateway in the **Configuration > Modules** section. From here you can add, remove, and restart modules. Modules integrate their settings into the Gateway configuration tree, and therefore do not offer settings in this section.

Installing or Upgrading a Module

Watch the Video



Module Configuration

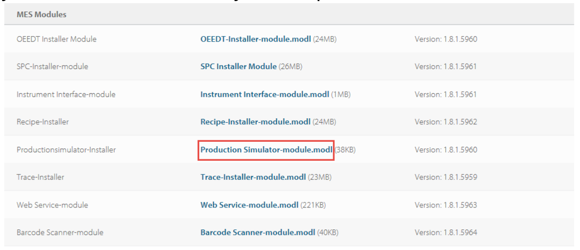
Name	Version	Description	License	State	
Alarm Notification	3.8.0-rc2 (b2015093014)	Provides alarm notifications via email	Trial	Running	view license restart uninstall
Allen-Bradley Drivers	3.8.0-rc2 (b2015093014)	Allen-Bradley driver suite for the OPC-UA module.	Trial	Running	view license restart uninstall
DNP3-Driver	1.8.0-rc2 (b2015093014)	A driver supporting DNP3 (Distributed Network Protocol) device.	Trial	Running	view license restart uninstall
Enterprise Administration	1.8.0-rc2 (b2015093014)	A remote Gateway administration system, allowing you to manage Gateways and automate tasks from a single controller.	Trial	Running	view license restart uninstall
Logix Driver	2.8.0-rc2 (b2015093014)	A driver for communicating with Allen-Bradley Logix5000 series PLCs, and includes firmware version 21 support.	Trial	Running	view license restart uninstall
Mobile	3.8.0-rc2 (b2015093014)	Adds mobile device launching capability for Vision projects.	Trial	Running	view license restart uninstall

To install or update a module

1. On the **Configuration > Modules** page of the Gateway, go to the bottom on the list, find the orange arrow, and click the **Install or Upgrade a Module** link.

2. Click **Choose File**, select a `.modl` file that you have previously downloaded.

   Go to the Inductive Automation downloads page. Search for the module you want and save it to your Downloads folder on your computer.

MES Modules		
OEEDT Installer Module	OEEDT-Installer-module.modl (24MB)	Version: 1.8.1.5960
SPC-Installer-module	SPC Installer Module (26MB)	Version: 1.8.1.5961
Instrument Interface-module	Instrument Interface-module.modl (1MB)	Version: 1.8.1.5961
Recipe-Installer	Recipe-Installer-module.modl (24MB)	Version: 1.8.1.5962
Productionsimulator-Installer	Production Simulator-module.modl (38KB)	Version: 1.8.1.5960
Trace-Installer	Trace-Installer-module.modl (23MB)	Version: 1.8.1.5959
Web Service-module	Web Service-module.modl (221KB)	Version: 1.8.1.5963
Barcode Scanner-module	Barcode Scanner-module.modl (40KB)	Version: 1.8.1.5964

   Note: If you are using Internet Explorer, IE changes the file extension to a '.zip' file. Rename '.zip' to '.modl' (I.e., Production Simulator-module.modl).

   Go back to the **Install or Upgrade Module** screen and browse your Downloads folder for your module, select it and click **Open**. Ignition will copy the path to the module to the Browse field and all that's left for you to do is click **Install**.
3. Click **Install**.
   When the page reloads you can now see the module you installed in the list of modules.

To uninstall a module

1. On the **Configuration > Modules** page of the Gateway, go to the module you want to uninstall.

2. Click the **uninstall** link on the far-right next to the module entry in the table.
   Another page opens asking you to confirm.

3. Click **Confirm**.
   The module is no longer installed and is removed form the list.

Modules are hot-swappable, this means these actions can be performed while the system is running. Furthermore, the isolated nature of modules ensures that performing one of these actions only affects that particular module, and any modules which depend on it. For example, uninstalling the SQL Bridge module will not affect any running Vision module clients.

Next ...

Gateway Setup

The Ignition Gateway is the primary software service that drives everything in Ignition. It is a single application that runs an embedded web server, connects to data, connects to PLCs, executes modules, communicates with clients, and more.

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 a computer that you can log into, you access it from `http://localhost:8088`.

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 **Configure** section of the Gateway. The different broad categories of what you can configure are as follows:

- **System** (Status, Licensing, Backup/Restore, Console, and User Manual)
- **Configuration** (Projects, Modules, Gateway Settings, and Redundancy)
- **Security** (Users, Roles, and Auditing)
- **Database** (Connections, Drivers, and Store and Forward)
- **Alarming** (General, Journal, Schedules, Notification, On-Call Rosters)
- **Tags** (Realtime and History)
- **OPC-UA** (Certificates, Settings, Devices)
- **OPC Connections** (Servers and Quick Client)
- **Mobile** (Settings)

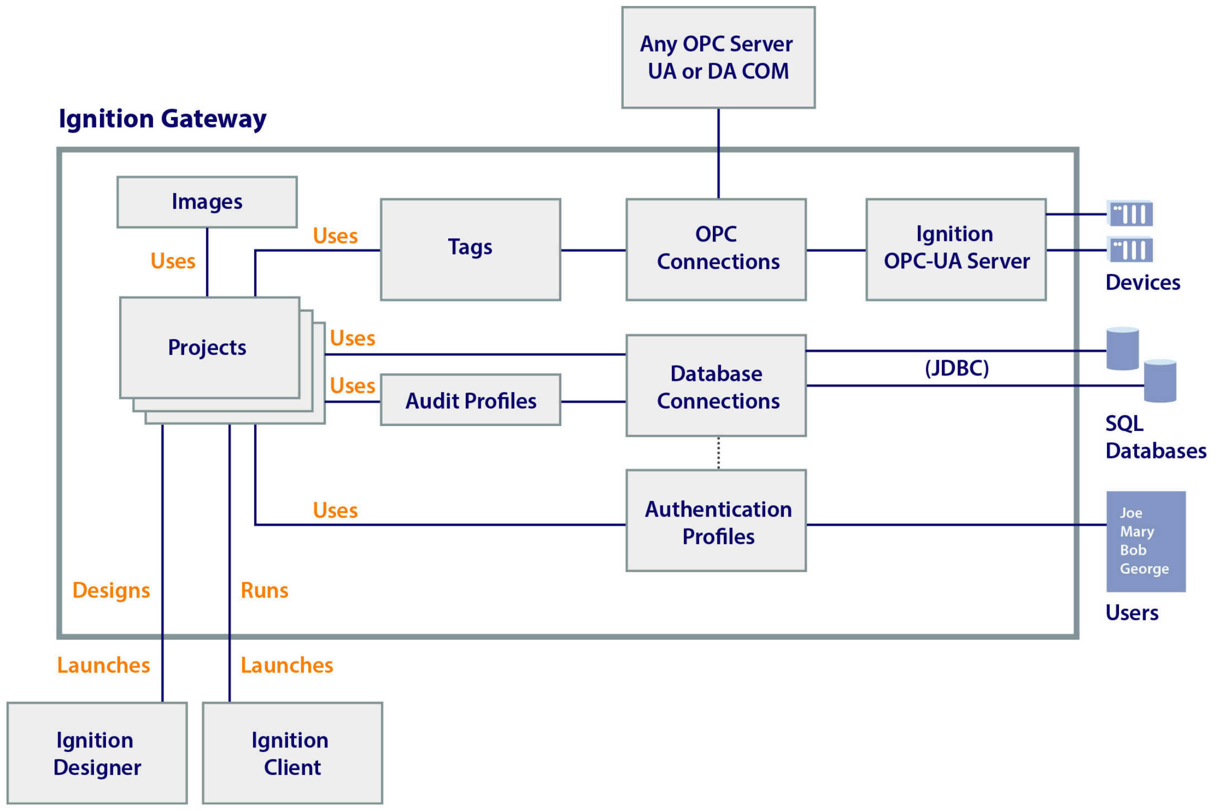
Depending on what modules you have installed, other categories can be available.

Gateway's 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 Notification Settings
- Tags Provider Settings
- OPC Connectivity
- Mobile Settings
- Projects - the most important part!

The following figure shows the different parts of the Gateway and how projects work within the Gateway.



In this section ...

Gateway Basics

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, access important functions from the Gateway Control Utility, and customize the Gateway's homepage to fit your needs.

On this page ...

Gateway Basic Settings

After you launch Gateway, you can define the high-level settings that apply to the entire Gateway by going to the **Configuration > Gateway Settings** page. 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 Roles	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.
Global Resource Protection	The roles required to protect or modify a protected global resource such as an alarm pipeline. If blank, anyone that can log into the Designer can protect global resources.
Allow User Admin	Allows the administration of the Gateway's system user source from the Designer and Client. Unless this is enabled, the Vision module's **User Management Component** is prevented from altering the Gateway's system user source.
Use SSL	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.
Persist Alarms	Whether or not alarm properties such as acknowledgment should persist across Gateway restarts.
Update Notifications	When enabled a notification bar is displayed at the top of the configuration page when a new version of Ignition is available for download. This only works if your server is connected to the internet.
Web-Launch Settings	

Launch Link Script Policy	Controls how the HTML that launches Clients and Designer functions. If set to **JavaScript**, the links use `javascript` to attempt to launch directly using the Java browser plugin. If set to **Direct**, the links are direct links to the `*.jnlp` files that launch the Client or Designer.
Allowed JREs	A semicolon separated list of allowed JREs for web-launching Clients and Designers. For example: `1.7;1.8` allows Java 7 and Java 8.
Designer Memory	The maximum amount of memory that the Designer has access to.
Disable Direct3D/ Disable DirectDraw	These properties affects launched Clients and Designers on Windows OS only. These flags control whether or not the Java Swing windowing subsystem may use Direct3D and/or DirectDraw. Disabling these features can incur a performance penalty, but might be required for some video cards that have faulty DirectX drivers.
Local 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 style scheduling string representing when to make the backups.
Retentions 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.
Recieive 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.

Schduled Backups

The **Scheduled Backup Settings** (shown in the above table) 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. 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. 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.

Here are some examples:

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 * *	Mon 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.

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

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 Monitoring

You can monitor the Ignition Gateway from the **Status** section of the Gateway or from the Gateway Control Utility (**GCU**).

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

Status about ...	
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.
Modules	A list of installed modules, their status, as well information about their version and current license state.
Tag	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.
Database Connections	This important section shows your database connections, their status, and their current load. Each status panel has information about the connection such as queries/second, total queries, and the average query duration.
Store & Forward	Shows you what each of your S&F engines are doing. They show the number of pending and quarantined records that exist in the various stages of the S&F engines, as well as the throughput of records going through each stage. Note that if the final sink (the Database Storage) is available, data will jump directly from the memory buffer to the database storage, skipping the local cache. See also: Controlling Quarantine Data.
OPC Connections	Shows status details about all of your OPC connections and any subscriptions you have made through these connections. You can view the status of any connection as well as find details for trouble shooting when the status is bad. Statistics on the number of reads, writes, and updates are also kept.
Sessions	Shows status details about all of the Designer and Client sessions that are communicating with this Gateway. You can see information such as their subscriptions, user credentials, and so on.
Gateway Scripts Status	Shows status details about all the scripts that are running in Gateway. You can see information such as their execution status, whether the scripts are running or not, and so on.
Alarm Piplines	Shows the status details about the alarm piplines. You can see the status of an alarm and where the alarm is in the pipeline.
Ignition OPC-UA Server (Requires OPC-UA Module)	This section has two tabs: **Devices** and **Server Statistics**. The **Devices** tab lists of all configured devices, the status for each device, as well as details about the driver that device is using. The **Server Statistics** tab is a set of basic performance statistics accumulated for the duration the server has been running as well as a list of subscriptions and their corresponding subscribed nodes that the server currently knows about.

Starting and Stopping the Gateway

After installation, the Gateway starts automatically. You can manually stop and start the service. Depending on the platform you are using, the following is how you can manually start and stop the Gateway.

Windows

Access the service control utility at **Control Panel > Administrative Tools > Services**. Then use the **Ignition Gateway** service to control the run state of the Gateway.

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:

```
>./ignition.sh start
```

Troubleshooting the Gateway

The Gateway **Console** provides a wealth of information about the running state of the Gateway. These information are mainly used for advanced troubleshooting, and are not often consulted in normal operation.

To access the Console go to, **System > Console** in the **Configure** section of the Gateway. The **System Console** page is displayed showing the following tabs:

Logs
Of all of the tabs in this section, the Logs tab is the most useful for system administrators. Logs shows the most recent output of Gateway **Loggers** - units in the Gateway application that output information.

Levels
The Levels tab shows all of the registered system **Loggers**, and the **Level** of detail that they should record.

Threads
The Threads tab shows the current state of all system threads.

Memory
The Memory tab provides a breakdown of how much memory is being used by Ignition.

Execution
The Execution tab show a list of all registered "executors"- tasks that perform repeat operations.

Advanced
The Advanced tab is the entry point to the **Raw Settings Viewer**. The Raw Settings Viewer allows advanced users to make queries against the

internal database for advanced troubleshooting purposes. There is
potential to really do some damage to the Ignition installation if one isn't careful. Use extreme caution when working directly with the internal database.

In this section ...

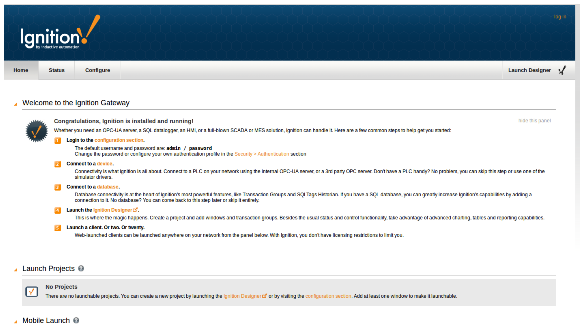
Gateway Webpage

Launching Gateway

The Ignition Gateway is a web server. When it is running, you access it through a web browser.
For example, if you are logged into the computer that you installed Ignition on, open up a web browser and go to the address:
http://localhost:8088

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 Gateway and bring up the Gateway Homepage as pictured here.



Welcome to the Ignition Gateway

Congratulations, Ignition is installed and running! hide this panel

Whether you need an OPC-UA server, a SQL datalogger, an HMI, or a full-blown SCADA or MES solution, Ignition can handle it. Here are a few common steps to help get you started:

1. **Login to the configuration section.**
   The default username and password are: `admin / password`
   Change the password or configure your own authentication profile in the Security > Authentication section
2. **Connect to a device.**
   Connectivity is what Ignition is all about. Connect to a PLC on your network using the internal OPC-UA server, or a 3rd party OPC server. Don't have a PLC handy? No problem, you can skip this step or use one of the simulator drivers.
3. **Connect to a database.**
   Database connectivity is at the heart of Ignition's most powerful features, like Transaction Groups and SQLTags Historian. If you have a SQL database, you can greatly increase Ignition's capabilities by adding a connection to it. No database? You can come back to this step later or skip it entirely.
4. **Launch the Ignition Designer.**
   This is where the magic happens. Create a project and add windows and transaction groups. Besides the usual status and control functionality, take advantage of advanced charting, tables and reporting capabilities.
5. **Launch a client. Or two. Or twenty.**
   Web-launched clients can be launched anywhere on your network from the panel below. With Ignition, you don't have licensing restrictions to limit you.

Launch Projects

No Projects
There are no launchable projects. You can create a new project by launching the Ignition Designer or by visiting the configuration section. Add at least one window to make it launchable.

Mobile Launch

The first time you go to the Gateway Homepage, it shows you 5 common steps to help you get started.

Gateway Webpage

The Ignition Gateway is a web server and when it is running you access it from a web browser. In a web browser, type http://localhost:8088 to display the Gateway homepage. The Gateway homepage has four buttons across the top of the page that lead you to the key sections of the server: Home, Status, Configure, and Launch Designer.

Home

The **Home** button takes you to a page that lets you do the following:

- **Launch Projects** in a client
- **Mobile Launch** a project on a mobile device
- See the **Transaction Groups** that are running for your project
- See a list of all the connected **Devices**
- Download the **Native Client Launchers**
- See a list of all **Database Connections**

Status

The **Status** button provides in-depth information about the status of the different parts of the Ignition system. The menu on the left of the Status page takes you to the following different parts for detailed information:

- Overview
- Modules
- Redundancy
- Tags
- Database Connections
- Store & Forward
- OPC Connections
- Sessions
- Gateway Scripts
- Gateway Network
- Gateway Tasks
- Alarm Pipelines
- OPC-UA Server
- Reports

On this page ...

Gateway Webpage

Watch the Video

To Continue on the step-by-step simple workflow...

After you **Launch Gateway**, described on the left, click the link below for Step 2 to continue.

Step 1. **Launch the Gateway**

Step 2. **Connect to a Device**

Step 3. **Connect to a Database**

Step 4. **Launch the Designer**

Step 5. **Create Tags**

Step 6. **Launch Clients**

Step 7. **Add History to Tags**

Step 8. **Create a Graph and View Graphs in Clients**

Configure

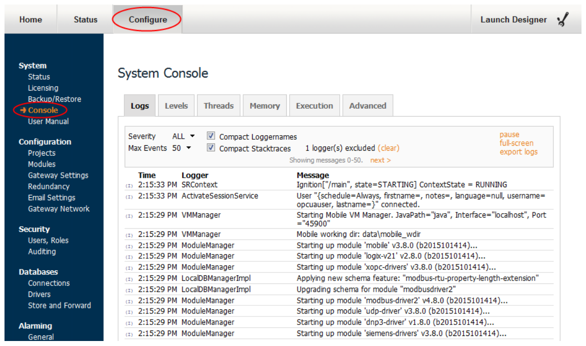
The **Configure** button first takes you to a login page. Use **admin** and **password** as your Username /Password to login.

Once you are logged in, you are in the Configure page. From this page, you do all the Gateway and platform operations. The main menu section on the Configure page changes based on which modules are installed and what your settings are but the defaults are as follows:

- System
- Configuration
- Security
- Databases
- Alarming
- Tags
- OPC-UA
- OPC Connections
- Mobile
- Enterprise Administration

Gateway Console

One of the most important troubleshooting tools of the Gateway Webpage is the **Console** 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 the console in the **Configure** section of the **Gateway Webpage** under **System -> Console**.



Time	Logger	Message
2:15:33 PM	SRContext	Ignition["/main", state=STARTING] ContextState = RUNNING
2:15:33 PM	ActivateSessionService	User "{schedule=Always, firstname=, notes=, language=null, username= opcuauser, lastname=}" connected.
2:15:29 PM	VMManager	Starting Mobile VM Manager. JavaPath="java", Interface="localhost", Port ="45900"
2:15:29 PM	VMManager	Mobile working dir: data\mobile_wdir
2:15:29 PM	ModuleManager	Starting up module 'mobile' v3.8.0 (b2015101414)...
2:15:29 PM	ModuleManager	Starting up module 'logix-v21' v2.8.0 (b2015101414)...
2:15:29 PM	ModuleManager	Starting up module 'xopc-drivers' v3.8.0 (b2015101414)...
2:15:29 PM	LocalDBManagerImpl	Applying new schema feature: "modbus-rtu-property-length-extension"
2:15:29 PM	LocalDBManagerImpl	Upgrading schema for module "modbusdriver2"
2:15:29 PM	ModuleManager	Starting up module 'modbus-driver2' v4.8.0 (b2015101414)...
2:15:29 PM	ModuleManager	Starting up module 'udp-driver' v3.8.0 (b2015101414)...
2:15:29 PM	ModuleManager	Starting up module 'dnp3-driver' v1.8.0 (b2015101414)...
2:15:29 PM	ModuleManager	Starting up module 'siemens-drivers' v3.8.0 (b2015101414)...

Launch Designer

The **Launch Designer** button, on the top right of the page, directly launches the Ignition Designer.

Next ...

- Gateway Control Utility
- Customizing Gateway Homepage

Gateway Control Utility

The Gateway Control Utility, simply called the "GCU", is a lightweight standalone application that provides information about the Gateway. You can use the GCU to monitor the Gateway and perform high-level tasks that aren't available inside of the web application. It provides basic administrative controls, such as stopping and restarting the server and setting the ports used between the Gateway and clients.

Note: The Gateway Control Utility must run on the same machine as the Ignition Gateway and requires administrative privileges.

To launch GCU in Windows

Fom the **Start** menu, go to **All Programs > Inductive Automation > Ignition > Launch Gateway Control Utility**.

You can also launch GCU from the command-line or from the **Start > Run** dialog by typing

```
launch -gcu
```

To launch GCU in Linux

Open a command shell and type

```
gcu
```

If you receive an error saying `"gcu can't be found"`, add the Ignition installation directory to your system path.

If you are running in a headless Linux environment or you have logged into the Linux machine through an SSH shell, the GCU functions are still available in command-line form. See Command-line Utility below.

On this page ...

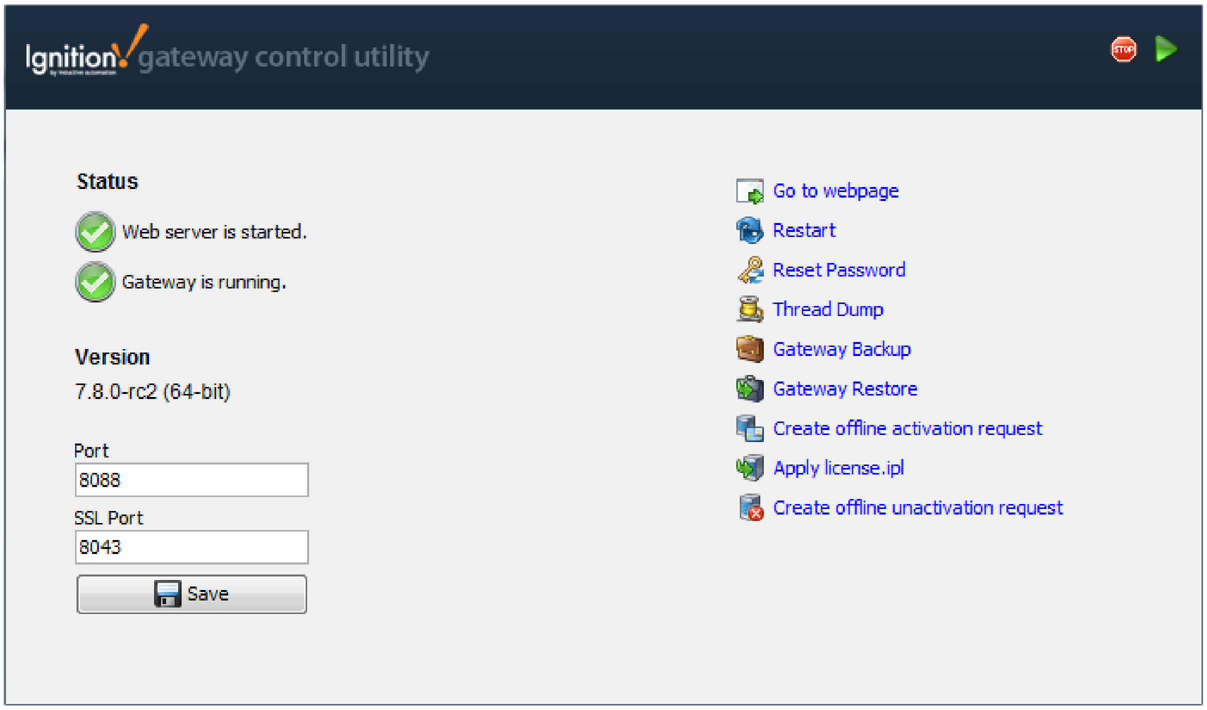
INDUCTIVE UNIVERSITY

Gateway Control Utility

Watch the Video

GCU Window

When GCU is launched, the following window is displayed:


Ignition gateway control utility
STOP

Status

Web server is started.

Gateway is running.

Version

7.8.0-rc2 (64-bit)

Port

8088

SSL Port

8043

Save
Go to webpage
Restart
Reset Password
Thread Dump
Gateway Backup
Gateway Restore
Create offline activation request
Apply license.ipl
Create offline unactivation request

Status

On the upper-left of the window, **Status** shows the state of the Tomcat web server and the Ignition Gateway web application. It is possible for the web server to be running while Gateway has failed. For example, this can occur when the Gateway has faulted on startup. You can also see the Ignition software version that is running.

Go to web page

On the upper-right of the window, **Go to web page** launches a web browser to the Gateway home page.

Restart

On the upper-right of the window, **Restart** restarts the Tomcat web server.

Reset Password

On the upper-right of the window, **Reset Password** lets you reset the root password of the system. Normally, this is not considered a security risk, because the GCU can only be used from the machine the software is installed on, which should be secure. However, it is important to know about this feature here so that the GCU can be removed if the machine can't be properly secured, for example when the server is also used as a client.

Thread Dump

On the upper-right of the window, **Thread Dump** downloads a file with the current states of all threads in the server. Inductive Automation uses this file for troubleshooting problems.

Gateway Backup

On the upper-right of the window, **Gateway Backup** downloads a Gateway backup `.gwbk` file to the local file system. A Save dialog opens to let you specify where you want to save the `.gwbk` file.

Port

On the lower-left of the window, **Port** lets the primary, non-encrypted port, used by clients to communicate with the server. You can see the ports that you are using and you can change the ports and click the **Save** button to save the ports to the Gateway. Note that Gateway must be restarted for the changes to take effect.

Command-line Utility

All the functions in the GCU are also available as a command-line utility for both Windows and Linux.

To run the utility, open a command shell and type:

```
gwcmd <option>
```

Options are as follows:

Options	Description
`a,-–activate <CDk-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 CD-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. If a new file name is not specified at the end of the path, the file name becomes the current date and time along with the `.gwbk` extension. 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.
`-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.
`-p,--passwd`	Resets the Gateway login password.
`-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.

Next ...

- Customizing Gateway Homepage

Customizing Gateway Homepage

You can customize the Gateway homepage to display only the information you want. You can show/hide and reorder the different sections or panels on the homepage.

When you first install Ignition, there are a number of homepage panels that are helpful in beginning a project, but when it
comes time to deploying to production, all the homepage panels may no longer be necessary.

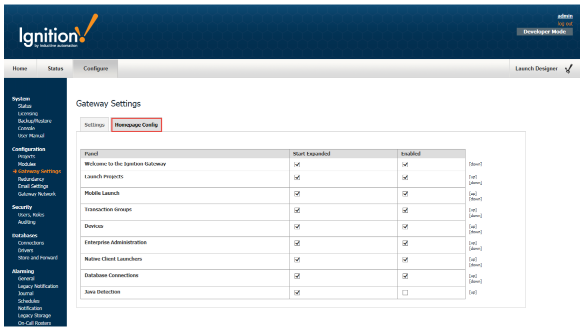
INDUCTIVE UNIVERSITY

Customizing Gateway Homepage

Watch the Video

To change how things are displayed on the Homepage

1. Go to the **Configure** section of the Gateway.
2. Select **Configuration > Gateway Settings**.
3. Select the **Homepage Config** tab.
   You can now see a table that shows all the available panels on the Gateway homepage.
4. Click the boxes you want to display under **Start Expanded** or **Enabled**.
5. Click **up** or **down** at the end of the row showing the panel name to move it up or down.
6. Go back to the **Home** section to see and verify your changes.



Gateway Settings

Panel	Start Expanded	Enabled	
Welcome to the Ignition Gateway	✓	✓	[down]
Launch Projects	✓	✓	[up] [down]
Mobile Launch	✓	✓	[up] [down]
Transaction Groups	✓	✓	[up] [down]
Devices	✓	✓	[up] [down]
Enterprise Administration	✓	✓	[up] [down]
Native Client Launchers	✓	✓	[up] [down]
Database Connections	✓	✓	[up] [down]
Java Detection	✓	☐	[up]

Next ...

- add link here

Advanced Gateway Settings

In addition to customizing the Gateway's homepage, there are a number of advanced features in the Gateway that you can customize, such as changing ports, changing the Java version, and changing the allocated memory size.

Port References

An Ignition server uses many ports to manage information between itself and other Ignition servers, OPC servers, devices, and other services such as SMTP and ERP servers. The following table describes the default ports an Ignition server may use for communication. Most are configurable and some of these ports may be on a computer other than the one Ignition is installed on (that is, if MySQL is installed on a different computer, the Ignition server will not be using port 3306).

Port	Operation	Protocol	Configurable	Description
8088	Listening	tcp	Yes	Default port setting to access the Ignition gateway
8043	Listening	tcp	Yes	Default SSL port setting to access the Ignition gateway
4096	Listening	tcp	Yes	Default port for OPC-UA server
45900	Listening	tcp	No	Callback port for the mobile module
4445	Outgoing	udp/broadcast	Yes	Default send port for a multicast messaged that makes the gateway discoverable on a local network
4446	Listening	udp/broadcast	Yes	Default receive port for a multicast messages that makes the gateway discoverable on a local network
8750	Listening	tcp	No	Port used for Redundancy/Network Configuration
389	Outgoing	tcp	Yes	Default port for Active directory if this is being used
465	Outgoing	tcp	No	SMTP protocol used if Alarming is configured
5060	Listening	tcp/SIP	No	Send Voice Notification to SIP server
8000	Listening	tcp/RTP	No	Transfer/com for SIP server
17341	Outgoing	tcp	No	Send port for SMS with Alarming
17342	Listening	tcp	No	Receive Port for SMS with Alarming
502	Outgoing	tcp	Yes	Default Modbus port
3306	Outgoing	tcp	Yes	Default MySQL port used for connection
1433	Outgoing	tcp	Yes	Default MSSQL port used for connection
1521	Outgoing	tcp	Yes	Default Oracle port used for connection
50000	Outgoing	tcp	Yes	Default IBM DB2 port used for connection
5432	Outgoing	tcp	Yes	Default Postgresql port used for connection
3050	Outgoing	tcp	Yes	Default Firebirdsql port used for connection
6501	Listening	tcp	No	Server fallback port
5500	Listening	tcp	Yes	Default port for OPC browse of external tags

In this section ...

Changing Gateway Ports

The Ignition web server runs on the 8088 port by default, but you can easily change it from the Gateway Control Utility (GCU). You may want to change the default port because the 8088 port is already being used by another application, or maybe you just want to use a more user-friendly port, such as 80.

INDUCTIVE UNIVERSITY

Changing Gateway Ports

Watch the Video

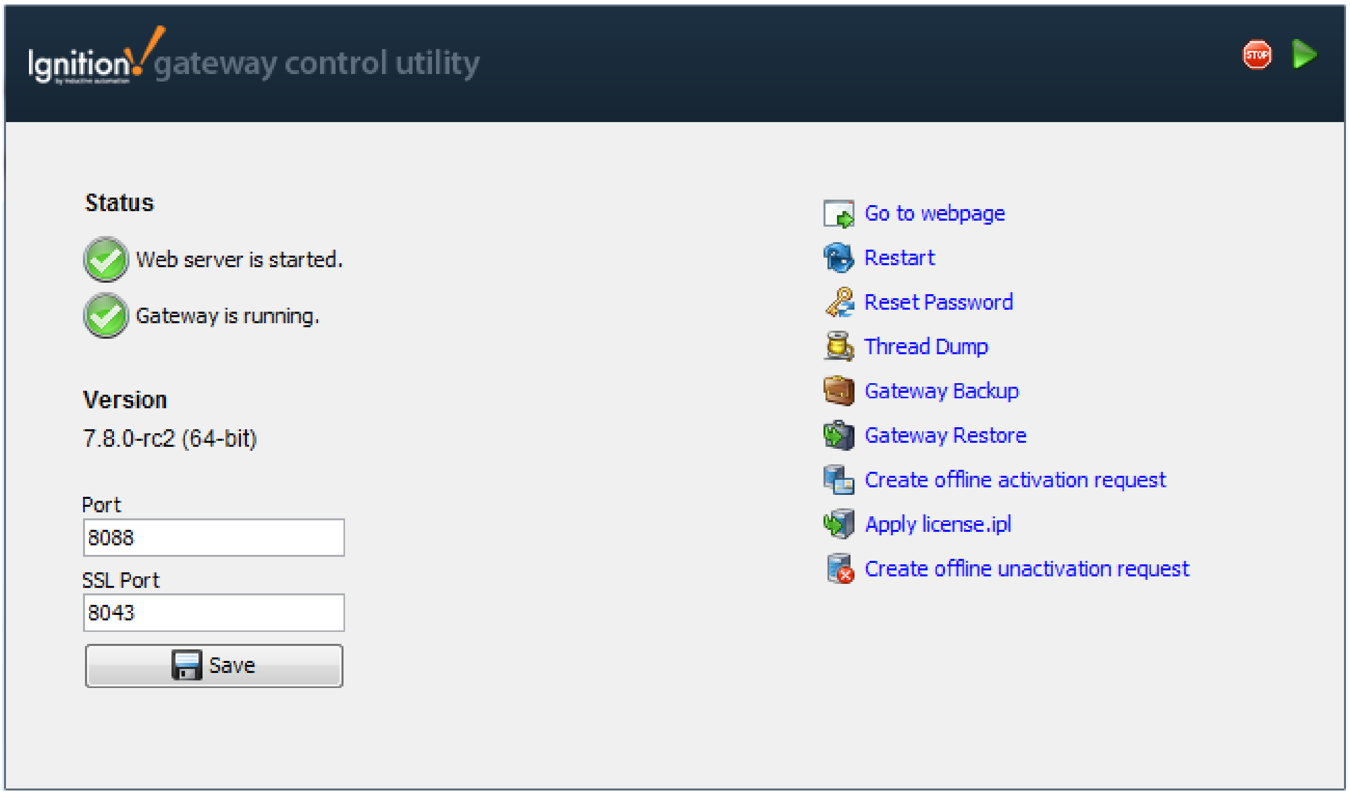
To change the port

Here is how you can change the port:

1. Launch Gateway Control Utility (GCU).
   Note that you can't change the port from the Gateway **Configure** section.
2. In the lower-left side of GCU, type a different number for **Port** and **SSL Port**.
   For example, type **80** for **Port** and **443** for **SSL Port**.
3. Click **Save**.
   Ignition Gateway restarts. If the port is already in use, Ignition will not start. You can go back to GCU and choose another port for the server.
4. Go to any webpage and if you have set Port to 80, type **Localhost** instead of **Localhost:8088** to bring up Gateway.

If using Linux, be aware that ports under 1024 are considered Privileged Ports, and can't be bound to by non-root users. In these cases it's generally recommended to set up iptable rules to redirect packets from the privileged ports to non-privileged ports.

Detailing how to do that is outside the scope of this user manual, but there are plenty of posts and discussions online on the subject.


Ignition gateway control utility

Status

Web server is started.

Gateway is running.

Version

7.8.0-rc2 (64-bit)

Port: 8088

SSL Port: 8043

Next ...

- Changing Gateway Java Version

Changing Gateway Java Version

You can change the Java version that the Gateway Service is using by editing the Ignition configuration `ignition.conf` file. The **Configure>Status** page of the Gateway shows the current Java version you are running.

Here is how you can change the current Java version:

1. Go to **Program Files > Inductive automation > Ignition > data** and find the `ignition.conf` file.
2. Open the `ignition.conf` file and scroll down to the **wrapper Java Properties** section.
3. Change
   `wrapper.java.command=C:/Program Files/`**`Java64`**`/bin/java.exe`
   to
   `wrapper.java.command=C:/Program Files/`**`Java/jre8`**`/bin/java.exe`
4. Save the `ignition.conf` file.
5. Launch the Gateway Control Utility (GCU) and click **Restart** to start the Service again.

After Ignition is restarted, you can verify that the Java version is updated by going back to the **Configure> Status** page to see the Java Version you are now running.

Changing Gateway Java Version

Watch the Video

Next ...

- Changing Gateway Memory Allocation

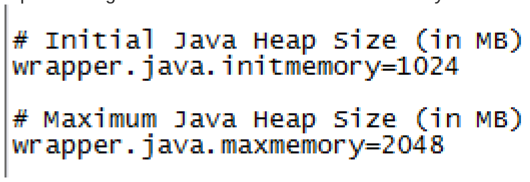
Changing Gateway Memory Allocation

You can change the amount of memory (RAM) dedicated to the Gateway service by editing the Ignition configuration `ignition.conf` file.

The **Configure>Status** page of the Gateway shows the current status of the **Memory (used/max)** – how much memory is being used and the maximum amount of memory allocated.

To change the memory allocation

1. Go to the install directory at **Program Files > Inductive automation > Ignition > data** and find the `ignition.conf` file.

2. Open the `ignition.conf` file and scroll halfway down to **Maximum Java Heap Size (in MB)**.


```
# Initial Java Heap Size (in MB)
wrapper.java.initmemory=1024

# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=2048
```

3. Change
`wrapper.java.maxmemory=`**`2048`**
to
`wrapper.java.maxmemory=`**`4096`**

4. Save the `ignition.conf` file.

5. Launch the Gateway Control Utility (GCU) and click **Restart** to start the service again.
If there is not enough memory in the system, Ignition will not restart and you need to go back to the `ignition.conf` file and change the max memory to the lower number as before.

When Ignition is restarted, you can verify that the memory allocation is updated by going to the **Configure >Status** page and checking the numbers shown for **Memory (used/max)**.

Next ...

- add link here

Changing Gateway Memory Allocation

Watch the Video

Making Backups and Restoring

You can easily backup and restore your Ignition Gateway through a single file. The backup includes all of your projects, tags, and configurations from your Ignition server.

In this section ...

Making Gateway Backups

It is a good idea to create backups anytime you make changes to Ignition. You can create an Ignition backup in two different ways:

- From the Gateway webpage
- From the Gateway Control Utility (GCU)

INDUCTIVE UNIVERSITY

Making Ignition Gateway Backups

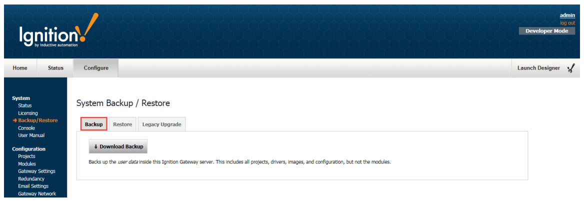
Watch the Video

To create a backup from the Gateway Webpage

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

2. Click on **System > Backup/Restore**.
   The System Backup/Restore page is displayed.

3. Click the **Download Backup** button.
   This downloads a `.gwbk` file to your local file system, which includes everything in Gateway, that is, all projects, security profiles, database connections, and configuration but not the modules and the data that is in the database (you can backup the database separately).
   By default, the `.gwbk` is saved in the **Downloads** folder.

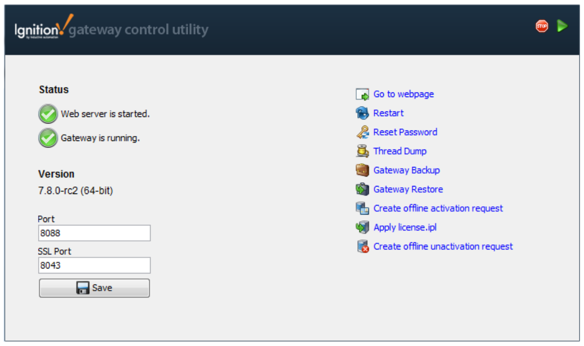


Backs up the *user data* inside this Ignition Gateway server. This includes all projects, drivers, images, and configuration, but not the modules.

To create a backup from the Gateway Control Utility (GCU)

Once you have the GCU window up, click on **Gateway Backup** to create the `.gwbk` file.

Note: You can only access GCU and make the backup file ONLY from the same machine that Gateway is installed on.



Web server is started.
Gateway is running.
7.8.0-rc2 (64-bit)
Port: 8088
SSL Port: 8043

Next ...

- Restoring Gateway Backups

Restoring Gateway Backups

After you created the Ignition backup file, you can restore it using the Gateway Webpage.

Note: As ALL the server's current configuration (except the activation status) will be permanently lost, we recommend you always make a backup of the existing server before restoring.

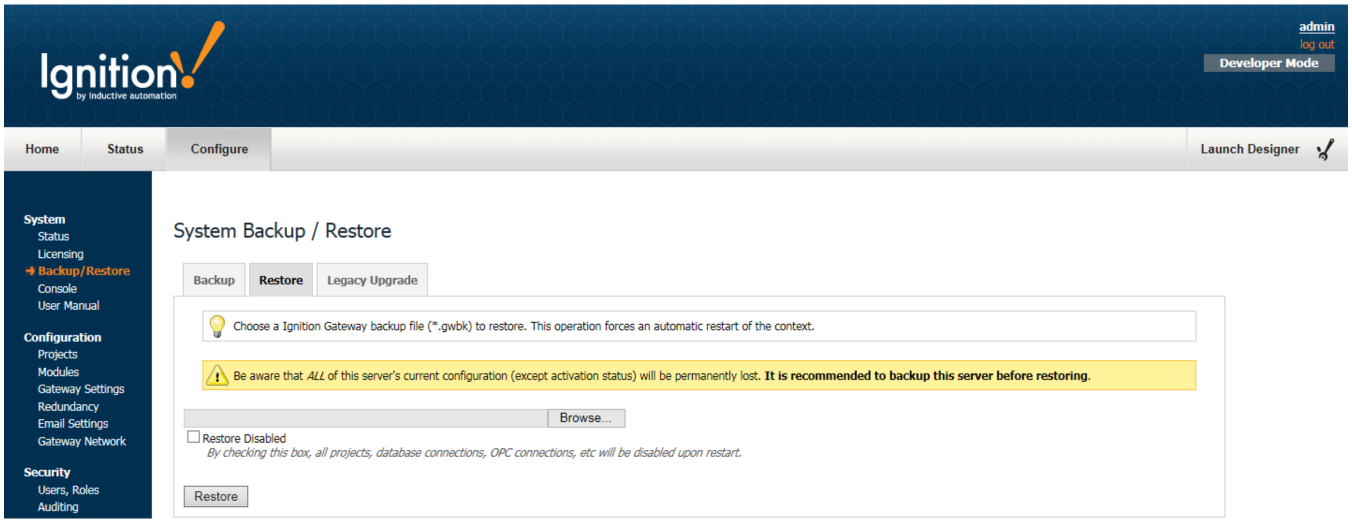
INDUCTIVE UNIVERSITY

Restoring Ignition Gateway Backups

Watch the Video

To restore the backup file

1. Go to the **Configure** section of the Gateway.
2. Click on **System > Backup Restore**.
3. On the **System Backup / Restore** page, click the **Restore** tab at the top of the page.
4. Click on **Choose File**, then go to **Downloads** folder, choose the `.gwbk` file, and click **Open**.
5. Click on **Restore**.
   Gateway stops while restoring the backup file. When restoring is done, restarts itself to apply the restored settings and comes up again.



System Backup / Restore

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.**

☐ Restore Disabled
By checking this box, all projects, database connections, OPC connections, etc will be disabled upon restart.

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.

Next ...

- Setting up Automatic Gateway Backups

Setting up Automatic Gateway Backups

You can set up a daily schedule for automatic Gateway backups in the Gateway Webpage.

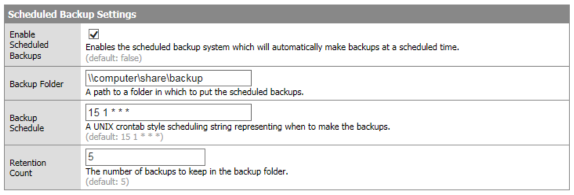
To setup the automatic backup

1. Go to the **Configure** section of Gateway.
2. Click on **Configuration > Gateway Settings**.
3. On the **Gateway Settings** page, scroll down to the bottom of the page to the **Scheduled Backup Settings** section.
   The properties in this section allow you to enable and set the scheduled backups.
4. Select the check box for **Enable Scheduled Backups**.
   This enables the automatic backups at a scheduled time.
5. In **Backup Folder**, type a path to a folder where the backups should 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 (`\\fileserver\backups`).
6. In **Backup Schedule**, type a UNIX crontab scheduling string to specify when you want to make the backups.
   For example, `15 1 ***`, makes a backup `15` minutes after `1` hour every day, every month, and every year.
7. In **Retention Count**, you specify a number which will be the number of backups kept in the backup folder. For example, you can specify 5 if you want to only keep the last 5 backups.
8. Click **Save Changes** at the bottom of the page.
   Now the Gateway will automatically make backups for you.

Setting up Automatic Gateway Backups

Watch the Video



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	5 The number of backups to keep in the backup folder. (default: 5)

Next ...

- Making Project Backups

Making Project Backups

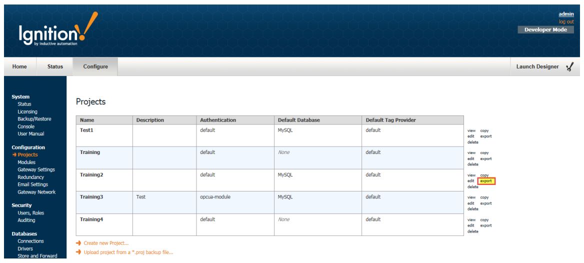
From the Gateway webpage, you can easily make a backup file for a project and upload a project from an existing backup file.

Making Project Backups

Watch the Video

To back up a project and create a backup file

1. Go to the **Configure** section of the Gateway webpage.
2. Select **Configuration > Projects**.
   The **Projects** page is displayed and you can see your existing projects.
3. Click on the **export** link at the right of row for the project you want to backup.
   A `.proj` backup file is created.



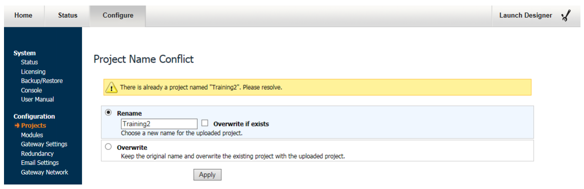
Projects

Name	Description	Authentication	Default Database	Default Tag Provider	
Test1		default	MySQL	default	view copy edit export delete
Training		default	None	default	view copy edit export delete
Training2		default	MySQL	default	view copy edit export delete
Training3	Test	opcua-module	MySQL	default	view copy edit export delete
Training4		default	None	default	view copy edit export delete

Create new Project...
Upload project from a *.proj backup file...

To upload a project backup file

1. Do steps 1 and 2 above.
2. Find the orange arrows and click on **Upload project from a *. proj backup file...**.
   The **Upload Project** page is displayed.
3. Click on **Choose File** and select your `.proj` file and click **Open**.
4. On **Upload Project** page, click on **Upload**.
   If there is already a project with the same name, the **Project Name Conflict** page is displayed giving you the option to **Rename** the project or **Overwrite** it.
5. **Rename** the project and click **Apply**.
   The Projects page is displayed again and you will see your new project is added to the list along the other the projects.



Project Name Conflict

There is already a project named "Training2". Please resolve.

[✓ Rename]
Training2 ☐ Overwrite if exists
Choose a new name for the uploaded project.

Overwrite
Keep the original name and overwrite the existing project with the uploaded project.

Next ...

- Project Backups vs. Ignition Gateway Backup

Project Backups vs. Ignition Gateway Backup

There are two main backup types that can be made in Ignition: a **Gateway** backup and **Project** backup.

Making Project Backups

Watch the Video

Gateway Backup

The **Gateway** backup (*.gwbk file) includes everything you find in the Ignition Gateway webpage. This means all of the projects, global resources, Ignition tags, database connections, OPC and Device connections, plus all the settings of the gateway. The only things that aren't included in this backup is information stored in other programs, IE: the data in a database, PLC programs, etc.

Note that restoring a Gateway backup is the easiest way to restore your system, but it will overwrite all of the existing settings including your projects. There is no merge option for a Gateway backup.

Project Backup

A **Project** backup (*.proj file) is much smaller than a Gateway backup and only includes data from the selected project. This means all its resources such as windows, transaction groups, templates, reports, project scripts, etc. Notably missing from the project backup are any Gateway scoped resources, even if they are presented in the Designer. IE: Ignition tags, stored raster images, global items like SFCs, alarm pipelines, global script functions etc. When a project backup is made, it is recommended to also backup your tags, images, and anything else you might be using. While the project may rely heavily on a specific database connection or tag provider, those are not included in the project backup.

When you upload/restore a project backup 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 backups can also be created/restored in the Designer. This will even allow you to select which parts of the project backup you want to include and will merge them into the currently open project.

Next ...

- About Licensing
- Making Backups and Restoring
- Using Designer User Interface

Licensing and Activation

How Licensing Works

Ignition is a modular platform, therefore it is licensed based on modules. Licensed and unlicensed modules can operate side-by-side, so some modules can be tested in trial mode while others are running in a licensed state.

On this page ...

How Activation Works

Activation refers to locking down a CD-Key to the install machine and notifying the modules of their license state.

CD-Key

When the software and the modules are purchased, you are provided with a six-digit CD-Key. If you add a module, your account is updated, and you can re-use your existing CD-Key to activate the new features. You can also unactivate your CD-Key, and reuse it to activate Ignition on a different machine.

Two methods of Activation

You can do the activation in two ways:

- Over Internet
- Manually – by email or through the Inductive Automation website

CD-Key and License File

Despite the modular licensing, there is only one CD-Key and license file for each server. That is, there is a single license file that specifies which modules are currently activated.

When module(s) are purchased, you receive a CD-Key, a six-digit code, that identifies your purchase. You use this CD-Key to activate the software through the Ignition Gateway. If you are adding any additional modules, your account is updated and you can re-use your existing CD-Key to activate the new features.

You can unactivate your CD-Key, thus freeing it for activation on a different machine.

Licensing for Production Use

One thing to consider when deploying an Ignition installation to production use is the way Ignition will be licensed.

USB License Key

If you anticipate that the installation might move from server to server frequently, you may want to consider purchasing a USB license key to ease transition to new servers. This also makes things more convenient when the server is being deployed in an area without an active internet connection.

File-based Licensing

When the installation is on one server and not moving, you can use a file-based licensing scheme, and with an internet connection you can activate the installation online. If you don't have an internet connection, you can download an activation request file and put it on a portable memory device and take it to a workstation with an active internet connection.

Then you upload the file to the Inductive Automation website and you will receive a license file, called `license.ipl`, in return. Take this file back to the Gateway you are trying to activate, in the **Config** section under **System > Licensing** you can upload and activate the license.

Information Collected at Activation

When activating an Ignition installation, there are several pieces of information that are communicated to the activation servers and then stored internally at Inductive Automation. Some of this information is used to generate a unique license for the server on which Ignition is being installed.

The following are the type of information that is collected at the time of activation:

- CD Key
- System ID - a hash representing the hard drive serial number (windows) or inode number of the /bin folder (linux)
- Java Runtime Version

- Java Vendor
- Java Version
- Java Virtual Machine Name
- Java Virtual Machine Vendor
- OS Architecture
- OS Name
- OS Version
- Processor Count
- Total Physical Memory

How Un-activation Works

For a given CD-Key, only one Ignition Gateway instance is allowed to be activated at a given time. If you want to activate Ignition on a different server, you must first unactivate the previous server.

To unactivate the Gateway, do the following steps:

1. From the **Configure** section of Gateway, go to **System > Licensing**.
   The **Licensing and Activation** page is displayed and you can see the currently installed License Key.

2. Scroll-down, find the orange arrows, and click on **Unactivate License**.
   The **Really Unactivate?** page is displayed.

3. Click **Confirm** to remove the activation from this copy of Ignition.
   If the Ignition server has internet access, the unactivation occurs immediately, and the license is again available for activation on another Gateway server.
   If the Ignition server has no internet access, an unactivation request file is generated, and the license is not allowed to be activated until the file is loaded into the Inductive Automation website, or emailed to Support. See Transferring a License for step-by-step instruction to activate the license on a new Gateway server.

In this section ...

About Licensing

Ignition is a modular platform, therefore it is licensed based on modules. Licensed and unlicensed modules can operate side-by-side, so some modules can be tested in trial mode while others are running in a licensed state.

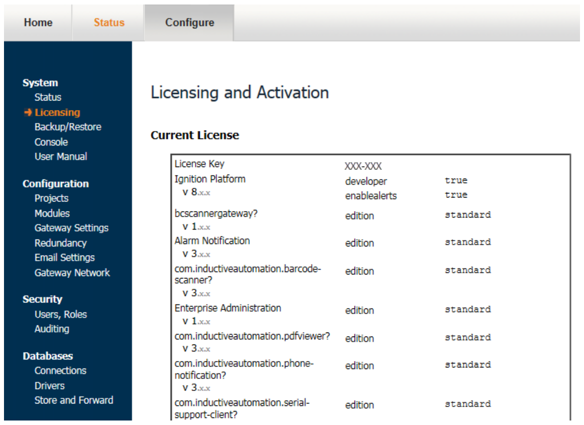
INDUCTIVE UNIVERSITY

About Licensing

Watch the Video

To see detailed information about the modules licensing status along with the version number for each module

1. Go to the **Configure** section of the Gateway webpage.
2. Select **System > Licensing**.
   The **Licensing and Activation** page is displayed. Here you can see the CD-Key for the Ignition license plus all the modules that Ignition is licensed for. The version number for each module is also shown.

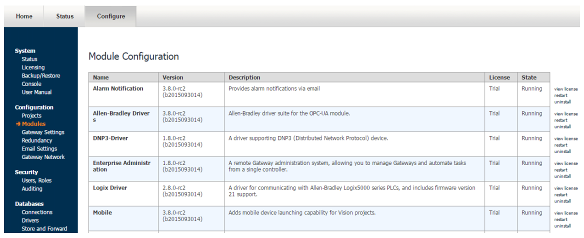


Licensing and Activation

Current License

License Key	XXX-XXX	
Ignition Platform v 8.x.x	developer	true
	enablealerts	true
bcscannergateway? v 1.x.x	edition	standard
Alarm Notification v 3.x.x	edition	standard
com.inductiveautomation.barcode-scanner? v 3.x.x	edition	standard
Enterprise Administration v 1.x.x	edition	standard
com.inductiveautomation.pdfviewer? v 3.x.x	edition	standard
com.inductiveautomation.phone-notification? v 3.x.x	edition	standard
com.inductiveautomation.serial-support-client?	edition	standard

3. Go to **Configuration > Modules**.
   The **Module Configuration** page is displayed. Here you can see all the modules that are currently running with their version numbers and license status. The licenses are either in **Trial, Activated,** or **Free** mode. On this page, you can add or remove modules from your Ignition. If you are licensed for a module you are running, it will run in **Activated** mode. If you are not licensed for a module you are running, it will run in **Trial** mode until the Trial time is expired.



Module Configuration

Name	Version	Description	License	State	
Alarm Notification	3.8.0-rc2 (b2015093014)	Provides alarm notifications via email	Trial	Running	view license restart uninstall
Allen-Bradley Drivers	3.8.0-rc2 (b2015093014)	Allen-Bradley driver suite for the OPC-UA module.	Trial	Running	view license restart uninstall
DNP3-Driver	1.8.0-rc2 (b2015093014)	A driver supporting DNP3 (Distributed Network Protocol) device.	Trial	Running	view license restart uninstall
Enterprise Administration	1.8.0-rc2 (b2015093014)	A remote Gateway administration system, allowing you to manage Gateways and automate tasks from a single controller.	Trial	Running	view license restart uninstall
Logix Driver	2.8.0-rc2 (b2015093014)	A driver for communicating with Allen-Bradley Logix5000 series PLCs, and includes firmware version 21 support.	Trial	Running	view license restart uninstall
Mobile	3.8.0-rc2 (b2015093014)	Adds mobile device launching capability for Vision projects.	Trial	Running	view license restart uninstall

4. Compare the module version numbers from Step 2 and Step 3.
   If you find from Step 2 that your license has an older version of a module than from Step 3, the one you are running in Trial mode, you can continue to use the Trial mode as long as the Trial time is not expired. When a module is in Trial mode, Ignition shows the Trial time remaining in the **Configure** section of the Gateway webpage under **System > Status**.

Next ...

- About the Trial Period

About the Trial Period

Inductive Automation offers you an easy way to access and learn Ignition. You download Ignition on the web and start using it for 2 hours at a time, with no other restrictions.

When unlicensed, the Ignition Gateway will run for two hours at a time. At the end of the 2-hour period, all unlicensed modules are stopped and the system will stop most of its functions. All the different parts of the Gateway, including all the modules, share the same clock and will timeout simultaneously. For example, all transaction groups will stop logging and clients will show a demo screen.

You can re-start the trial period by logging into the Gateway and enabling another 2 hours of execution. The trial period may be restarted any number of times.

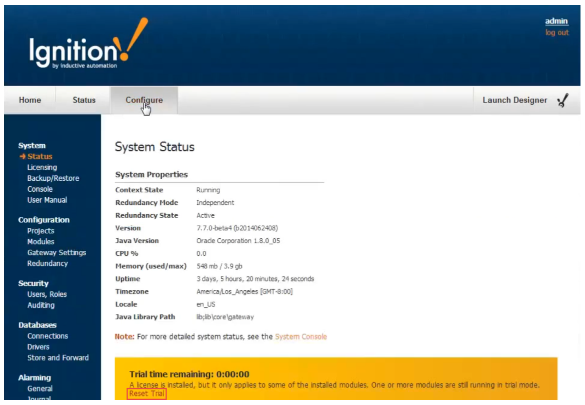
You can check the amount of time remaining on the Trial period from the **System Status** page of the **Con figure** section on the Gateway webpage. If you find the Trial period has ended, click on **Reset Trial** at the very bottom of the page to start a new 2-hour time.

Depending on the module, you may need to take some additional actions. For example, the Vision Clients requires you to log out and back in again in order to continue the trial.

INDUCTIVE UNIVERSITY

About the Trial Period

Watch the Video



System Status

System Properties

Context State	Running
Redundancy Mode	Independent
Redundancy State	Active
Version	7.7.0-beta4 (b2014062408)
Java Version	Oracle Corporation 1.8.0_05
CPU %	0.0
Memory (used/max)	548 mb / 3.9 gb
Uptime	3 days, 5 hours, 20 minutes, 24 seconds
Timezone	America/Los_Angeles [GMT-8:00]
Locale	en_US
Java Library Path	lib;lib\core\gateway

Note: For more detailed system status, see the System Console

Trial time remaining: 0:00:00

A license is installed, but it only applies to some of the installed modules. One or more modules are still running in trial mode.

Reset Trial

If you have a license and want to try out other modules, just add them to your gateway. **Only the modules that are unlicensed will time out** when the Trial period ends. All other modules will continue working normally.

If you see the Trial timer when you think you shouldn't, go check the Modules page in the Configure section. Sometimes you have a module installed that you didn't know about, or you have a module that is a newer version than your license.

Next ...

- Activating Over Internet

Activating Over Internet

Before you do the following steps...

Make sure you have received a CD-Key from Inductive Automation.

To activate your license over the internet

1. Go to the Gateway webpage and login to the **Configure** section.
2. From the menu on the left, select **System > Licensing**.
3. Look for the orange arrow and click on the **Purchase or activate this Ignition Gateway** link. The **Purchase or Activate?** page is displayed.
4. Click on **Activate**.
5. Enter your **CD-Key** and click **Activate**.

   The request to Activate is processed over the internet.When you don't have an Internet connection, the **Offline Activation** page is displayed describing how you can activate manually.

   The **Licensing and Activation** page is displayed. You can see the **Current License** along with all the modules that come with your CD-Key.
6. From the menu on the left, select **Configuration > Modules**.
   The Module Configuration table is displayed. You can see that the license for all the modules are now activated.

Activating Over Internet

Watch the Video

Information Collected at Activation

When activating an Ignition installation, there are several pieces of information that are communicated to the activation servers and then stored internally at Inductive Automation. Some of this information is used to generate a unique license for the server on which Ignition is being installed.

The following are the type of information that is collected at the time of activation:

- CD Key
- System ID - a hash representing the hard drive serial number (windows) or inode number of the /
- bin folder (linux)
- Java Runtime Version
- Java Vendor
- Java Version
- Java Virtual Machine Name
- Java Virtual Machine Vendor
- OS Architecture
- OS Name
- OS Version
- Processor Count
- Total Physical Memory

Next ...

- Launching Gateway

Activating Manually

Before you do the following steps...

Make sure you have received a **CD-Key** from Inductive Automation.

To activate your license manually, that is offline, not over internet

When you do not have internet connection, you can do the following steps to activate your license manually:

1. Go to the Gateway webpage and login to the **Configure** section.
2. From the menu on the left, select **System > Licensing**.
3. Look for the orange arrow and click on the **Purchase or activate this Ignition Gateway** link. The **Purchase or Activate?** page is displayed.
4. Click on **Activate**.
5. Enter your **CD-Key** and click **Activate**.

   When you don't have an Internet connection, the **Offline Activation** page is displayed describing how you can activate manually.
   In the offline activation process, an activation request file, called `activation_request.txt`, is generated and downloaded.
6. Take the `activation_request.txt` file to a machine with Internet access and either go to the **Support** area on our website to activate there or email it to support@inductiveautomation.com. Either way when activation is done, a license file called `license.ipl` is generated.
7. Take the `license.ipl` file back to the Gateway machine to enter it into the **Licensing and Activation** page and click the **Activate** button.

INDUCTIVE UNIVERSITY

Activating Manually

Watch the Video

Next ...

- Launching Gateway

Reloading a License

Your license needs to be reloaded once it has been upgraded by Inductive Automation to include new modules, removed modules, or a different license level. The same CD-Key is used for the license upgrade. You can update your license using the same CD-Key by reloading the license from the same location in the Gateway as activation.

Reloading a License

Watch the Video

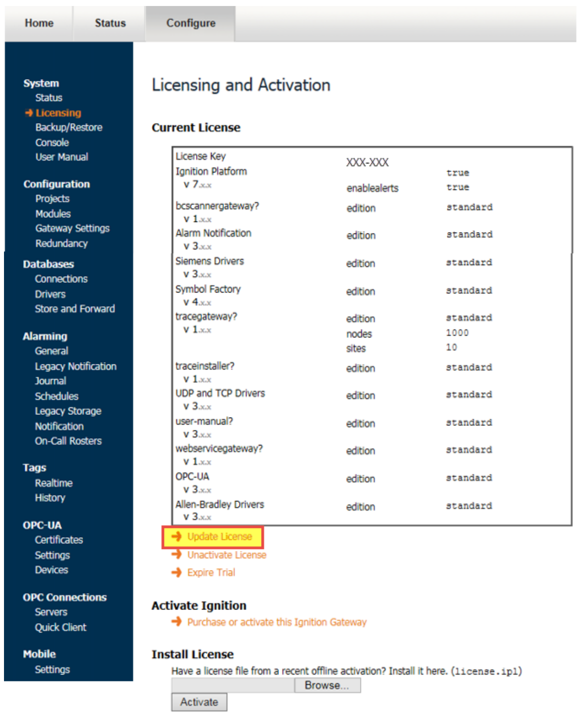
With Internet Connection

With internet connection, you do the following steps to update and reload your license:

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

2. Click on **System > Licensing**.
   The **Current License** CD-Key and activated modules are displays.

3. Find the orange arrows at the bottom of the page and click on **Update License**.
   This connects your machine to the Ignition licensing server. The licensing server verifies your license CD-Key to be valid then updates your license. You will see new modules, if you have purchased any, and new versions for the modules you already have.



Licensing and Activation

Current License

License Key	XXX-XXX	
Ignition Platform		true
v 7.x.x	enablealerts	true
bcscannergateway? v 1.x.x	edition	standard
Alarm Notification v 3.x.x	edition	standard
Siemens Drivers v 3.x.x	edition	standard
Symbol Factory v 4.x.x	edition	standard
tracegateway? v 1.x.x	edition	standard
	nodes	1000
	sites	10
traceinstaller? v 1.x.x	edition	standard
UDP and TCP Drivers v 3.x.x	edition	standard
user-manual? v 3.x.x	edition	standard
webservicegateway? v 1.x.x	edition	standard
OPC-UA v 3.x.x	edition	standard
Allen-Bradley Drivers v 3.x.x	edition	standard

Update License
Unactivate License
Expire Trial

Activate Ignition

Purchase or activate this Ignition Gateway

Install License

Have a license file from a recent offline activation? Install it here. (license.ipl)

Browse...
Activate

Without Internet Connection

If you do not have internet connection and want to reload and update your license, you must unactivate and manually activate Ignition. Do the following steps:

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

2. Click on **System > Licensing**.
   The **Current License** CD-Key and activated modules are displays.

3. Unactivate Ignition. Find the orange arrows at the bottom of the page and click on **Unactivate**.
   Ignition will attempt to unactivate automatically but if it cannot, a new page will open with instructions for manual Unactivation. Follow the steps there. This will include downloading an unactivation file and taking it to another computer that has an internet connection. Have a thumb drive handy to transfer files if you can't send them directly to another computer.

4. Activate Ignition. With any new activation, all updates to the license will be automatically applied. Follow the previous steps but this time click the **Purchase or activate this Ignition Gateway** link. For more help activating Ignition manually, see the Activating Manually section.

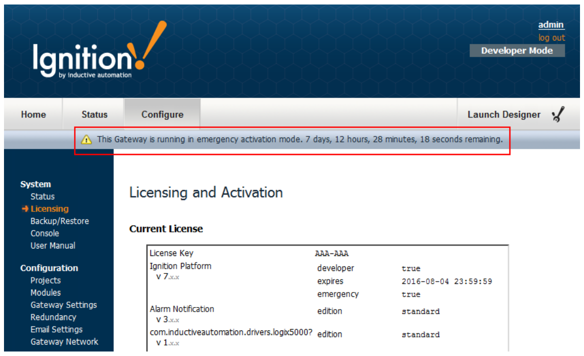
Next ...

- add link here

Emergency Activation

What is Emergency Activation?

After activating your License Key in Ignition, you may see a banner that says Ignition is in **Emergency Activation Mode**.



This Gateway is running in emergency activation mode. 7 days, 12 hours, 28 minutes, 18 seconds remaining.

Licensing and Activation

Current License

License Key	AAA-AAA	
Ignition Platform v 7.x.x	developer	true
	expires	2016-08-04 23:59:59
	emergency	true
Alarm Notification v 3.x.x	edition	standard
com.inductiveautomation.drivers.logix5000? v 1.x.x	edition	standard

Emergency Activation

Watch the Video

What does this mean?

This means the License Key that you used has already been activated. Either it was used previously on another computer, or something went wrong during activation and it registered twice. Fortunately, you have plenty of time to fix this problem. Emergency Activation More means **you are fully licensed for 7 days**, so even at night or over the weekend, you can still run without interruption until it is fixed.

How do I fix it?

In order to fix your License Key, you must **contact Inductive Automation**. You can reach us by email or by phone and we will walk you through fixing your license. It is usually a very quick procedure that involves you re-activating after we have fixed your License Key from our end.

Why does this exist?

Basically: computers sometimes fail and you need to get running again, fast.

When you initially activate your copy of Ignition, it will run as long as your computer does. As we all know, sometimes there are hardware failures or your computer just stops working. There are many things you can do to minimize this, but once it happens you need to be able to get Ignition up and running again quickly. This Emergency Activation Mode allows you to do this without having to involve Inductive Automation until after your facility is back on track.

Quick Disaster Recovery Plan

What you need before an emergency happens

If you already have Ignition installed and running in your facility, it's easy to get back to running quickly. Just make sure you have the following stored on a computer or shared drive **that is not your Ignition Gateway**:

- A Gateway Backup. If you don't already, you should set up Automatic Backups.
- Your Ignition License Key. Make sure you store the 6 digit code somewhere that you can easily retrieve it if your computer fails.
- The Ignition software Installer. In an emergency, it's best to have the installer for the version you are currently using. You can always download an archived version from our website, but having one on hand is preferred. *Pro tip*: Edit the installer filename to include your License Key so it's easily available.

Emergency Restore Steps

1. Find another computer or create a new Virtual Machine. You need to get something up and running to put Ignition on first. We recommend having something on standby that is the same as the original computer. Further considerations:
    a. Set your new computer to the same IP Address of the original Ignition Gateway. This way your clients won't need to do anything special to start running again.

    b. Make sure your firewall is set correctly. At minimum, you must have the Ignition port open (8088 by default). In an emergency situation, you may want to just disable the firewall temporarily.

2. Install Ignition on the new computer. If you don't have Java installed already, Ignition will install it for you.

3. Load a Gateway Backup. Everything in Ignition is in the Gateway Backup, once it's loaded, you are running again!

4. Activate your License. Your existing License Key will only work normally the first time it is used. After that, if you try to use it again, it will instead go into an **Emergency 7-Day Trial**. This gives you plenty of time to get back up and running before you have to deal with the License Key, even if your failure is on the weekend or outside Inductive Automation's business hours.

5. Contact Inductive Automation. You can reach us by email or by phone (**1-800-266-7798**) and we will walk you through fixing your license.

Transferring a License

You can transfer your license from one machine to another by simply unactivating the currently licensed machine and activating the other one.

The unactivation process is similar to the licensing activation procedure, but in reverse. If the licensed Ignition server has internet access, the unactivation occurs immediately, and the license is again available for activation.

If you do not have internet access, an unactivation request file is generated, and the license is not allowed to be activated until the file is loaded into the Inductive Automation website, or emailed to support.

Transferring a License

Watch the Video

Transferring Licenses with Internet Access

Lets say you have two different copies of Ignition, an **old Ignition** at:

`10.20.7.132:8088/main/web`

and another **localhost Ignition** at:

`localhost:8088/main/web`

On the old Ignition

1. Go to the **old Ignition**, as in our example, go to

   `10.20.7.132:8088/main/web`
2. Go to the **Configure** section of the Gateway webpage.
3. Select **System > Licensing**.
4. On the **Licensing and Activation** page, look for your current license CD-Key, and write down the **CD-Key.**
   You will need to enter this CD-Key to activate the new Ignition.
5. Scroll-down, find the orange arrows, and click on **Unactivate License**.
   The **Really Unactivate?** page is displayed.
6. Click **Confirm** to remove the activation from this copy of Ignition.

On the other copy of Ignition

1. Go to the other copy of Ignition, as in our example at localhost:

   `localhost:8088:main/web`
2. Select **System > Licensing**.
   On the **Licensing and Activation** page you can see that under **Current License** there is **No license installed**.
3. Find the orange arrow, and click on **Purchase or activate this Ignition Gateway**.
   Purchase or Activate? page is displayed.
4. Click on **Activate**.
5. Enter your **CD-Key** that you wrote down previously and click **Activate**.

You have now transferred the CD-Key from one copy of the Ignition to another copy of it.

Next ...

- add link here

Ignition Redundancy

Ignition uses redundancy to maximize uptime execution by using a two-node system, a master node and a backup node. In this section you will learn how to set up redundancy and what to consider about databases when using this model.

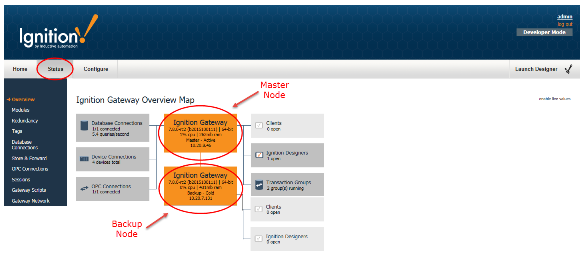
In this section ...

How Redundancy Works

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:

On this page ...



How Redundancy Works

Watch the Video

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. By default, all communication goes from the backup to the master node on **port 8750**. Therefore, that port must allow TCP listening on the master machine. You can change the port from the Gateway webpage in the **Redundancy** settings page.

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, 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 does not subscribe to tag values, and does not communicate with any device. This allows the system to standby without putting additional load on the devices and network. Fail-over 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 Fail-over

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.

Next ...

- Setting Up Redundancy

Setting Up Redundancy

Before You Set Up Redundancy

In redundancy, both nodes will share the exact same configuration state. When a backup node joins a master node, it downloads a backup of the master's state and restores that backup on itself. Therefore, before you set up for redundancy, you need to consider and take care of the following things:

1. **Start with a fresh install for the backup node.**
   Because the current configuration of the backup node is overwritten, make sure that it does not contain anything valuable in it when enabling redundancy.

2. **All system configuration 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 a variety of ports. Turning off software firewalls or adding special exception rules for each others' addresses is required. Specifically, the master node must be able to receive data on TCP/IP **port 8750** (you can change this in the Settings), and the backup node must be able to send outgoing data on that port.

The OS platform (i.e. Windows, OS X, Linux) for the Master and Backup must be the same for Redundancy. This means that if your Master Ignition is installed on Windows 8, your backup can be on a Windows 10 machine, but not on OS X.

On this page ...

INDUCTIVE UNIVERSITY

Setting Up Redundancy

Watch the Video

To Set Up Redundancy

Once the above two points are done, setting up redundancy is fairly simple. You do the following steps to set up your redundant pair:

On the Master Gateway

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

2. Select **Configuration > Redundancy**.
   The **Redundancy and Network Configuration** page is displayed showing different sections and settings.
   You can use most of the default settings on this form.

3. Change the following Settings:
   - **Redundancy Settings**, set **Mode** to **Master**
   - **Backup Mode Settings**, type in the **Master Node Address** (for example, 10.20.8.12)

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

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

6. Select **System > Status** to ensure the redundancy mode and state is properly set.

On the Backup Gateway

1. Do the exact same steps 1-6 above on the **Backup Gateway** webpage, except in step 3 set the **Redundancy Settings Mode** to **Backup**.

2. Verify the correct port setting in **Network Settings**.

3. 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 see the **System Map**. The System Map should show the connected nodes and their current states.

   After saving, the system connects to the master and downloads a system backup and 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 Ignition configuration Gateway under **Configuration>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.

Redundancy Settings	
Mode	**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 operates when it is not the **Active** node. **Cold** - Performs minimal activities, does not connect to devices, and so on. The purpose is to minimize the load on the network and on devices. **Warm** - Connect to devices, subscribe to tags and set up all executing objects. The purpose is to minimize fail-over time.
Fail-over Timeout	The time, in milliseconds, before the opposite node is determined to be unavailable and this node takes over.
Startup Connection Allowance	The time, in milliseconds, to wait on initial startup for a connection to be established before making a decision on the node's activity 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	
Port	For the master, the port to listen on. For the backup, the port to connect to on the master.
Auto-detect Network Interface	If true, the system will automatically select which network interface to use on the machine. If false, the system will bind itself to the interface of the specified address.
Network Bind Interface	The address to bind to if Auto-detect is false.
Auto-detect HTTP Address	When clients are launched, they are provided with a list of addresses that they may connect to. If this option is true, the list will be generated automatically. If false, they will be provided with the list specified.
HTTP Addresses	The list of addresses to give to the clients if auto-detect is turned off. These are the addresses that the clients will attempt to connect to, so the HTTP and HTTPS ports must match the configuration of the system in the Gateway Control Utility

Master Node Settings	
Recovery Mode	How the master acts when it connects to a backup node that is currently active. **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	How many "runtime status updates" can be queued up in memory before the system stops tracking them and forces a full refresh. 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.

Buffer Size	

Backup Node Settings	
Master Node Address	The address where the master is located.
Ping Rate	The time, in milliseconds, between messages from the backup to the master.
Reconnect Period	How often in millisecond, to re-attempt connection when the backup node is not connected 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.

Next ...

- Database Considerations

Database Considerations

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.

On this page ...

INDUCTIVE UNIVERSITY

Database Considerations

Watch the Video

Ignition Database Requirements

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

If using the default internal provider, **Tags** only rely on the database for tags which 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. If using an external provider, which is housed in the database, the tags will no longer function. Therefore it's much more important to have a read-capable database connection available when using external tags.

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 log 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.

Database Architectures

Single Shared Server

A single database server is used. The Ignition gateways will both 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.

Next ...

Redundant Licensing

In redundant systems, both Ignition Gateways need their own licenses. It is possible to use two standard Ignition licenses in a redundant system: one license will be applied to the master gateway, and the other license will be applied to the backup gateway.

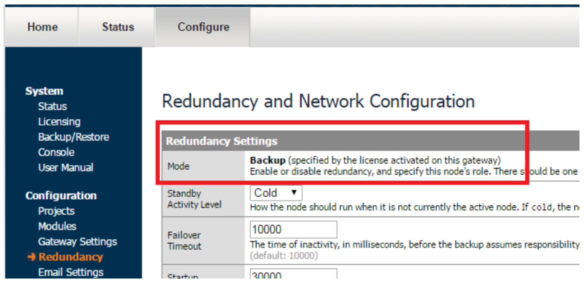
Redundant Licensing

Watch the Video

Backup-Only Licenses

Backup-Only licenses are available for purchase. This license force 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.

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



Redundancy and Network Configuration

Redundancy Settings	
Mode	**Backup** (specified by the license activated on this gateway) Enable or disable redundancy, and specify this node's role. There should be one
Standby Activity Level	Cold ▼ How the node should run when it is not currently the active node. If `cold`, the
Failover Timeout	10000 The time of inactivity, in milliseconds, before the backup assumes responsibility (default: 10000)
Startup	30000

Instead of the dropdown list that normally appears on this property, the license has 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 operator under a 2 hour trial.

Next ...

Designer

The Ignition Designer is a web-launched application that lets you configure and build your projects to be viewed by the network Clients. The application is launched from the Gateway homepage by clicking on the **Launch Designer** button on the top-right of the page.

In this section ...

Designer Basics

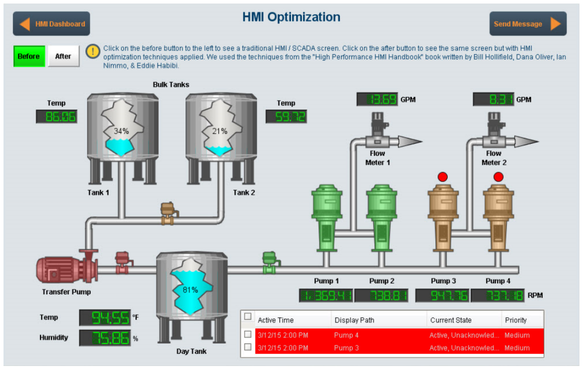
The Ignition Designer is where the majority of configuration and design work is done. You web-launch it from the Gateway and use it to build projects, which are the major unit of design. The projects are then viewed by the Clients.

On this page ...

What is a Project?

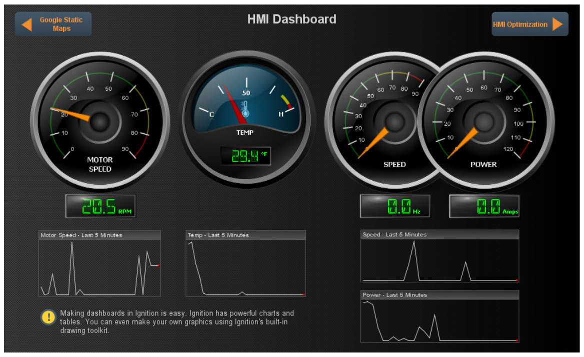
In Ignition, a project is a unit of configuration that contains: Windows, Components, Transaction Groups, General Settings, and Security Settings. You use the Designer to configure and create projects. The projects are then viewed in the runtime Clients.

Here are two examples of projects as created in the Designer and viewed in the Client:



Click on the before button to the left to see a traditional HMI / SCADA screen. Click on the after button to see the same screen but with HMI optimization techniques applied. We used the techniques from the "High Performance HMI Handbook" book written by Bill Hollifield, Dana Oliver, Ian Nimmo, & Eddie Habibi.

	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



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.

When you launch the Designer, you're asked 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. Projects that contains viewable elements, such as windows or reports, will have a launch link on the Gateway homepage. Otherwise, the project runs in the Gateway and will not have a Client runtime.

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

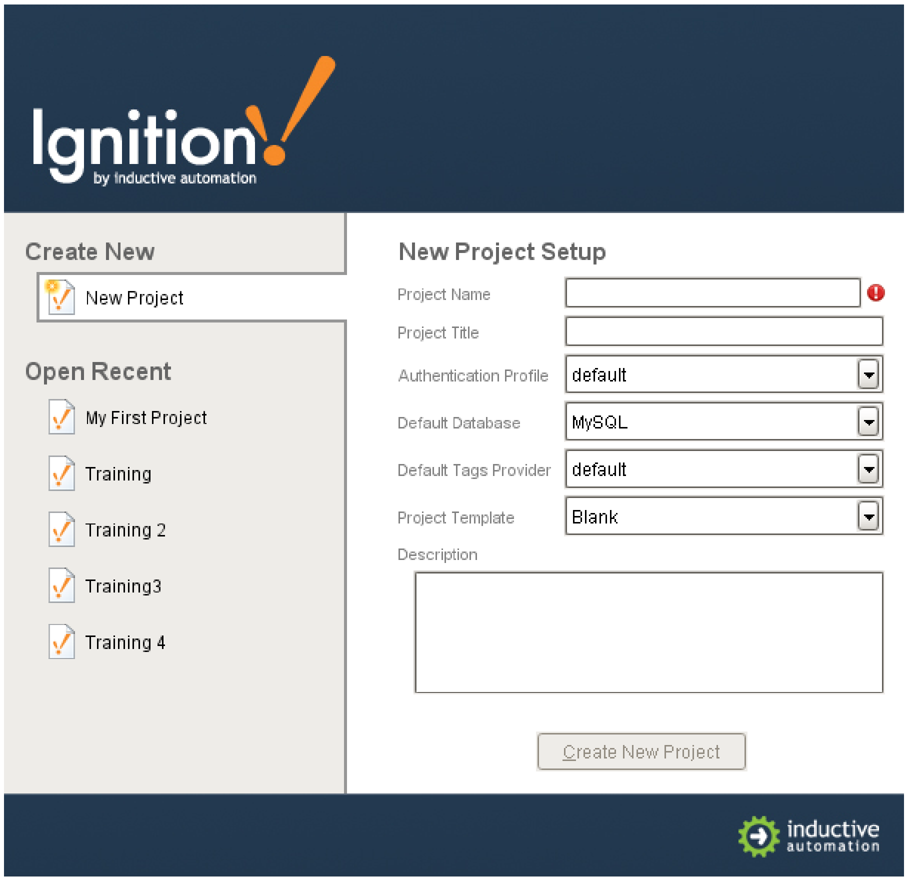
After you make all your changes and save the project, all the project settings and resources are sent to the Gateway, where they are handled by the appropriate modules.

What is not a Project?

There are a number of settings that are shared by ALL projects on the Gateway system, such as, tag providers, database connections, OPC server connections, OPC-UA module device configurations, and alarm notification pipelines. A project does not contain these settings.

Creating a Project

While launching the Designer, the **Open/Create Project** window is displayed. You will have the option to create a new project or open an existing project.


Create New

New Project

Open Recent

My First Project

Training

Training 2

Training3

Training 4

New Project Setup

Project Name	
Project Title	
Authentication Profile	default
Default Database	MySQL
Default Tags Provider	default
Project Template	Blank
Description	

To create a new project, click on **New Project** and define following information:

Project Name
A unique name for the project in the system. The name can only consist of alphanumeric characters and the '_' character. Spaces and other special characters are not supported.
Note: it is not advisable to change the name after it's been created, instead, change the **Project Title** if you want to change how the project appears later.

Project Title
Is how the project will appear to users in runtime Client.

Authentication Profile
The profile to use for granting access to the project. For more information, see Security.

Default Database
All elements of the project will use this default database connection unless explicitly specified otherwise.

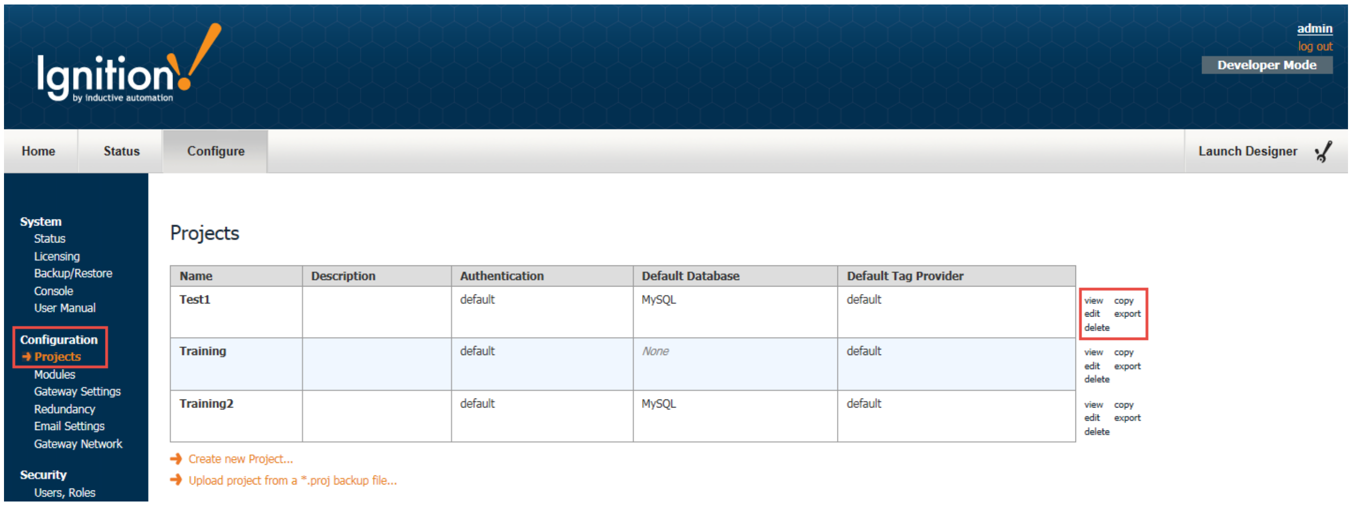
Default Tags Provider
The primary tag provider for the project. Most installations will likely only have one provider, but in situations where there are more than one, this is the provider that will be used by default.

Project Template
Gives you the option to choose a template for your project. Click the dropdown to see all the available options.

Managing Projects

You manage the projects from the **Configure** section of Gateway by going to **Configuration > Projects**.

The **Project** window is displayed and from here your can **view**, **edit**, **delete**, **copy**, or **export** each project. Look to the right of the project name, to see the link for these actions.



Projects

Name	Description	Authentication	Default Database	Default Tag Provider	
Test1		default	MySQL	default	view copy edit export delete
Training		default	*None*	default	view copy edit export delete
Training2		default	MySQL	default	view copy edit export delete

Create new Project...
Upload project from a *.proj backup file...

Note: There are other things you can do to manage your projects in the Designer, such as the Client Poll Rate and Publish Mode. See the Project > General Property Settings section for more information.

Viewing a Project

The **view** link on the **Project** window, takes you to the **Project details for 'Project Name'** page where you can choose to view the **Published** or **Stagi ng** version of the project.

Published vs Staging

Each project can have two distinct versions at the same time: the **Staging** version and the **Published** version. Having a published version and a staging version allows you to make changes to a project, save the changes, and test out the changes before "publishing" it to a production environment.

Usually, Vision module Clients run the published version of a project. However, by launching a Client in a special mode (from the Designer or from the Config section of the Gateway), you can launch a Client that runs the staging version of that project. This staging Client will receive updates on every save, where the production Clients receive updates only on publish. This lets you test out your changes to the project in an actual Client, which is more realistic than the Designer's preview mode.

Not all aspects that comprise a project use this system. It is primarily intended for systems such as the Vision module's Clients. Features that run persistently on the Gateway, such as Tags, the SQL Bridge's Transaction Groups, and Gateway-side scripting always run the most recently saved changes (the Staging version). Since these features by definition must run in exactly one place, they cannot be effectively "tested out" by simultaneously running a staging version alongside a published version.

In the Designer, choose the **Project > Properties** menu item, then go to the **Project > General** page to set the **Publish Mode**. By default, a new project is configured to be in **Auto** publish mode, however you can change it to **Manual** publish mode.

- In the default **Auto** mode, when you want to save your project, Designer lets you do a **File > Save and Publish**, which means both the Staging and the Published versions will be the same and identical.
- In **Manual** mode, when you want to save your project, Designer lets you do both a **File > Save** or a **File > Publish**. **File > Save** updates the Staging version, and **File > Publish** updates the Published version.

Project Versioning and History

Each project keeps a log of recent changes. These include both saves and publishes. Every save increments a number called the "edit count" for the project, which can be used like a serial number. The user, time, affected resources, and a commit messages (described next) are logged as well.

Commit Messages

A project may be configured to prompt the user making changes to describe those changes, either on every publish event, or on every save and publish event. These messages, called commit messages, are used to describe the changes that have been made to the project. By inspecting the project's history and reading these commit messages, you can learn what changes have been made to the project, for what reason, and by whom.

Editing a Project

To edit some of the project settings, go to the **Configure** section of Gateway, and then to **Configuration > Projects**. You will see a list of all your projects. Click **edit** to the right of project name in the list. The **Edit Project** page is displayed. You can now change some of the project settings such as project name, description, and title. You can also change connections to the Authentication profile, default database, and default tag provider.

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 backup or auto-backup.

To delete a project, go to the **Configure** section of Gateway, and then to **Configuration > 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 **Configure** section of Gateway, and then to **Configuration > 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.

Exporting and Importing a Project

There are two primary ways to export/backup or import/restore projects:

- Project export and import
- System backup and restore

Project export and import

Projects can be exported (backuped) and imported (restored) individually. Project exports only include project-specific elements, like windows and groups. They do not include Gateway settings, like database connections and device configurations.

To export the project, go to the **Configure** section of Gateway, and then to **Configuration > Projects**. You will see a list of all your projects. To the right of project name, click **export**. This exports your project as a *.proj backup file.

To import the project, go to the **Configure** section of Gateway, and then to **Configuration > Projects**. You will see a list of all your projects. Below the project table, next to the orange arrow, click the **Upload project from a *.proj backup file** link. Then choose your backup file and click **Upload**.

System backup and restore

You can backup a project as part of a full system backup. The system backup includes all of the projects in the system, in addition to all of the settings. Restoring a system backup replaces all current projects and settings on a Gateway. See Making Gateway Backups for more information.

Concurrently Working on a Project

Often in large projects there are multiple people working on the same project. Multiple Designers can be launched concurrently so everyone can work on the same project. When a particular resource is being edited, it will be locked, and the other designers won't be able to modify it. When the project is saved, the resource is unlocked. Concurrent edits are received by other designers only when you do a **File > Update Project**.

In this section ...

Launching Designer and Opening a Project

Now that we have set up some device and database connections (or not, if you skipped the last two steps as they were optional, after all), lets web-launch the Designer so you can work on your projects.

Web-launching is one of the best parts about Ignition. This is how we launch both the Designer, where you'll configure your projects, and the Ignition Vision Clients, where you can view the projects.

Web-launch technology lets you launch a full-fledged application with zero installation just by clicking a link on a webpage. This means that with Ignition, you'll only need to install the Gateway. All of your Designers and Clients do not need to be installed and they are always kept up-to-date.

Once you start using web-launched Clients, you'll wonder how you ever did without them!

Launching Designer and Opening a Project

Watch the Video

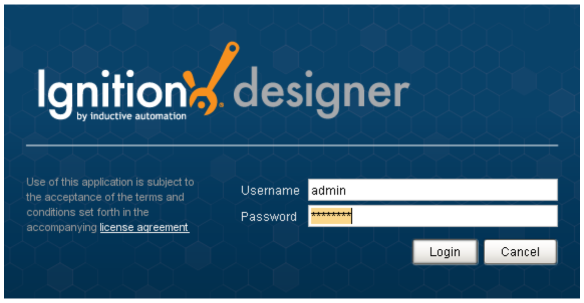
Before you successfully web-launch the Designer, make sure you have Java installed.

If you're on the computer that's running the Ignition Gateway, you already have Java installed, the Ignition installer makes sure of that. If you're on a computer that is accessing the Gateway over the network, the **Java Detection** panel on the bottom of the Gateway's **Home** page detects whether or not Java is installed.

Launching the Designer

1. Click **Launch Designer** on the upper-right of any Gateway page.
   Your web browser asks you if you want to Keep or Discard a **designer.jnlp** file.

2. Click on the **designer.jnlp** file to start the Designer.
   The Designer starts up and the login window is displayed.


Ignition designer
by inductive automation
Use of this application is subject to the acceptance of the terms and conditions set forth in the accompanying license agreement
Username
Password
Login
Cancel

Username: admin
Password: ********

3. Enter the default username and password for the Designer. is the same as for the Gateway Configuration section, enter
   Username: **admin**
   Password: **password**

   This assumes that this username and password were never changed between installing the database and creating a new project. If this username and password does not work then the default username and password are no longer valid. Please refer to Security to understand further how Ignition manages its users and their passwords.

4. Click on **Login**.
   The **Open/Create Project** window is displayed giving you two options, to create a new project or open an existing one. As we don't have any existing projects yet, we will create a new project.

To Continue on the step-by-step simple workflow...

After you **Launch the Designer and opened your project**, as described on this page, click the link below for Step 5 to continue.

Step 1. **Launch the Gateway**

Step 2. **Connect to a Device**

Step 3. **Connect to a Database**

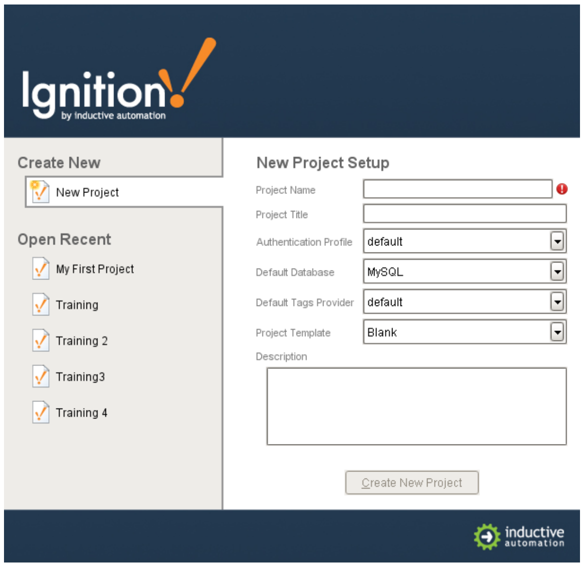
Step 4. **Launch the Designer**

Step 5. **Create Tags**

Step 6. **Launch Clients**

Step 7. **Add History to Tags**

Step 8. **Create a Graph and View Graphs in Clients**



Create New

New Project

Open Recent

My First Project

Training

Training 2

Training3

Training 4

New Project Setup

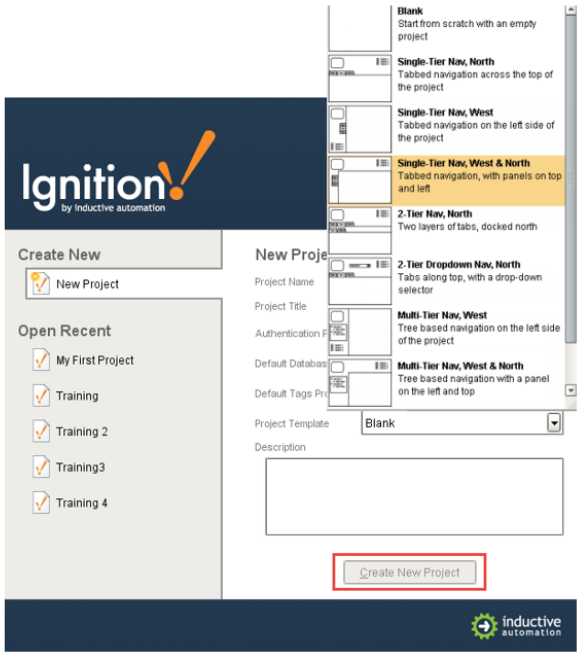
Project Name	
Project Title	
Authentication Profile	default
Default Database	MySQL
Default Tags Provider	default
Project Template	Blank
Description	

Create New Project

5. In **Project Name**, enter a name for your new project (no spaces!).

6. For **Project Template**, select a template from the drop-down list, for example, you can select the **Single-Tier Nav, West & North** template.

   Rather than starting from a blank project, the templates let you start from one of the eight skeleton projects that already have navigation windows setup. These projects are completely customizable and are designed to help you get started faster.



Blank
Start from scratch with an empty project

Single-Tier Nav, North
Tabbed navigation across the top of the project

Single-Tier Nav, West
Tabbed navigation on the left side of the project

Single-Tier Nav, West & North
Tabbed navigation, with panels on top and left

2-Tier Nav, North
Two layers of tabs, docked north

2-Tier Dropdown Nav, North
Tabs along top, with a drop-down selector

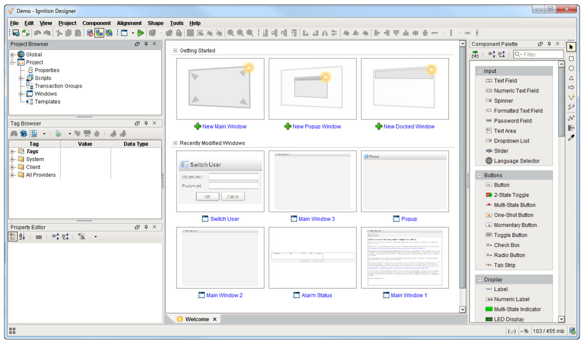
Multi-Tier Nav, West
Tree based navigation on the left side of the project

Multi-Tier Nav, West & North
Tree based navigation with a panel on the left and top

Project Template: Blank

Create New Project

7. Click on **Create New Project**.
   The Designer is now launched and you can work on your project!



Once in the Designer, your can create or open a project using the Designer's **File > Open** menu. The **Open/Create Project** window is displayed and you can follow the same steps as above to open existing projects or create a new one.

Troubleshooting

What is Web-launching

Web-launching is the mechanism by which Clients and and the Designers are opened on a computer. They are launched from the Ignition Gateway, you download and run them without requiring any installation steps.

How Web-Launching Works

Web-launching relies on Java Web Start technology. When the user clicks on a project or the Designer link in the Ignition Gateway (or embedded in a separate website), they download a small JNLP file that describes the application. When this file is opened (usually automatically), Java is invoked on the user's machine and directed to the remote application. The application is downloaded and cached, and then executed.
The running application (an Ignition Client or the Designer) communicates with the Gateway via HTTP. It is cached for increased subsequent launch speed, and can optionally install a link in the Start menu and on the desktop for easy access. By default a shortcut is created for you but this setting can be changed in the Java Control Panel.

Troubleshooting Web-launch Problems

There are a few common problems that can cause difficulties with web-launching. Fortunately, they are often easy to fix.

No Java Installed

Before you successfully web-launch the Designer, make sure you have **Java 8** installed, see http://www.java.com.

If you're on the computer that's running the Ignition Gateway, you already have Java installed, the Ignition installer makes sure of that. If you're on a computer that is accessing the Gateway over the network, the **Java Detection** panel on the bottom of the Gateway's **Home** page detects whether or not Java is installed

No Network Connection

Web-launched clients depend on network connectivity to connect to the server. If the network is unavailable, the client cannot be launched. This is often a problem with clients and the Designers launched directly from desktop/start menu links.

Cached References to Modified Servers

The cached projects/launch shortcuts contain the address of the Gateway machine. If the server is relocated, these links will no longer work. They can be updated by re-launching from the Gateway.

Still having problems?

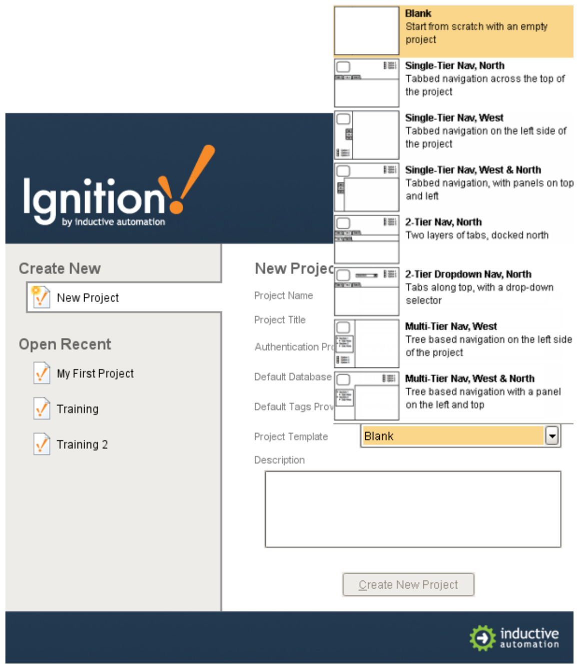
You can also see Native Client Launchers.

Next ...

- About Project Templates

About Project Templates

When creating a **New Project** in the Ignition Designer, you are given the option to choose from a number of project templates in the **Project Template** field of the **Open/Create Project** window.



Blank Start from scratch with an empty project

Single-Tier Nav, North Tabbed navigation across the top of the project

Single-Tier Nav, West Tabbed navigation on the left side of the project

Single-Tier Nav, West & North Tabbed navigation, with panels on top and left

2-Tier Nav, North Two layers of tabs, docked north

2-Tier Dropdown Nav, North Tabs along top, with a drop-down selector

Multi-Tier Nav, West Tree based navigation on the left side of the project

Multi-Tier Nav, West & North Tree based navigation with a panel on the left and top

Project Template: Blank

About Project Templates

[Watch the Video]

Rather than starting from scratch, you can choose an existing template to create your project much faster and easier. The following are the available **Project Templates** you can choose from:

Blank (Default)
Start from scratch with an empty project.

Single-Tier Nav, North
Tabbed navigation across the top of the project.

Single-Tier Nav, West
Tabbed navigation on the left side of the project.

Single-Tier Nav, West & North
Tabbed navigation with panels on top and left of the project.

2-Tier Nav, North
Two layers of tabs docked north of the project.

2-Tier Dropdown Nav, North
Tabs along top, with a dropdown selector, of the project.

Multi-Tier Nav, West
Tree-based navigation on the left side of the project.

Multi-Tier Nav, West & North
Tree-based navigation with a panel on the top and left side of the project.

Mobile Project
A project template optimized for a small Mobile device. You don't have to use this template to launch in a mobile device. If you have the Mobile module, you can use all of the project templates with your mobile device.

Next ...

- Using Designer User Interface

Using Designer User Interface

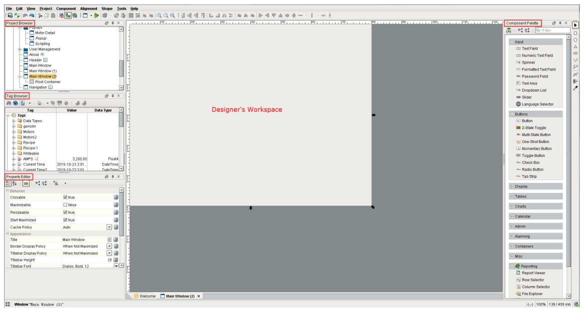
Designer's Workspace

The Designer is organized in panels around a central **workspace**. Some of these panels include: **Project Browser**, **Tag Browser**, and **Property Editor**. The workspace changes depending on the type of resource that you are currently editing. For example, if you are editing a Window, your workspace will be the Window Designer. If you are editing a Transaction Group, your workspace will be the Transaction Group Editor, and so on.

There are many dockable and dragable panels that surround the workspace, as well as the familiar menu bars and toolbars. The dockable panels can be rearranged as you wish. Each type of workspace can have panels that are only valid when that workspace is active. Each workspace remembers its own perspective, which is the docking arrangement of the panels around it.

If you have closed a panel and want to get it back, re-enable it from the **View > Panels** submenu. The **View > Reset Panels** will reset the user interface back to its default window arrangement.

On this page ...

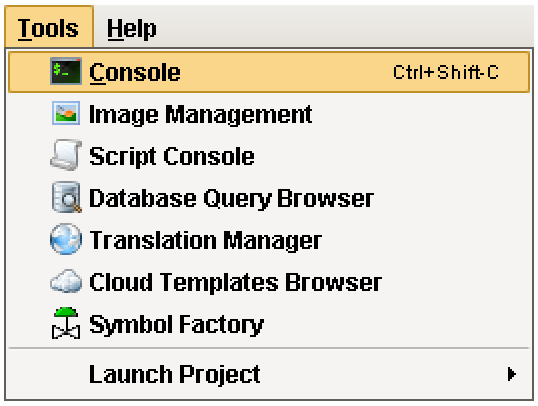


Using Designer User Interface

Watch the Video

Tools Menu

The **Tools** menu in the Designer provides some tools to help you when creating projects. The **Console**, **Image Management**, and **Symbol Factory** tools are described here.



Tools > Console

The Output **Console** is the script-writers best friend. It is a dockable panel, and you can open it via the **Tools > Console** menu or the **Ctrl-Shift-C** keyboard shortcut. The output console is most frequently used to test and debug Python scripts in Ignition.

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
```

Note: The output console is also available in the Vision Client from **Help > Diagnostics**.

See also: Scripting, Help Menu

Tools > 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 toolbar at the top 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 also: Images (png, jpg, gif)

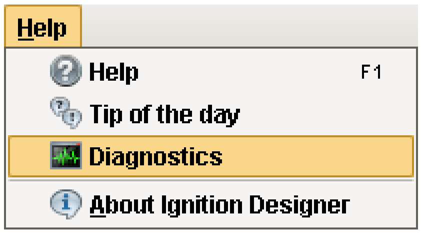
Tools > Symbol Factory

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.

Help Menu

Help > 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. These tabs or the troubleshooting features are as follows:

Performance Tab
Displays a number of small realtime charts that show various aspects of the currently executing Designer or Client's performance. These charts can be very useful to help troubleshoot performance issues, especially slow queries. 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.

The statistics shown on this page apply to the current session only, you can see the Gateway statistics from the **Status** section of the **Gateway**.

Console Tab
Displays the Ignition Output Console.

Log Viewer
Displays the logged events for the current Designer or Client session. Whenever errors occur, they are logged and displayed in this tab. This is a good

place to go when troubleshooting an issue, as any errors shown here may illuminate the cause of the problem. To view entries across all categories chronologically, uncheck the **Group Categories** checkbox.

Logging Levels
Determines the verbosity of a host of internal loggers. Most users will not use this tab unless prompted by a technical support representative.

Thread Viewer
Shows information about the currently running threads. Most users will not use this tab unless prompted by a technical support representative.

Connections
Displays the Gateway connection status as being connected or not.

Docking System

The Designer's docking system provides a very flexible user interface, allowing you to customize the layout as you wish. To re-arrange the dockable panels, simply drag on their title bars. As you are dragging the panel, use the highlighted border that appears to gauge where the panel will be moved to. 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 have 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.

Next ...

- About Communication Modes

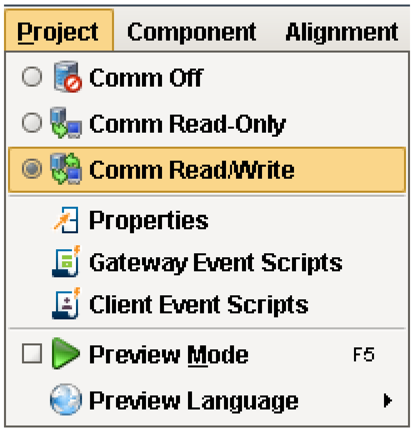
About Communication Modes

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.

INDUCTIVE UNIVERSITY

About Communication Modes

Watch the Video



To set or change the communication mode

There are two ways to set or change the comm mode in the Designer:

- Using the **Project** menu
- Using the **Project Properties** window

Using the Project menu

On the Designer toolbar, go to the **Project** menu and choose: **Comm Off**, **Comm Read-Only**, or **Comm Read/Write**. See below on this page for a description of these comm modes.

Note: The three icons in the toolbar under the **Project** command, correspond to the three comm modes.

Using the Project Properties window

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 the **Designer > General** setting section.

3. Under **Startup Options**, for the **Initial Gateway Comm Mode**, choose from the dropdown **Com m Off**, **Comm Ready-only**, or **Comm Read/Write**.
   Note: These property settings are saved on a per-project basis.

- **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, improving performance.
  Note: This setting does not affect the execution of a project's transaction groups, because transaction groups execute on the Gateway, not in the Designer.

- **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 mod e, see the Designer/General Properties section.

- **Comm Read/Write**
  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 attempti
ng to test a window's functionality in **Preview** mode.

Next ...

- Previewing the Project

Previewing the Project

Preview the project by testing and interacting with the screens in the Designer before deployment.

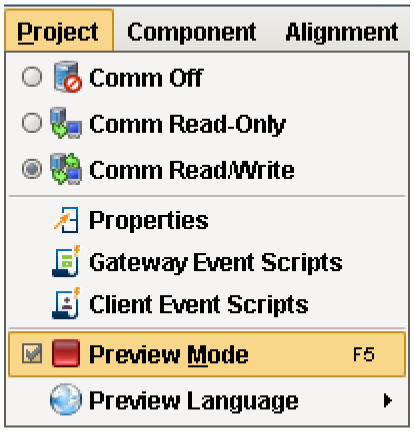
INDUCTIVE UNIVERSITY

Previewing the Project

Watch the Video

Preview Mode

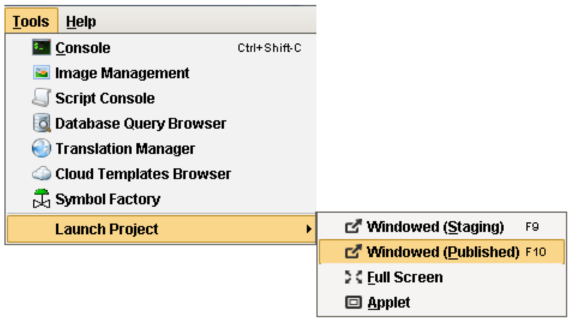
The window workspace operates in two distinct modes: design mode and preview mode. You can switch between these modes with the play/stop buttons in the toolbar or the **Project > Preview Mode** menu item. You can also use the **F5** key to toggle between the two modes.



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.

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 three launch modes via the **Tools > Launch Project** menu.



Next ...

- Using Find and Replace

Using Find and Replace

The **Edit > Find/Replace** menu command in the Designer is a very handy tool. You can search your entire project for specific components, properties, scripts, Transaction Groups and more. You can then use the replace feature to make mass changes to a project with very little effort.

INDUCTIVE UNIVERSITY

Using Find and Replace

Watch the Video

To Find and Replace

1. In the Designer, go to **Edit > Find/Replace** or use the shortcut **Ctrl-F**.
   The **Find/Replace** window is displayed.
2. Type what you're searching for in the textbox at the top to search through your project.
   You can use the wildcard character (*) which will match anything, and the single-character wildcard character (?).
   For example, to find all references to a tag that include the string `Motor`, you'd search for `Motor*`. This would match things like `Motor15`, `MotorHOA`, and so on, whereas the search query `Valve?Status` would match `Valve1Status` but not `Valve38Status`.
3. In the **Target** section, narrow down your search by choosing the options you are interested in.
   It is often useful to specify a narrow search target. Find/Replace searches through many different parts of a project as well as through Tags. The target settings let you specify exactly what Find/Replace should search through. By unchecking boxes in the Target section, you can avoid search results that you aren't interested in.
4. 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.
5. 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.
6. 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.
7. 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.

Next ...

- Using the Query Browser

Using the Query Browser

The **Query Browser** is a very convenient tool that lets you make simple edits in a database table, and interact with all of the databases that Ignition is connected to, for example running queries and browsing tables and schemas. Because Ignition is so heavily integrated with databases, it is very common in the course of project design to need to inspect the database directly, or to experiment with a SQL query to get it just right.

INDUCTIVE UNIVERSITY

Using the Query Browser

Watch the Video

To Edit the Database using the Query Browser

1. In Designer, go to the **Tools > Database Query Browser** menu item.
   The **Database Query Browser** is displayed which lets you see a list of databases that Ignition is connected to plus a text box area so you can enter your queries.

2. Choose a database from the list under **Schema**.
   Notice the database is expanded and the Query textbox shows a SELECT query which includes the table's primary key(s) such as, `Select * From database_name`.

3. Click **Execute**.
   See the data that gets filled in the **Resultset** tab.

4. Click **Edit** to change the values in the Resultset, and click **Apply** to commit your edits or **Discard** to back out.
   Note: You can **Edit** the values only when the table is displaying the **id** numbers (table's primary keys), otherwise you can not **Edit** the table values.

5. To verify the changes you made, click **Execute**.
   You should now see what you have modified is actually applied.

To Monitor the Database using the Query Browser

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.

Next ...

- Using the Popular Keyboard Shortcuts

Using the Popular Keyboard Shortcuts

You can interact with screens and more by using some of the popular keyboard shortcuts in the Designer.

To...	Use these Keyboard Shortcuts
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.	Alt + ↑ ↓ ← →
Resize Right Moves the right edge of the component left or right. Add Alt to use the alt-nudge distance.	Shift + ← →
Resize Bottom Moves the bottom edge of the component to top or bottom.	Shift + ↑ ↓
Resize Left Moves the left edge of the component left or right.	Ctrl + Shift + ← →
Resize Top Moves the top edge of the component to top or bottom.	Ctrl + Shift + ↑ ↓
Move Forward/Move Backward Moves selected component(s) in the Z-order.	PgUp / PgDn
Move to Front/Move to Back Moves selected component(s) to Front or Back.	Home / End
Copy-Move Copies the component when holding Ctrl while doing a mousemove.	Ctrl + ✥
Orthogonal-Move Restricts to only moving straight up, down, left, or right when holding Shift while doing a mousemove.	Shift + ✥
Selection-Move Drags the current selection when holding Alt while dragging.	Alt + ✥

Using the Popular Keyboard Shortcuts

Watch the Video

Copy-Axis-Move Combines copy move with axis move.	Ctrl + Shift + ✥
Proportional Resize Resizes a component while maintaining its aspect ratio.	Ctrl + ⤡
On-Center Resize Resizes the selection using the center as the anchor point.	Shift + ⤡
Select All Selects all components that are siblings of the selected component.	Ctrl + A
Select Same Type Selects all components that are siblings of the selected component and the same component type.	Ctrl + Shift + A
Select Same Type in Window Selects all components in a window that are the same type as the selected component.	Ctrl + Alt + Shift + A
Layout Opens Layout Constraints to let you specify layout for the component(s).	Ctrl + L
Size & Position Opens Size & Position to let you specify exact size and position for selected component or window.	Ctrl + P
Customizer Opens Custom Properties to let you configure complex component properties.	Ctrl + U
Customizer 2 Opens the second Customizer if available.	Ctrl + 2
Jython Opens Component Scripting to let you configure actions for component or window.	Ctrl + J
Security Opens Security Settings to let you set security for Component(s) or window.	Ctrl + E
Script Modules Opens Script Module Editor for reusable Jython functions.	Ctrl + M
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 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
Find/Replace Opens Find/Replace to let you Search and replace the project based on string, pattern, or regex.	Ctrl + F
Delete Deletes the current selection.	Del
Snap to Grid Toggles whether or not moving and resizing components snaps to the grid.	Ctrl + G
Snap to Guides Toggles whether or not	Ctrl + Shift + G

moving and resizing components snaps to guides.	
Console Opens the Output Console.	Ctrl + Shift + C
Help Launches the Ignition User Manual in a web browser.	F1
Preview Mode Toggles preview/design mode.	F5
Rename Renames the selected item (tag, window, transaction group, component, and so on).	F2
Launch Staging Project Launches the staging client from the Designer.	F9
Launch Published Project Launches published client from the Designer.	F10
Update Project Updates project to server. Receives concurrent edits from other Designers.	Ctrl + Shift + U
Zoom Zooms in/out in the Designer.	Ctrl +
Lasso Select Draws a lasso and every component in the path will be selected.	Alt +

Next ...

- add link here

Project Properties

There are a number of properties you can set or customize for your projects, the Designer, the Clients, and your Mobile device. For example, there are properties for setting the touchscreen mode, customizing a client's auto-login, or configuring how the clients receive updates.

In this section ...

About Project Properties

The property settings on the **Project Property** window apply to the whole project. Here you can set the properties for your project, the Designer, the Client, and your Mobile device.

INDUCTIVE UNIVERSITY

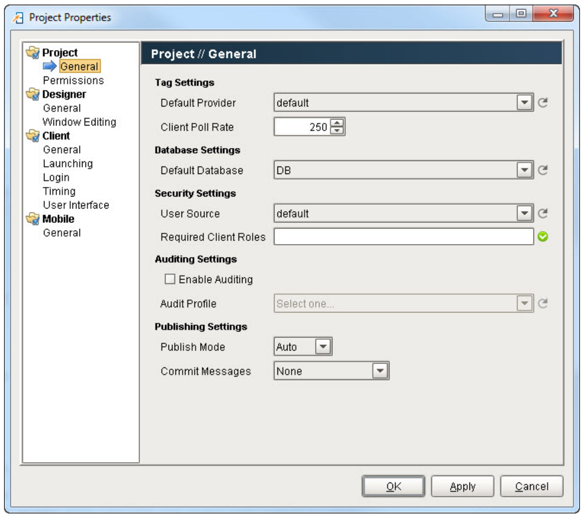
About Project Properties

Watch the Video

To set the Properties

In the Designer, either double-click on the **Project > Properties** node in **Project Browser**, or click on **Pro ject > Properties** on the top menu.

The **Project Properties** window is displayed.



Project // General

Tag Settings

Default Provider	default
Client Poll Rate	250

Database Settings

Default Database	DB

Security Settings

User Source	default
Required Client Roles	

Auditing Settings

☐ Enable Auditing

Audit Profile	Select one...

Publishing Settings

Publish Mode	Auto
Commit Messages	None

For a detailed description on how to set these properties see the following links:

- **Project**
  General
  Permissions
- **Designer**
  General
  Window Editing
- **Client**
  General (Update Mode, Touch Screen)
  Launching
  Login
  Timing
  User Interface
- **Mobile**
  General

Project/General 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 (see next). For example, there are default settings for the tag provider, database, publish mode, initial comm mode, window editing, and Client launching.

Note: 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 **Configuration > Projects** section, click on the **edit** link next to the project.

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 designed 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`**.

The following are a description of the project properties you will see under **Project/General** in the **Project Properties** window:

Tag Settings

- Default Provider

  The Tag provider chosen here will act as the project's default provider. To use the default provider, simply omit the source section of a tag path, or leave it blank, for example: `Path/To/MyTag` or `[]Path/To/MyTag`.

See also: Keeping Tags Organized

- Client Poll Rate
  The Client poll rate 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

- User Source
  Choose the User Source that governs this project's security. The User Sources are all defined in Gateway's **Configure** section under the **Security > User,Roles** page.

- Required Client Roles
  This profile is for Client logins. You can also 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

Auditing Settings

- Enable Auditing
  If auditing is enabled, audit events will be stored that relate to this project in the chosen audit profile.

- Audit Profile
  The audit profile which will store the audit events when auditing is enabled.

Publishing Settings

- Publish Mode
  Here you specify whether or not a project is to have a separate staging and published versions.
  Manual: In this mode, you can do a **File > Save** to update the Staging version, and then a **File > Publish** to update the Published version.
  Auto: In this mode, the only command available for saving your project is **File > Save and Publish** which saves and then publishes, so the two Staging and Published versions are always the same.

See also: Viewing a Project

- Commit Messages
  You can specify whether or not commit messages are required, and if so, under what conditions. You can specify the condition as: **None**, **On Publish**, or **OnSaveandPublish**.

Designer/General Properties

These property settings are saved on a per-project basis.

Startup Options

- Initial Gateway Comm Mode
  Designer starts up in the default Comm Read-Only mode. You can choose any mode you want the Designer to startup in. The options are: **Comm Off**, **Comm Ready-only**, **Comm Read/Write**.

See also: About Communication Mode

Designer/Window Editing Properties

These options affect the operation of the Designer as it applies to the Vision module's window design.

Window & Template Committing

- Commit on Close
  Prompt: By default, every time you close a window, you are prompted whether or not you wish to *commit* the window.
  Always: This option allows you to skip the commit prompt, opting to always commit the window on close.

- Template Auto-Commit
  On: Automatically commits template masters when switching to another window or template.
  Off: In this option you commit the templates manually, it reverts all changes to the last time the window was committed.

Component Manipulation

These options affect how the user interface acts in manipulating components.

- Constrain to Parent Container Bounds
  Disabling the constraint on parent bounds allows you to position components outside of their parents bounds, which can be helpful in advanced layouts.
- Nudge Distance
  The number in this box is the distance (in pixels) that a nudge moves (when using the arrow keys) or resizes a component.
- Alt-Nudge Distance
  The number in this box is the distance (in pixels) that a alt-nudge moves (when using the arrow keys plus the Alt key) or resizes a component. Altering the handle opacity can be helpful when dealing with lots of very small components, so that you can see through the resize handles to align the component perfectly.

Default Component Layout

The layout constraints specified here are the layout constraints used for all newly added components.

Layout Mode

- Relative
  This mode makes a component's size and location relative to its parent's size and location. When the parent changes size, the component will change accordingly. This creates components that auto-scale.
- Anchored
  This mode makes the edge of a component's two axes (horizontal and vertical) anchored to the edge or edges of its parent.

Relative Layout Options

- Maintain Aspect Ratio
  If selected, the component's original aspect ratio is preserved. Otherwise, it can stretch tall or wide.
- Center
  When maintaining the aspect ratio, centers the component with respect to its parent.
- Leading
  When maintaining the aspect ratio, aligns the component with the paper leading edge.
- Trailing
  When maintaining the aspect ratio, aligns the component with the parent's trailing edge.
- Scale Font
  If selected, the component's font will scales along with its size as the relative layout adjusts the component.

Default Color Mapping

The color mapping defined here will be the initial color mapping when configuring a new **number-to-color** property binding.

Client/General Properties

These properties apply to the Vision Client in general.

Timezone Behavior

The Vision Client can emulate any timezone. By default, it will appear to be in the same timezone as the **Gateway Timezone**. This has the effect of all Clients behaving the same, regardless of the timezone setting on the Client's host operating system. Depending on your project's requirements, this may not be optimal. You can have the Client use the host's timezone by choosing the **Client Timezone** option, or you may specify an explicit timezone for all Clients to emulate.

Update Mode
This setting affects how Clients receive updates when the project is saved.

- Notify
  This is the default, which means that all running Clients will display a yellow information bar at the top of their display that notifies the operator that an update is available. The update gets installed when the operator clicks on the yellow bar.

- Push
  This option automatically pushes all project changes and updates to all running Clients with no operator interaction. This is often desirable when a Client is running in a situation where keyword and mouse access is inconvenient, such as in a large overhead display.

Touch Screen

All Clients can operate in touch-screen mode. When Touch Screen mode is enabled, clicking on numeric and text entry boxes will pop up on-screen keyboards that can be used for data entry.

- Touch Screen Mode Enabled
  By default, this is enabled, which means an operator can activate the mode on the startup screen.

- Touch Screen Mode Active on Startup
  This option sets the Clients to start up with the touch-screen mode active.

These settings are helpful for mixed-use projects, that is, those that are launched on both touch-screen devices and traditional computers and laptops.

Data

- Disable Tag History Data Cache
  The Clients normally maintain a cache of data retrieved from Tags History, improving repeat operations on graphs and tables. When this option is disabled, no data is cached, and the full queries execute against the Gateway each time data is required.

Client/Launching Properties

These properties apply to the Vision Client launch process.

Gateway Launch Page

- Default Launch Mode
  The default launch mode determines what kind of launch occurs when the user hits the **Launch** button that appears next to the project in the Gateway **Home** page. Each launch mode can also be enabled individually, which will turn the launch button into a split-button, allowing the user to choose the launch mode.

- Windowed Button/Full Screen Button/Applet Button
  These properties affect how the launched project will display when launched through one of the Java Web Start launch modes: Windowed, Full Screen, or Applet.

Launch Icon
The image specified here is used to represent the project on the launch page and desktop shortcut. This needs to be a path to an image that has been uploaded to the Gateway. Use the browse button to choose or upload a new image.

Windowed Properties

- Vendor
  This property is displayed as the project is launching through Java Web Start, as well as in the JWS application manager.

- Homepage
  This property is the height of the Client window when launched in Windowed mode.

- Width
  The width of the Client window when launched in Windowed mode.
- Height
  The height of the Client window when launched in Windowed mode.
- Screen Index
  The number here is the index of the screen to use for Full Screen mode, it starts at 0. This property is not supported on all operating systems.
- Start Maximized
  If the box is selected, when the Client is launched in Windowed mode, it starts maximized. Note that this is not the same thing as Full Screen mode, which is only available when the Client is launched in Full Screen mode. In full-screen mode, the width, height, and start maximized properties have no effect. When launched in Full Screen mode, the user is given an **Exit** button on the login screen by default. For terminals where the application should not be exited, this button can be removed by checking the **Hide Exit Button** box.
- Start Centered
  If the box is selected, when the Client is launched in Windowed mode, it starts centered.
- Hide Exit Button
  If the box is selected, when the Client is launched in Full Screen mode, the exit button is hidden to prevent the application from closing.

Applet Properties

- Width
  The width of the Client when launched in Applet mode.
- Height
  The height of the Client when launched in Applet mode.

Client Memory
The Initial and Maximum properties govern how the Client uses RAM resources on its host machine.

- Initial
  The initial memory setting is how much memory the Client will require on startup. While this is typically left alone, boosting it a bit can improve performance somewhat.
- Maximum
  The maximum memory setting sets a cap on how much memory the Java VM is allowed to use. When you have launched a Client on a machine with plenty of RAM, you'll also need to boost this setting to allow the Client to use more RAM.

See also: Images, Launching Clients

Client/Login Properties

These properties affect how the Vision Client's login process behaves and appears.

Login Screen
These properties affect the appearance of the login screen. By default, the title area of the login screen will contain the project's title (or its name, if the title is blank), along with the project's description. You can override this by entering a welcome message for your project here. You may use HTML to format the message. You can also set an image to use instead of the Ignition logo on the login screen's header. You may also override the text used in the login controls.

Auto Login
By enabling auto-login, you can have the launched Client skip the login process. The Client will log in behind the scenes using the credentials supplied here. If they fail, the login screen will be presented.

Client/User Interface Properties

These properties affect how the Vision Client appears and behaves while it is running.

Minimum Size

- Width and Height
  Typically, a Vision Client is designed to run on multiple different resolution and sizes of monitors. The various component layout features help design elastic screens, but sometimes you need to set a lower bound as to how small you'll allow the Client's usable area to shrink. This is what the Minimum Size settings are for. You can see these settings visually represented in the Designer as lines on the Vision workspace. Whenever the usable space shrinks smaller than these bounds, scrollbars will appear, capping the width and height to these minimums. This defaults to 800x600.

Client Background Color
This option allows you to specify the color of the Vision workspace, which will be visible when not obscured by windows.

Client Menu
These options allow you to alter the appearance, or remove completely, the menu bar that appears in a running Vision Client.

See also: Component Layout, Menubar

Next ...

- About Client Updates

About Client Updates

When a Client is launched, the most recent version of the project is used. If you make changes to the project and save the changes (by doing a **File > Save**) while a Client is open, the Client is notified about the updates. The Clients can receive the updates in two different mode: **Notify** or **Publish** mode.

- **Notify**
  This is the default mode to automatically notify operators when project updates are available. In this mode, every time you update a project in the Designer and save the changes, Clients will display a yellow information bar at the top of their display. This yellow bar notifies the operator that an update is available. By clicking on the notification banner, the new project modifications are downloaded and applied.
- **Push**
  When you save your changes in the Designer, this mode automatically pushes all project changes and updates to all running clients with no operator interaction, that is, the new version is downloaded and applied automatically. This is often desirable when a Client is running in a situation where keyword and mouse access is inconvenient, such as in a large overhead display.

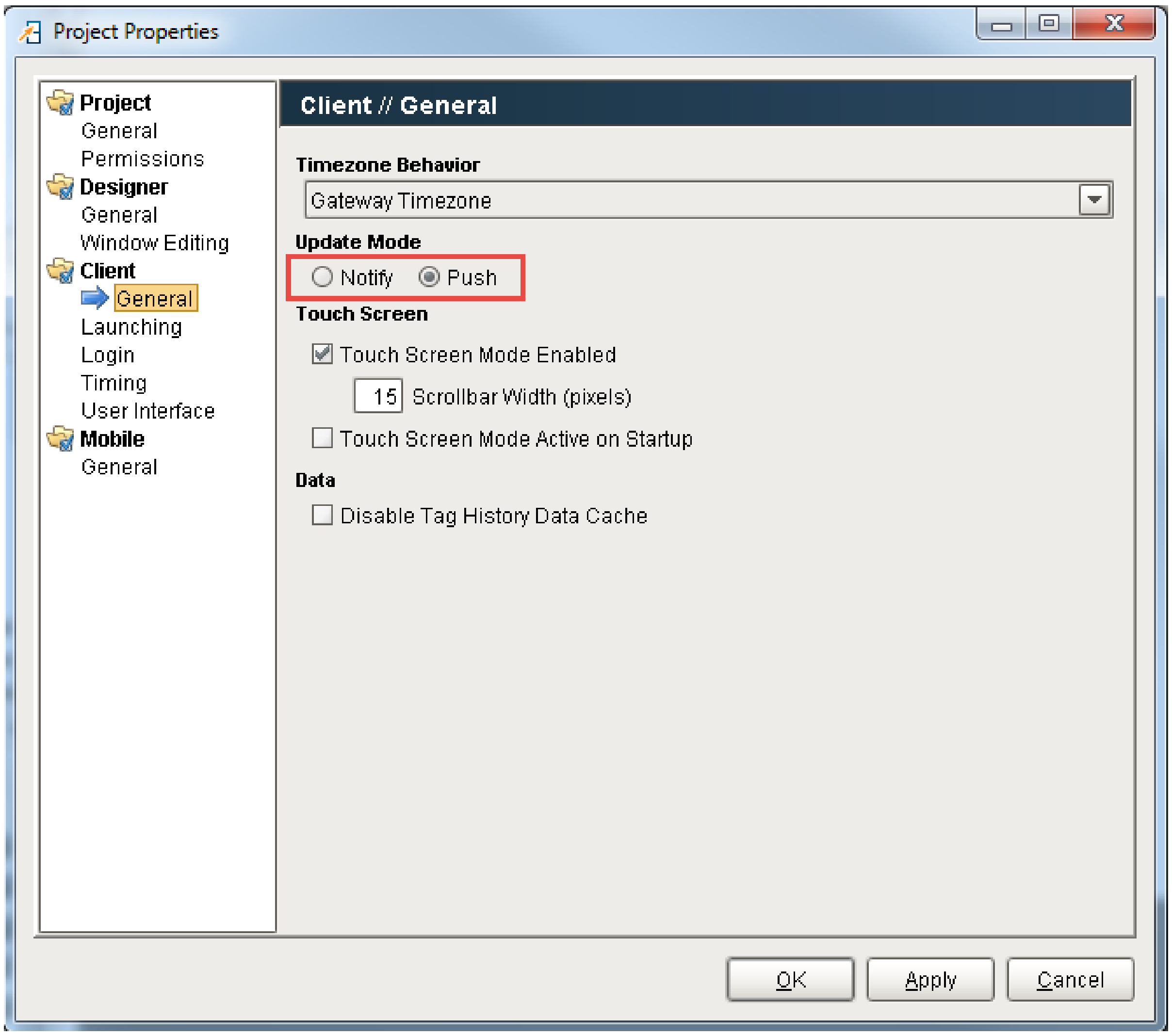
INDUCTIVE UNIVERSITY

About Client Updates

Watch the Video

To set the Client Update modes

1. In the Designer, from Project Browser, double-click on **Project > Properties**, or from the top menus go to **Project > Properties**.
   The **Project Properties** window is displayed.
2. Go to **Client > General** page, set either **Notify** or **Push** as your client update mode.



Timezone Behavior: Gateway Timezone

Update Mode: Notify [✓ Push]

Touch Screen

[✓] Touch Screen Mode Enabled

15 Scrollbar Width (pixels)

[ ] Touch Screen Mode Active on Startup

Data

[ ] Disable Tag History Data Cache

Next ...

- Using Touchscreen Mode

Using Touchscreen Mode

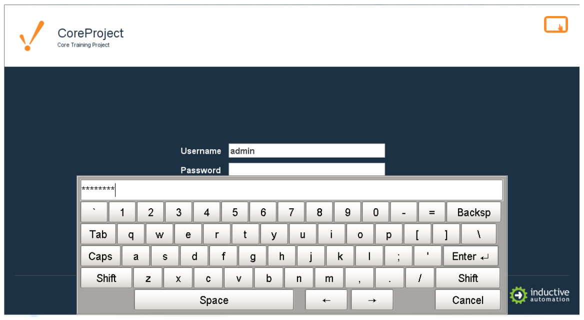
It is very common to deploy Ignition Vision projects on touchscreen computers. Often, these are industrial panel-pcs acting as Human Machine Interface (HMI) or Operator Interface Terminal (OIT). The touchscreens simply act like a mouse input device and usually don't have a keyboard attached. For this reason, all of the input components in Vision are touchscreen-enabled.

Under normal circumstances, you don't have to do anything special other than enable touchscreenmode on your project. This will allow the operator to enable touchscreen mode when they log in. You can also enable touchscreen mode via scripting.

INDUCTIVE UNIVERSITY

Using Touchscreen Mode

Watch the Video


CoreProject
Core Training Project
Username admin
Password
****
` 1 2 3 4 5 6 7 8 9 0 - = Backsp
Tab q w e r t y u i o p [ ] \
Caps a s d f g h j k l ; ' Enter
Shift z x c v b n m , . / Shift
Space ← → Cancel
Inductive automation

Touchscreen-enabled input components all have an expert level property called **Touchscreen Mode**. Normally, this is set to Single-Click, meaning that the touchscreen keyboard or numeric keypad (depending on the type of input component) will appear on a single click in that component. You can also change this to Double-Click, which should be self-explanatory, or None. None means that automatic touchscreen support is disabled for this component. You may want to set this to handle touchscreen logic via scripting.

To handle touchscreen logic via scripting, the general pattern is to respond to a mouse event, popup up a keyboard, and then set the component's value to whatever was entered in the keyboard. For example, for a text field, you would write a script like this:

```
if system.gui.isTouchscreenModeEnabled():
currentText = event.source.text
newText = system.gui.showTouchscreenKeyboard(currentText)
```

See also: Client/General Properties, system.gui.setTouchscreenModeEnabled

Touchscreen support is built into Ignition. Turn it on through the Project Properties or scripting.

To turn on the touchscreen mode

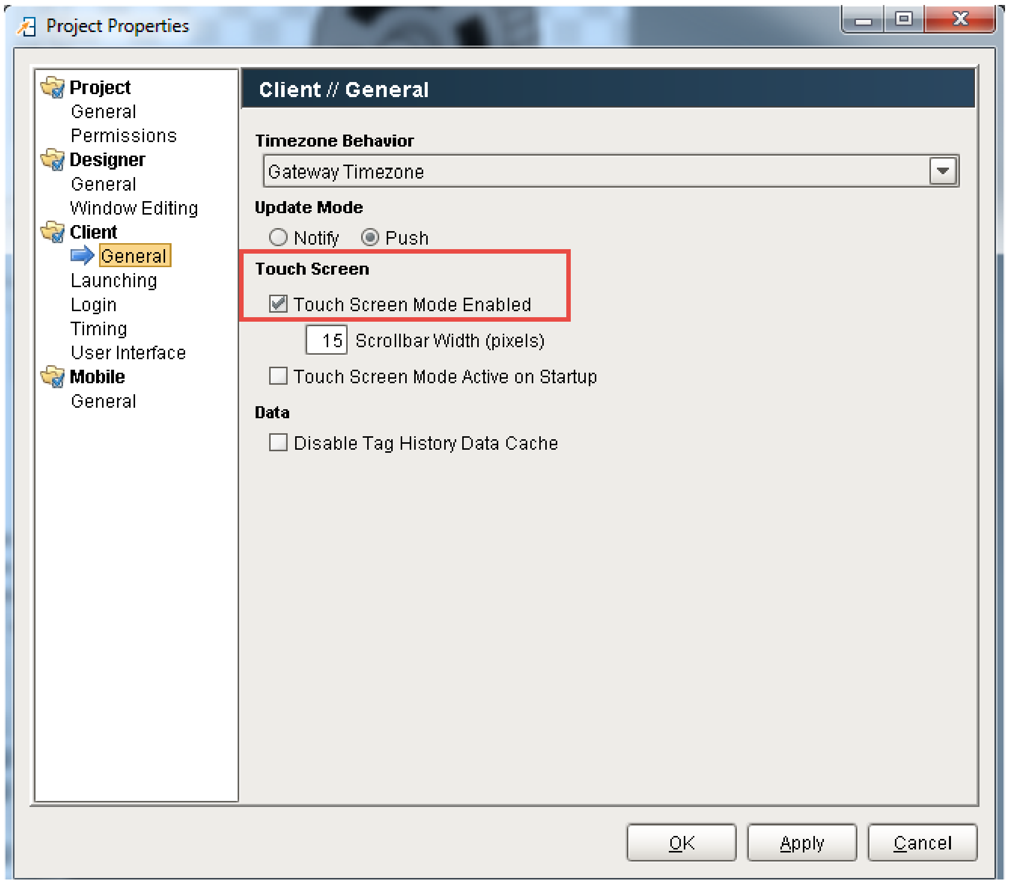
1. In the Desinger, from Project Browser, double-click on **Project > Properties**, or from the top menus go to **Project > Properties**.
   The **Project Properties** window is displayed.

2. Go to **Client > General** page, to see the **Touch Screen** options.

   Touch Screen Mode Enabled
   By default, this is enabled, which means an operator can activate the mode on the startup screen. All Clients can operate in touch-screen mode. When Touch Screen mode is enabled, clicking on numeric and text entry boxes will pop up on-screen keyboards that can be used for data entry.

   Touch Acreen Mode Active on Startup
   This option sets the Clients to start up with the touch-screen mode active.

These settings are helpful for mixed-use projects, that is, those that are launched on both touch-screen devices and traditional computers and laptops.



Project Properties

Project
- General
- Permissions

Designer
- General
- Window Editing

Client
- General
- Launching
- Login
- Timing
- User Interface

Mobile
- General

Client // General

Timezone Behavior

Gateway Timezone

Update Mode

Notify [✓ Push]

Touch Screen

[✓] Touch Screen Mode Enabled

15 Scrollbar Width (pixels)

[ ] Touch Screen Mode Active on Startup

Data

[ ] Disable Tag History Data Cache

Next ...

- Setting up Client Auto Login

Setting up Client Auto Login

Clients can log in automatically when launched, once you specify the **Auto Login** settings in the **Client /Login** section of the **Project Properties** window.

To set the Clients to automatically login

1. In Designer, go to **Project > Properties**.
   The **Project Properties** window is displayed.
2. Go to the **Client > Login** page, and select the **Enable Auto-Login** box.
3. Type any Username and Password you want, for example you can type:
   Username: **guest**
   Password: **password**
4. Click **OK** and select the **File > Save and Publish** to save your changes to the Gateway and publish your project.
5. Go to **Tools > Launch Project > Windowed (Published).**
   This will launch the Client and automatically login you in.
6. If you want to switch user, from the top-right of the Client window, click on **Switch User**, enter your **username** and **password**, and click **OK**.

Setting up Client Auto Login

Watch the Video

Next ...

- Setting up Mobile Auto Login

Setting up Mobile Auto Login

Mobile Clients can log in automatically when launched once you specify the **Auto Login** settings in the **M obile/General** section of the **Project Properties** window.

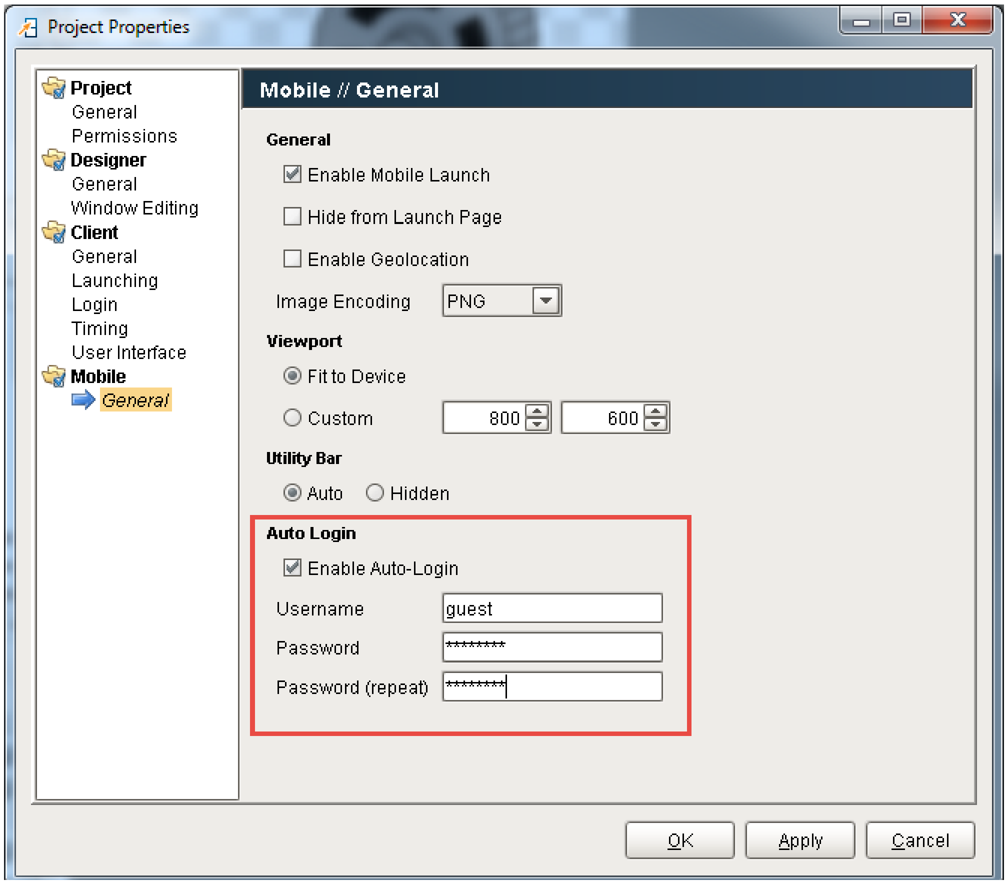
INDUCTIVE UNIVERSITY

Setting up Mobile Auto Login

Watch the Video

To set the Mobile Clients to automatically login

1. In Designer, go to **Project > Properties**.
   The **Project Properties** window is displayed.
2. Go to the **Mobile > General** page, and select the **Enable Auto-Login** box.
3. Type any Username and Password you want, for example you can type:
   Username: **guest**
   Password: **password**
4. Click **OK** and select **File > Save and Publish** to save your changes to the Gateway and publish your project.
5. Go to the Gateway **Home** page, find the **Mobile Launch** section, and click on the link.
   This launches the Client in your mobile device and automatically logs you in the web browser.



Mobile // General

General

[✓] Enable Mobile Launch
[ ] Hide from Launch Page
[ ] Enable Geolocation

Image Encoding: PNG

Viewport

[✓ Fit to Device] Custom 800 600

Utility Bar

[✓ Auto] Hidden

Auto Login

[✓] Enable Auto-Login

Username	guest
Password	*********
Password (repeat)	*********

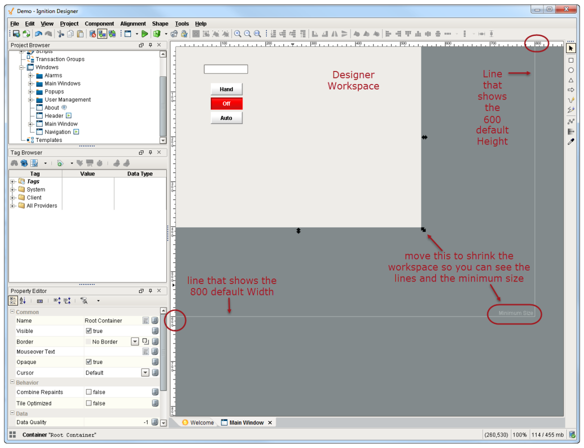
Next ...

- Setting Client Minimum Size

Setting Client Minimum Size

Typically, a Vision Client is designed to run on multiple different resolutions and different sizes of monitors. The various component layout features help design dynamic screens, but sometimes you need to set a lower bound as to how small you'll allow the client's usable area to shrink so that the content remains readable. This is done with the **Minimum Size** settings, which by default it is set to 800X600.

You can see the Minimum Size settings visually in the Designer as lines on the Vision workspace. Whenever the usable space shrinks smaller than these bounds, scrollbars will appear, capping the width and height to these minimums. This defaults to 800x600 is shown as a square that says Minimum size on the right corner.


Designer Workspace
Line that shows the 600 default Height
move this to shrink the workspace so you can see the lines and the minimum size
line that shows the 800 default Width
Minimum Size

To set the Clients minimum size

1. In Designer, go to **Project > Properties**.
   The **Project Properties** window is displayed.

2. Go to the **Client > User Interface** page.
   You can see the **Minimum Size** defaults are set to **800X600**.
   These values set the minimum size for the client workspace. If client is made to be narrower than this value, scroll bar will appear.

INDUCTIVE UNIVERSITY

Setting Client Minimum Size

Watch the Video

Next ...

- Hiding Client Menu Bar

Hiding Client Menu Bar

The Client screens display a menu bar with a number of commands on them. You can completely or partially hide this menu bar.

To hide the Client Menu Bar

1. In Designer, go to **Project > Properties**.
   The Project Properties window is displayed.
2. Go to the **Client > User Interface** page.
   On this page, under **Client Menu**, there are two checkboxes: **Hide Menu Bar** and **Hide Windows Menu**.
3. To hide only the Windows menu, select the **Hide Windows Menu** checkbox.
4. To hide all the menus, select the **Hide Menu Bar** checkbox.
5. Logout from the Client and log back in to see the menu bar changes.

Hiding Client Menu Bar

Watch the Video

Next ...

- Customizing Client Login Screen

Customizing Client Login Screen

All of the information shown on the Client Login screen can be modified. You can customize the text fields for the description, image, and username/password on the login screen to fit your needs.

INDUCTIVE UNIVERSITY

Customizing Client Login Screen

Watch the Video

To customize Client Login screen

1. In Designer, go to **Project > Properties**.
   The **Project Properties** window is displayed.

2. Go to the **Client > Login** page.
   On this page, under **Login Screen**, you can see all the text boxes that you can customize.

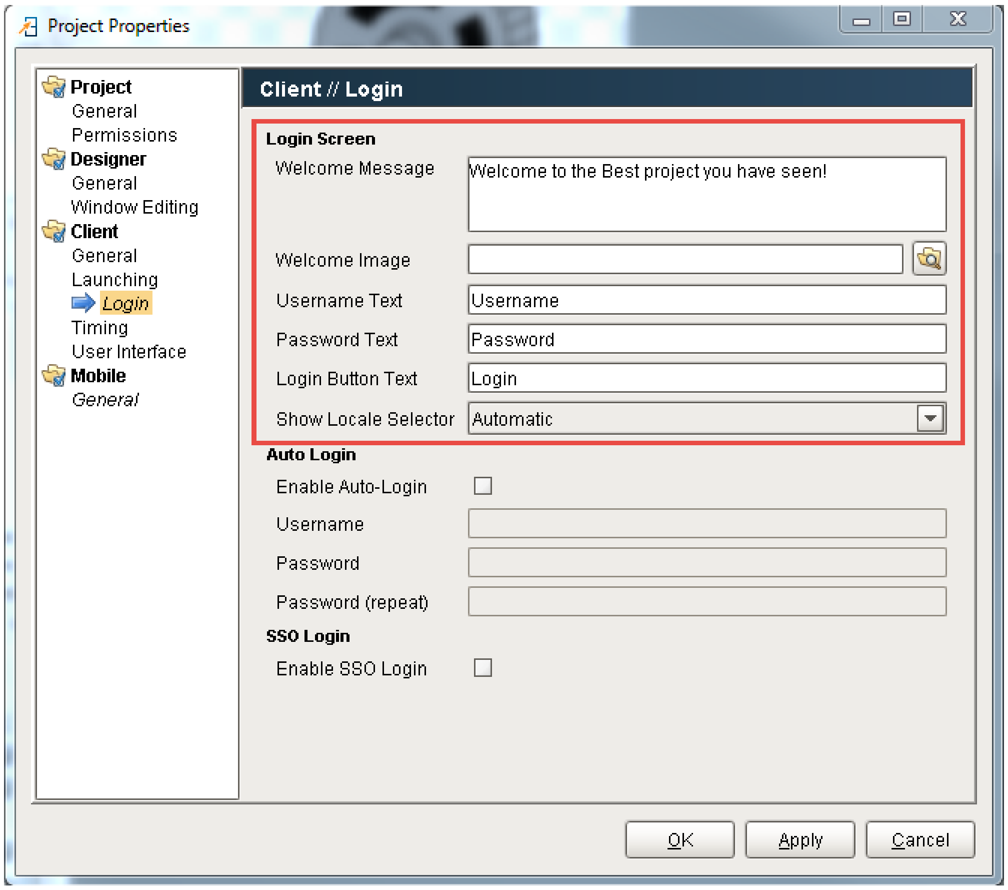
3. For example, you can change them by entering:

   Welcome Message: **Welcome to BEST project you have seen!**
   Welcome Image: click on the icon to the right of the text field and pick an image
   Username Text: **Enter your Username**
   Password Text: **Enter your Password**
   Login Button Text: **Click to Login**

4. Select the **Always Show Locale Selector** check box.
   When there is a **Language Selector** component in your project and you have this selected this option, you will see a **Language** box on the Client login page which gives you the option to choose a language.

5. Click **OK** and save your project with **File > Save and Publish**.

6. Open the Client to see all your changes to the Login screen.



Login Screen

Welcome Message	Welcome to the Best project you have seen!
Welcome Image	
Username Text	Username
Password Text	Password
Login Button Text	Login
Show Locale Selector	Automatic

Auto Login

Enable Auto-Login	☐
Username	
Password	
Password (repeat)	

SSO Login

Enable SSO Login	☐

Next ...

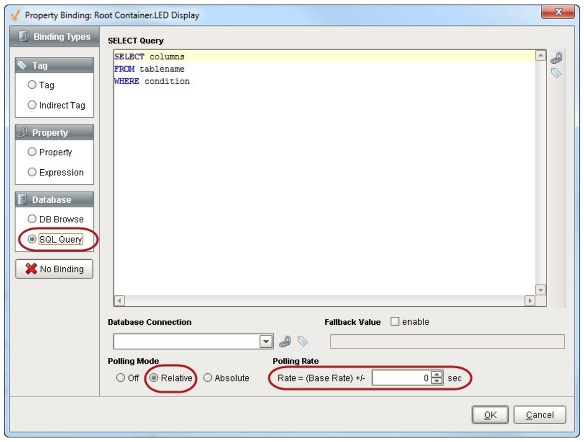
- Setting Project Polling Base Rate

Setting Project Polling Base Rate

SQL Query bindings have three **Polling Mode** options: Off, Relative, and Absolute. The **Relative** mode has an adjustable **Base Rate** setting. You can set this **Polling Base Rate** so it applies to all the SQL Query bindings for the whole project.

Setting Project Polling Base Rate

Watch the Video



Property Binding: Root Container.LED Display

Binding Types

Tag: ○ Tag ○ Indirect Tag

Property: ○ Property ○ Expression

Database: ○ DB Browse [✓ SQL Query]

No Binding

SELECT Query

```
SELECT columns
FROM tablename
WHERE condition
```

Database Connection

Fallback Value ☐ enable

Polling Mode: ○ Off [✓ Relative] ○ Absolute

Polling Rate: Rate = (Base Rate) +/- 0 sec

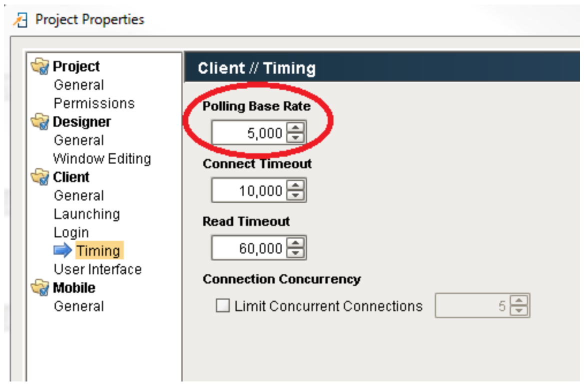
OK Cancel

To set the Polling Base Rate for the whole project

The project's base polling rate as set in the Project Porperties window will ONLY apply to the SQL bindings that are polling at the Relative Mode. Therefore, before you set the Polling Base Rate for the project, select the Relative mode as the Polling Mode for all the SQL Query bindings you wish.

1. In Designer, from the toolbar go to **Project > Properties**.
   The **Project Properties** window is displayed.

2. Go to the **Client > Timing** page.
   You can see the **Polling Base Rate** along with other timing settings on this page.



Project Properties

- Project
  - General
  - Permissions
- Designer
  - General
  - Window Editing
- Client
  - General
  - Launching
  - Login
  - Timing
  - User Interface
- Mobile
  - General

Client // Timing

Polling Base Rate: 5,000

Connect Timeout: 10,000

Read Timeout: 60,000

Connection Concurrency

☐ Limit Concurrent Connections 5

3. Keep the default 5000 (5 sec) or change the **Polling Base Rate** to how often you want the data to refresh. For example, you can change the default and set it to 1000 (1sec).

4. Click **OK** and save your project with **File > Save and Publish**.
   The SQL property bindings that have the Relative rate/mode selected are now polled at a 1 second rate and the Client data is now updated at a 1 sec rate.

Next ...

- add link here

Clients

The web-launched Vision Clients in Ignition are the "runtimes" of the Vision module. You can launch one or more Clients to display the projects you created in the Designer. They run as full applications and feel like a traditional installed Client, without the need to install and manually synchronize projects.

Ignition Clients may look differently on Windows than on Linux. This is because Linux does not have the default Arial font that Windows uses. To have Linux use the same font, it will need to be download. Check out a forum post where a user goes over how to do this: https://forum.inductiveautomation.com/t/java-fonts-and-linux/1112

See also: About Client Updates

In this section ...

Launching Clients

Clients are launched from the Gateway **Home** page, for a specific project. They run as full applications and feel like a traditional installed Client, without the need to install and manually synchronize projects.

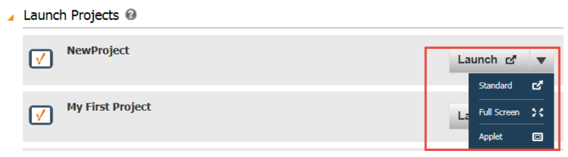
There are three ways to run Clients: **Windowed**, **Full Screen**, and **Applet**. You can choose the mode from the dropdown next to the project name. By just clicking on the project name, the project is launched in the default mode. Certain modes may not be available, depending on project settings.

- Using the **Windowed** Option
  The **Windowed** launch mode is the **Standard** launch method. The Client will be web-launched using Java Web-Start and will have its own window. In this mode, the Client runs as a full, independent application. After being launched, the browser can be closed and the project can be launched from a shortcut on the desktop.

- Using the **Full Screen** Option
  The **Full Screen** launch mode is similar to the Windowed mode, and will also use web-launching to run the Client as a full, independent application. In this mode, however, the Client will occupy the full screen, and will not have a title bar. This mode is ideal for touch-screen display panels, and other
  displays where the Ignition project is the sole focus on the screen.

- Using the **Applet** Option
  Selecting **Applet** launch mode will run the Client application as an applet embedded in your web browser. Applet mode is most commonly used to integrate Vision projects into existing web sites, such as in a corporate intranet setting. See for detailed information.

- Using the **Mobile** Option
  If you have the **Mobile** module installed, you can launch projects on your smartphone or tablet as well. All the user has to do to launch a mobile Client is to connect their mobile device to the wireless network and point the web-browser to the Gateway's LAN address. At this point, they'll be presented with a mobile-optimized version of the Ignition Gateway homepage, where they can select a project to launch. Note that projects must have at least one window defined and be enabled for mobile launching in order to show up in this list. After selecting a project and logging in, they can use the project like a normal project. To access the mobile project context menu, press and hold on your touch-sensitive device. A circular menu will appear allowing you to switch between pointer and pan/zoom mode, as well as options for logging out and entering text input.



To Continue on the step-by-step simple workflow...

After you **Launched your Client**, as described on this page, click the link below for Step 7 to continue.

Step 1. **Launch the Gateway**

Step 2. **Connect to a Device**

Step 3. **Connect to a Database**

Step 4. **Launch the Designer**

Step 5. **Create Tags**

Step 6. **Launch Clients**

Step 7. **Add History to Tags**

Step 8. **Create a Graph and View Graphs in Clients**

Launching a Client

Now that we've created a project, lets launch a Client to see it in action.

1. From the Designer toolbar, select **File > Save and Publish** to save your project.
2. Go back to the Gateway **Home** page.
   Your project will appear in the **Launch Projects** panel with a big **Launch** button to its right.
3. Click on the **Launch** button to start up a Client.
4. Login to the Client using
   Username: **admin**
   Password: **password**

   By default, a new project uses the same user source as the Gateway, so the **`admin/password`** credentials will work.
5. After you logged in, you can see your Client window displaying your project.
6. Go back into the Designer, make a change to the window, and click **File > Save and Publish**.
   Your Client will show a notification saying that there are updates to the project.
7. Click on the notification and the Client will update itself.

You can launch as many Clients as you want! If you've got other computers on the same network as the Gateway computer try launching on them too. Make sure that your Gateway computer doesn't have a Firewall enabled, otherwise, it is allowing traffic on port 8088 - the default port for the Ignition web server.

See also: Launching Clients and Using the Native Client Launchers.

Local Client Fallback

Ignition Clients are fully dependent on being able to communicate with a Gateway. If Gateway communication is lost, the Client suspends operation while it attempts to reconnect with the Gateway. This can be a problem when you need the Client to monitor critical operations on a plant floor.

Ignition provides a local Client fallback mechanism that lets you to use a Gateway running on the local machine. In normal operation, your Client can connect to a central Gateway located somewhere on the network. The central Gateway would be responsible for all data aggregation, such as storing historical data in a database. But if communication to the central Gateway is lost, the Client can automatically retarget to a project that you specify in the local Gateway. This project should contain the minimal realtime information that you need to keep your operation running. Note that in order to use local fallback, **port 6501** must be open on the local machine.

To enable local fallback, do the following steps:

1. Go to **Configuration > Gateway Settings** in the local Gateway.
2. Scroll down to the **Local Client Fallback** section and select the Enable Local Fallback.
3. Select a Fallback Project from the dropdown list.
   Note that the project that you select must be published in the local Gateway and it must have at least one main window.
4. Optionally, you can change the Seconds to Failover setting to a value other than 60 seconds.
   This setting controls the number of seconds to wait before fallback automatically starts. During comm failure, you can also click a button to load the local fallback project immediately.

When local fallback is enabled, the Client attempts to open port 6501 on the local machine. If the port can be opened successfully, the Client reads fallback settings from the local Gateway and shows a **Fallback Project** button on the bottom of the **Gateway Connection Lost** window. You can click this button at any time to load the fallback project, or simply wait for the fallback project to automatically load. You may want to set the local Client to automatically log in to avoid typing in a username and password when the Client loads. This can be set in the Login section of the project's properties.

Testing Local Fallback

Testing local fallback is highly recommended before you start to depend on it in a production setting. The easiest way to test fallback is to simply unplug the network cable to the Client machine, or disable the network card on the machine. If the **Fallback Project** button is not visible on the **Gatew ay Connection Lost** window, check your local Gateway console and verify that the message **Started Fallback Socket on port 6501** is present in the console. Any other error message related to the **Fallback Socket Controller** indicates that some other problem has occurred (most likely the port cannot be reserved) and local fallback is not available to Clients.

Client Retargeting

Client Retargeting is the method by which Clients running a particular project switch to a different project on the fly, even if the other project is hosted on a different Ignition Gateway. Retargeting is a key feature used to build distributed systems. It allows you switch between projects and servers as easily as switching between windows. Using Retargeting, even geographically dispersed projects can be presented as a single cohesive unit.

Reconnect to Central

In many circumstances, the communication loss to the central Gateway is only a temporary event. To minimize the amount of time that you need to run the local Client, you can add some functionality to the Client that allows you to check on the status of the central Gateway. One way to do this is to add a timer script to your local Client. The script calls the `system.util.getGatewayStatus()` function at regular intervals and updates an item such as a Client tag with the Gateway status. Then, you can bind an indicator component to the Client tag and get ongoing visual feedback on the central Gateway status. As soon as you can confirm that the central Gateway is running again, you can call the `system.util.retarget()` function in a button to seamlessly direct the Client back to the central Gateway.

In this section ...

Using the Windowed Option

Clients can be launched with Java Web Start in Windowed mode from the Gateway Webpage or the Designer. Windowed mode creates a Client that appears as a normal windows program.

To launch a Client in Windowed mode

1. Go to the Gateway **Home** page.
2. Click on the **Launch** button for the project you wish to open in the Client.
   This will launch the Client in the default **Windowed** mode.
3. Click **Keep** on the small dialog box that appears on the bottom-right of your window, and then click on the **.jnlp** file to start the Client.
   The login screen for the Client is displayed. You can keep the Client screen as large as the whole screen or as small as you wish.
4. Type in the **Username** and **Password** to see your project.
   Your project is now displayed in the Client.

INDUCTIVE UNIVERSITY

Using the Windowed Option

Watch the Video

Next ...

- Using the Full Screen Option

Using the Full Screen Option

Clients can be launched with Java Web Start in Full Screen mode from the Gateway Webpage or the Designer. Full Screen mode creates a Client that takes up the entire monitor including the Windows Taskbar. Settings and programs other than Ignition are required to block the user from getting to their desktop.

INDUCTIVE UNIVERSITY

Using the Full Screen Option

Watch the Video

To launch a Client in Full Screen mode

1. Go to the Gateway **Home** page.
2. From dropdown of the **Launch** button, select the **Full Screen** mode to launch the project you wish to open in the Client.
3. Click **Keep** on the small dialog box that appears on the bottom-left of your window, then click on the **.jnlp** file to start the Client.
   The login screen for the Client is displayed covering the full size of your screen. In the **Full Screen** mode, you can not make the screen smaller as you could do with the **Windowed** mode.
4. Type in the **Username** and **Password** to see your project.
   Your project is now displayed in the Client.

Next ...

- Using the Applet Option

Using the Applet Option

Clients can be launched in a web browser with Java in Applet mode from the Gateway Webpage or the Designer.

The google chrome web browser platform will no longer support Java applets starting in 2015. However, Java applets will still be available on other web browsers. Also, your browser may block a Java Applet from launching, therefore you may need to unblock access to Java Applets.

To launch a Client in Applet mode

1. Go to the Gateway **Home** page.
2. Select the **Applet** mode from the drop down option associated with the launch button associated with your project.
3. The project should launch in its own window on your web browser.

INDUCTIVE UNIVERSITY

Using the Applet Option

Watch the Video

Next ...

- Using the Mobile Option

Using the Mobile Option

Clients can be launched on a mobile device using the Mobile Module. They are opened behind the scenes on the server and sent using the mobile device's web browser, not as a native App. A desktop environment (that is, X Window) is required for Linux servers.

To launch a Client on a Mobile device

1. Go to the Gateway **Home** page.
2. From the **Mobile Launch** section, click on the `http://localhost:8088/main/system/mobile` link.
   The Ignition Mobile launch page is displayed showing a list of all your projects.
3. Click on your project name.
   This downloads and displays your project.

Using the Mobile Option

Watch the Video

Next ...

- Using the Direct Launch Links

Using the Direct Launch Links

Clients can be opened with the Direct Launch Links in addition to using the Gateway Home page. These links can be used in any browser on the network and will launch the **.jnlp** files without a desktop shortcut.

INDUCTIVE
UNIVERSITY

Using the Direct Launch Links

Watch the Video

To open a Client with the Direct Launch Link

1. Go to the **Configure** section of the Gateway.
2. Go the the **Configuration > Projects** page.
   You will see a list of your projects on the **Projects** page.
3. Click the **view** link to the far right of the row for the project you want.
   The **Project details for 'your_project_name'** page is displayed. You can see the **Published** and **Staging** launch links.
4. Right-click on **Published/Windowed** and choose **Copy link address**.
5. Go to a browser, paste the link, and click enter to launch the Client.

Next ...

- Using the Native Client Launchers

Using the Native Client Launchers

The Ignition Gateway contains downloadable native executables that can launch Clients directly without invoking Java Web Start. Using the native executables you can set up Clients in ways that are impossible to achieve with Java Web Start, such as being able to launch Clients automatically as part of a machine startup script.

The Native Client Launcher automatically searches the network for existing Ignition Gateways and lists existing projects. It is a more robust shortcut than the default desktop shortcuts created by Java.

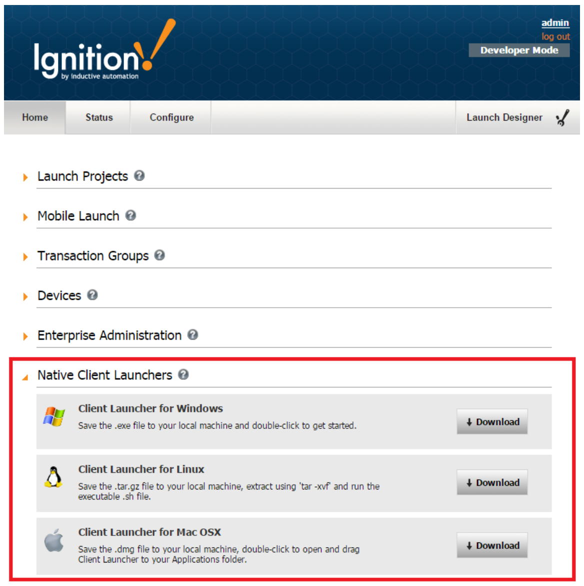
Note: The Native Client Launchers require Java to be installed on the Client machine.

INDUCTIVE UNIVERSITY

Using the Native Client Launchers

Watch the Video

Download and Installation



Client Launcher for Windows
Save the .exe file to your local machine and double-click to get started.

Client Launcher for Linux
Save the .tar.gz file to your local machine, extract using 'tar -xvf' and run the executable .sh file.

Client Launcher for Mac OSX
Save the .dmg file to your local machine, double-click to open and drag Client Launcher to your Applications folder.

Native client launchers are available on the Gateway home page for Windows, Linux and Mac OSX. Click on the Download link to download the native launcher that you need for your operating system. You can hide the native client launcher section on the gateway home page by disabling the section under the Gateway Settings page -> Homepage Config tab.

Windows Installation

Move clientlauncher.exe to a convenient location after it has downloaded to your local machine. Double-client clientlauncher.exe to start.

Linux Installation

Open the downloaded .tar.gz file using an archive manager, or use tar -xvf on the command line. Run clientlauncher.sh (the file is already executable, so you can simply double-click on the file).

> Ubuntu 14.04 users! Starting in 14.04, Ubuntu has turned off the list of execution options that pops up after double-clicking on a .sh file (display the file, execute directly, or execute in a shell). This means that double-clicking clientlauncher.sh will open the text editor and display a bunch of garbage. To re-enable the list of execution options, open a File Manager. Navigate to the Edit menu -> Preferences and click on the Behavior tab. Change the Executable Text Files setting from "View executable text files when they are opened" to "Ask each time".

Mac OSX Installation

Double-click on the ClientLauncher.dmg file. Drag the Client Launcher .app into the Applications folder. Double-click the Client Launcher.app to get started.

Java Requirements

The client launchers require a installed Java Runtime Environment (JRE) with the same minimum JRE version as the Ignition client. However, if you are using a Windows machine, and you are in a situation where you cannot install a JRE, you can still use the client launchers. You will need to package the JRE installation folder on another machine as a .zip file, and extract the zip to a /jre folder next to clientlauncher.exe. The client launcher will use your JRE both for its own execution and for the launched client.

Configuration

When the client launcher opens for the first time, you must select a Gateway on the Gateway Configuration screen. If your Gateways have multicast enabled, and multicast transmission is allowed on your network, you can see a list of Gateways under the Available Gateways tab. Select a Gateway from the list and click the Select Gateway button. The client launcher will attempt to connect to the Gateway that you selected.

If you do not see your Gateway, you can still manually enter Gateway address under the Manually Input Gateway tab. The format is "**host:port**":

```
//Examples
127.0.0.1:8088

//The following will connect to a Gateway running on the local machine
localhost:8088
```

After you have entered your address, click the Apply button. The client launcher will attempt to connect to the Gateway that you selected.

After you have selected a Gateway, the client launcher will attempt to contact the Gateway and will update the Status field at the top of the screen. If there was an error while attempting to contact the Gateway, you can hold your mouse over the Status field to see what the error was. You can also check <user home folder>/.ignition/clientlauncher-data/clientlauncher.log for error information. Keep in mind that you cannot use a client launcher to connect to older Gateways. The minimum Gateway versions that can be used by client launchers are 7.5.11 for the 7.5 Ignition platform and 7.6.4 for the 7.6 Ignition platform. Any 7.7 or later Ignition platform can be used with the client launchers.

To use the Native Client Launchers

1. Go to the **Home** page on the Gateway.

2. Scroll to the bottom of the page to see the **Native Client Launchers** section.

3. Click on the **Download** button to download the native launcher that you need for your operating system.

   • For **Windows**, you will see a **clientlauncher.exe** file.
   • For **Linux**, open the downloaded .tar.gz file using an archive manager, or use tar -xvf on the command line. You will see a **clientlauncer.sh** file.
   • For **Mac OS X**, double-click the **clientlauncher.dmg** file, drag the Client Launcher .app into the Applications folder.

4. Click on the **.exe**, **.sh**, or **.app** file (depending on your OS) to open the page that shows URL for available Gateways.

5. Choose the Gateway you want, click the **SELECT Gateway** button, and click **Save & Close**.
   On the window that is displayed you can launch your project.

   If your Gateways have multicast enabled (**Multicast Enable** check box is selected on the **Gateway Settings** page), and multicast transmission is allowed on your network, you can see a list of Gateways under the **Available Gateways** tab.

6. Select a Gateway from the list and click the **SELECT Gateway** button.
   The Client launcher will attempt to connect to the Gateway you selected.

7. If you don't see your Gateway, you can manually enter the Gateway address under the **Manually Input Gateway** tab, the format is **host:port**.

   For example, you can input `127.0.0.1:8088` or `localhost:8088` to connect to a Gateway running on your local machine. After you have entered your address, click **Apply**.
   The Client launcher will attempt to connect to the Gateway you selected.
   After you have selected a Gateway, the Client launcher will attempt to contact the Gateway and will update the Status field at the top of the screen. If there was an error while attempting to contact the Gateway, you can hold your mouse over the **Status** field to see what the error was. You can also check `<user home folder>/.ignition/Clientlauncher-data/Clientlauncher.log` for error information. Keep in mind that you cannot use a Client launcher to connect to older Gateways. The minimum Gateway versions that can be used by Client launchers are 7.5.11 for the 7.5 Ignition platform and 7.6.4 for the 7.6 Ignition platform.

   You can hide the **Native Client Launcher** section on the Gateway **Home** page by disabling it in the **Gateway Settings > Homepage Config** tab.

Managing Multiple Gateways

It is possible to configure the client launcher to always start on the Gateway Configuration screen. This allows you to connect to any number of Gateways using the same launcher. To enable this configuration, open the launch.xml file (details on this file are in the Custom Launch Settings section below). Set the initconnect property to false and save the file.

Gateway Favorites

If you are managing multiple Gateways, it may be helpful to mark some Gateways as favorites. These Gateways are always listed first when accessing the Gateway Configuration screen. You can also filter out non-favorite Gateways by checking the Favorites Only checkbox on the Gateway Configuration screen. By setting the initconnect property to false (as documented above in the Managing Multiple Gateways section), you can force the client launcher to always display your favorite Gateways list any time you start the launcher.

Launching Projects

After the Gateway has been configured, you can launch a client project or the Designer from the Projects screen. After a project has been launched, a desktop shortcut is created. You can use the desktop shortcut to directly launch the project from the client launcher without needing to select the project again from the Projects screen. A client project will not be visible for launch unless all the conditions below are satisfied:

- The project has at least one main window
- The project is not disabled in the Gateway
- The project is not hidden from launch via the project properties
- The project's configured window launch mode matches the client launcher window mode. For example, if a project's default launch mode is Windowed mode and the "Full Screen" button is not enabled in the project's properties, this means that the project cannot be launched when the client launcher is running in fullscreen mode.

If a project is no longer available in the Gateway, launching the desktop shortcut for the old project will cause the client launcher to display the Projects screen along with an error message at the top of the screen.

Temporary Configuration

The client launcher is typically used to open different projects residing on a single configured Gateway. However, you can also create temporary connections to other Gateways by checking the "Temporary Connection" checkbox on the Gateway Configuration screen. This prevents the selected Gateway from being saved in launch.xml, which may be helpful if you have already configured a default Gateway and you do not want to lose its connection settings. After the connection has been established and saved, the Projects screen will be shown for the temporary Gateway. You can then launch a project or the Designer as you normally would. If the "Create desktop icon" checkbox is checked for the launched project, the temporary Gateway network address is saved in the desktop shortcut.

Redundancy

The client launcher can take advantage of a redundant Gateway setup. Whenever a connection is established with a master Gateway, the backup Gateway IP address is automatically stored in the client launcher configuration file. If the master Gateway cannot be contacted the next time the client launcher is run, an attempt is made to contact the backup Gateway. If the backup cannot be contacted, the client launcher switches between contacting the primary Gateway and the backup Gateway until one responds or the user closes the launcher.

Custom Launch Settings

Client launcher configuration settings, log files and temp files are stored by default in **<user home folder>/.ignition/clientlauncher-data/**. The following settings can be configured in launch.xml:

Setting Name	Setting Description	Example Setting
gateway.addr	The gateway that you have configured which contains your launchable projects	10.20.7.122:8088
gateway.backup.addr	Automatically set by client launcher when using a redundant setup	10.20.7.122:8088
timeoutsecs	The maximum number of seconds allow for any gateway communication. Any communication that exceeds this amount will cause the client launcher to abort the communication and try again if configured.	30
retries	How many times to attempt to contact a gateway again if an error occurred during communication. The default setting of -1 cause the client launcher to try again forever until the client launcher is manually closed. A setting of 0 will cause the client launcher to abort communication after the first failure and display the Gateway Configuration screen. A setting of 1 or more will cause the client launcher to make X more attempts before aborting and showing the Gateway Configuration screen.	-1,0,1
windowmode	Controls whether or not to launch the client launcher in fullscreen mode. The fullscreen setting will cause the client launcher to take over the entire screen. When launching a client that also has fullscreen mode configured, the	window, fullscreen

	fullscreen client will replace the fullscreen client launcher after launch. The default setting is "window", which launches the client launcher in windowed mode.	
screen	The screen index indicates which monitor to use when launching in fullscreen mode.	0,1
show. closebutton	When this is set to false, all close buttons on the client launcher are hidden. Use this setting along with the fullscreen mode to set up a dedicated touch-screen terminal.	true,false
multicast. addr	This setting must match the gateway's multicast address setting in order to see a list of available gateways on the Gateway Configurations screen.	231.1.1.1
multicast. receive. port	This setting must match the gateway's multicast receive port in order to see a list of available gateways on the Gateway Configurations screen.	4446
initconnect	When this is set to false, the client launcher will not attempt to contact the configured Gateway, but instead will immediately open the Gateway Configuration screen. This allows you to choose different Gateways using a single client launcher.	true,false
tempconn ection	When this is set to true, the Temporary Connection checkbox on the Gateway Configuration screen will be checked when the client launcher is started. By setting this property to true and setting the initconnect property to false, you can use the client launcher to access any number of Gateways without saving the selected Gateway address in launch.xml.	true,false
jvm-args	Allows JVM-specific settings to be set on the client that is launched, such as MaxPermSize memory allocation.	-XX: MaxPermSi ze=512m
other-args	See the "Setting client tags" section below	

Startup Parameters

The settings below can also be passed to a client launcher executable as startup parameters. Note that many of the settings overlap with the custom launch settings above, and if present, they will override the settings in launch.xml.

Setting Name	Setting Description	Example Settings
gateway. addr	The gateway that you have configured which contains your launchable projects	10.20.7.122 :8088
gateway. backup. addr	Automatically set by client launcher when using a redundant setup	10.20.7.122 :8088
timeoutse cs	The maximum number of seconds allow for any gateway communication. Any communication that exceeds this amount will cause the client launcher to abort the communication and try again if configured.	30
retries	How many times to attempt to contact a gateway again if an error occurred during communication. The default setting of -1 cause the client launcher to try again forever until the client launcher is manually closed. A setting of 0 will cause the client launcher to abort communication after the first failure and display the Gateway Configuration screen. A setting of 1 or more will cause the client launcher to make X more attempts before aborting and showing the Gateway Configuration screen.	-1,0,1
windowm ode	Controls whether or not to launch the client launcher in fullscreen mode. The fullscreen setting will cause the client launcher to take over the entire screen. When launching a client that also has fullscreen mode configured, the fullscreen client will replace the fullscreen client launcher after launch. The default setting is "window", which launches the client launcher in windowed mode.	window, fullscreen
screen	The screen index indicates which monitor to use when launching in fullscreen mode.	0,1
show. closebutton	When this is set to false, all close buttons on the client launcher are hidden. Use this setting along with the fullscreen mode to set up a dedicated touch-screen terminal.	true,false
scope	Indicates whether the project being launched is a client (scope C) or a Designer (scope D)	C,D
project	The project name. This field is required if the scope is set to C, indicating that a client is being launched.	myproject
initconnect	When this is set to false, the client launcher will not attempt to contact the configured Gateway, but instead will immediately open the Gateway Configuration screen. This allows you to choose different Gateways using a single client launcher.	true,false
jvm-args	Allows JVM-specific settings to be set on the client that is launched, such as MaxPermSize memory allocation. Note that you need to substitute the '#' character for the '=' character, so that the command line can be parsed properly by the client launcher.	-XX: MaxPermSi ze#512m
		true,false

tempconnection	When this is set to true, the Temporary Connection checkbox on the Gateway Configuration screen will be checked when the client launcher is started. By setting this property to true and setting the initconnect property to false, you can use the client launcher to access any number of Gateways without saving the selected Gateway address in launch.xml.	

Client Startup Examples

```
Windows:
"C:\ClientLauncher\clientlauncher.exe" scope=C project=myproject windowmode=window

Linux:
./clientlauncher.sh scope=C project=myterminal windowmode=fullscreen screen=0 show.closebutton=false
```

For another example of automatically launching clients with the Native Client Launcher, please see the following Knowledge Base Article: Launch Project to Multiple Monitors

Setting client tags

Client tag values in Ignition clients can be set by the client launcher upon startup. There are two ways you can do this. The first way is to add the tag info to the other-args section in launch.xml, as shown below. These settings would take effect for all clients started from the client launcher. The examples below assume that you have two client tags in the root folder of your project, called "LaunchTag" and "LaunchTag2".

Client tags that are set by the client launcher cannot contain spaces in the tag names!

```
<other-args>
<arg>-Djavaws.launchparams=LaunchTag;LaunchTag2</arg>
<arg>-Djavaws.launchparam.LaunchTag=AAA</arg>
<arg>-Djavaws.launchparam.LaunchTag2=ZZZ</arg>
</other-args>
```

The other way is run the client launcher from a command line or a desktop shortcut and add these lines to the end of the command:

```
-Djavaws.launchparams="LaunchTag;LaunchTag2" -Djavaws.launchparam.LaunchTag="A A"-Djavaws.launchparam.
LaunchTag2="B B"
```

Starting the client launcher from a command line in this case would look like the example below (Linux example shown)

```
./clientlauncher.sh scope=C project=myterminal windowmode=window -Djavaws.launchparams="LaunchTag;
LaunchTag2" -Djavaws.launchparam.LaunchTag="A A" -Djavaws.launchparam.LaunchTag2="B B"
```

Other tricks

The client launcher can be pre-configured locally and then deployed to remote client machines via a zip file. Using this capability allows a client launcher to immediately connect to a Gateway on the very first launch.

- First, you need to configure a client launcher on a local machine to point to a Gateway.
- Next, copy the **<user home folder>/.ignition/clientlauncher-data** folder next to the launcher executable.
- Zip everything together and distribute the zip to other machines.
- For Window and Linux machines, the launcher can use the clientlauncher-data/ folder that was placed next to the executable. For OSX machines, you need to copy the clientlauncher-data/ folder to the user's home directory and place the Client Launcher. app wherever convenient.

Next ...

- Setting Up Mobile Clients

Setting Up Mobile Clients

With the Mobile module installed, you can launch your projects on any modern smartphone or tablet. This ability does not require any re-design of your projects – a mobile Client launches the same projects that the Vision Clients launch.

See the Mobile Module for more information.

To set the Mobile Module Settings

There is very little setup involved with the mobile module. Usually, with the exception of the the Server Address setting, you can keep all the the default settings.

1. Go to the **Configure** section of the Gateway.
2. Scroll down and go to **Mobile > Settings** page.

Here you will see all the settings and shown in the following table:

Settings	
Java Path	The path to the Java executable on the Ignition Gateway server machine. The Java 8 JRE is required for the mobile module. The default is **java**, which assumes that Java 8 is on the path and can be invoked merely with the **java** keyword.
Client Memory	The max amount of heap space allowed for each mobile client VM that is launched. Mobile clients are virtual clients that are launched on the server. All of the work is done on the server and transmitted to the mobile device so keep in mind that more mobile clients means more memory and CPU consumption on the server.
JVM Options	Command-line JVM options to use when launching mobile client VM's. Multiple options are separated with spaces. This option is made available mostly for troubleshooting by technical support staff, but if you are familiar with java and comfortable with command-line arguments, you can specify ones you may find useful.
Environment Variables	The environment variables to pass to the mobile client VM process. Use the NAME=VALUE format, one per line.
Idle VMs	The number of client VMs to startup and wait for incoming mobile connections. These will start when the Gateway is started and sit idle until a mobile connection is made. This should be left at the default value of 0 unless it is taking a long time to launch a mobile client. It's important to note that the VMs that are sitting idle are not connected to a project so it will still take time to load the selected project.
Server Address	This is the address to use when launching from the QR code on the launch page. This is a setting that often causes confusion. Initially this value is blank by default which results in the QR code pointing to the address of http://localhost:8088/. Since Ignition is not running on your mobile device, this address will not actually launch the mobile homepage. This should be set to the Ignition server address that can be reached by the mobile device.
Networking	
Callback Port	The port that the mobile Client VMs use to communicate to the Gateway on. The default is 45900, but if this port is already in use, you need to change this to an available port.
Callback Interface	The interface that mobile client VMs should use to communicate back to the Gateway on. The default, localhost, makes use of the loopback adapter, however if this host doesn't have a loopback adapter or if there are two network cards, set this to the IP address of the NIC that should be used for local loopback.
AJAX Timeout	The max time, in milliseconds, that each request has to complete. The default is 10,000 (10sec).
Advanced Properties **The Mobile Module also has an option to allow VNC connections. This allows certain thin clients that do not support the Java Runtime Environment and also do not have an HTML 5 compatible browser to launch Ignition clients. The settings listed under the advanced properties section all have to do with configuring the VNC connection.**	
VNC Settings	

Enable VNC	Allows direct thin-client connection over VNC (also known as, RFB) protocol.
VNC Port	The port used for the VNC connections.
Project Name	The Mobile module only allows one of the projects on the Ignition Gateway to be viewed through VNC so you have to specify that project here. Unlike the normal mobile launch screen that allows you to choose a project, the project that you specify in this setting will be automatically launched when you connect via a VNC viewer application.
Project Width	The width of the project when it's launched.
Project Height	The height of the project when it is launched.

In this section ...

About Mobile

Mobile Clients require the Mobile module. Mobile Clients are headless Clients launched behind the scenes on the server and the Client is shown on the mobile device using its web browser instead of a native App. HTML 5 allows users to interact with the browser as if it was a normal Client.

See also: Mobile Module

About Mobile

Watch the Video

To launch the Mobile Client

1. Go to the Gateway **Home** page.
2. In the **Mobile Launch** section, click on http://ipaddress:8088/main/system/mobile.
   The Client launches on your mobile device.
3. Choose the project you want to open, then login with your username and password.
   The Project is displayed. If there is only one project in the mobile project list, the list will be skipped and you will be sent directly to the login screen.
4. To check how many Clients are launched, go to the **Status** page of the Gateway and see the **Clients** box which shows how many are open.

Launch Mobile Client without viewing the mobile project selection page

It is possible to bypass the mobile project selection page by adding the project name to the URL: http://ipaddress:8088/main/system/mobile?project=projectName

Mobile Module Tags

Using the Mobile Module adds additional (client scoped) tags to the System Tags folder under **System -> Mobile**. This adds Remote Address and User Agent values. If location is enabled in the mobile browser, the Location tags will show Latitude, Longitude, etc.

Next ...

- Setting Mobile Size and Fit to Device

Setting Mobile Size and Fit to Device

Mobile Clients can be set to specific sizes or told to fit themselves to the device.

To set the Mobile size and Fit to device

In the Designer, go to the **Project > Properties**, and then to **Mobile > General**.

You can see that the **Fit to Device** is selected by default. **Fit to Device** will resize itself when the device is rotated. You can also choose **Custom** and specify any size you wish.

INDUCTIVE UNIVERSITY

Setting Mobile Size and Fit to Device

Watch the Video

Next ...

- Setting Java Path

Setting Java Path

The mobile clients can either use the same Java version as the Ignition server or it might be necessary to set a specific path to the Java version that you want the mobile clients to access. A likely scenario for this is if the Ignition server is upgraded to run on Java 9 but the underlying technology that the Mobile client uses to run can be limited to Java 8 pending technological upgrades in the future. Regardless, setting the Java path will result in more control over which version of Java runs the mobile clients. The settings take either the system path such as **Java** or a full path to the **java.exe** file.

To set the Java Path

1. Go to the **Configure** section of the Gateway.
2. Scroll down, go to **Mobile > Settings** page.
   See the **Java Path** setting with **java** as the default path to the Java executable.
3. Enter the path in **Java Path**, for example, for Java 8 JRE you enter,
   C:\Program Files\Java\jre8\bin\java.exe

Setting Java Path

Watch the Video

Next ...

- Setting Max Mobile Clients

Setting Max Mobile Clients

You can set the max number of concurrent mobile Clients in the Gateway **Home** page. This is a safety measure, because each Client consumes resources on the Gateway. Limiting the number of Clients to be launched prevents overloading the Gateway server.

To set the max number of Mobile Clients

1. Got to the **Configure** section of the Gateway.
2. Scroll down, go to **Mobile > Settings** page.
   You will see the **Max Clients** setting which is set to 15 by default.
   Max Clients is the max number of concurrent mobile clients. This number is set as a safety measure, because each client consumes resources on the Gateway, and limiting clients prevents overloading the Gateway server.

Setting Max Mobile Clients

Watch the Video

Next ...

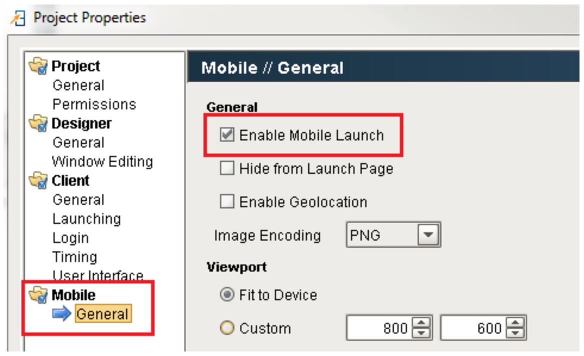
- add link here

Mobile Project Launch Links

Mobile project shortcuts can easily be added to the home screen of a mobile device. The links can point to either the project listing page, or directly to a single project.

Example - Adding a shortcut to the home screen of a mobile device

The following requires that the mobile device has **network access** to the Ignition Gateway. Additionally, at least a single project exists on the Gateway, and the project must be enabled for mobile launch. This property can be found under Project Properties:



General

- [✓] Enable Mobile Launch
- [ ] Hide from Launch Page
- [ ] Enable Geolocation

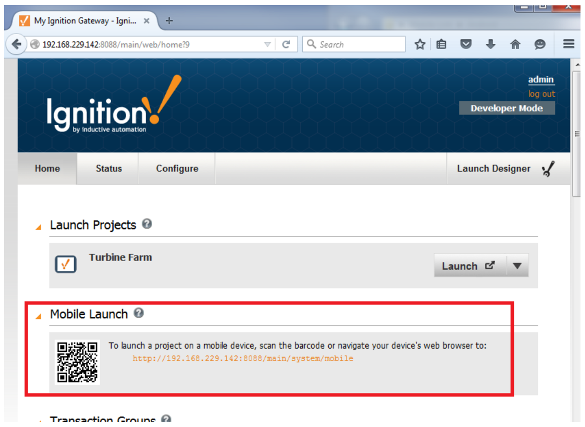
Image Encoding: PNG

Viewport

- [●] Fit to Device
- [○] Custom 800 600

Step 1 - Launch the project on the mobile device

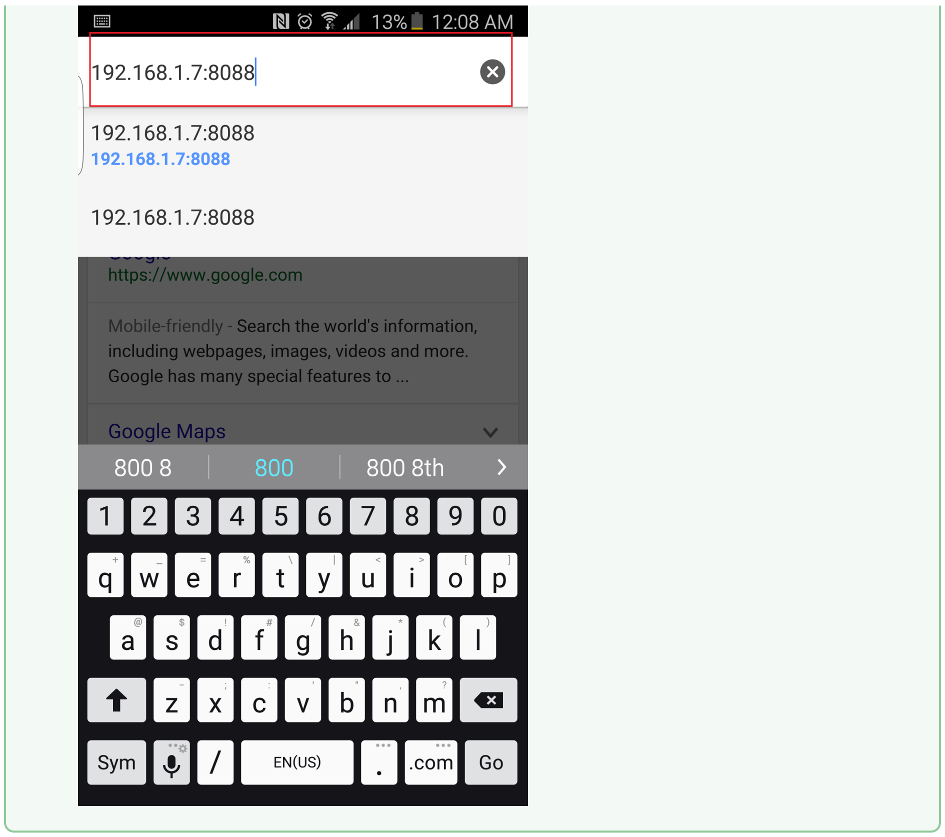
The Home tab of the Gateway Webpage contains a Mobile Launch link.



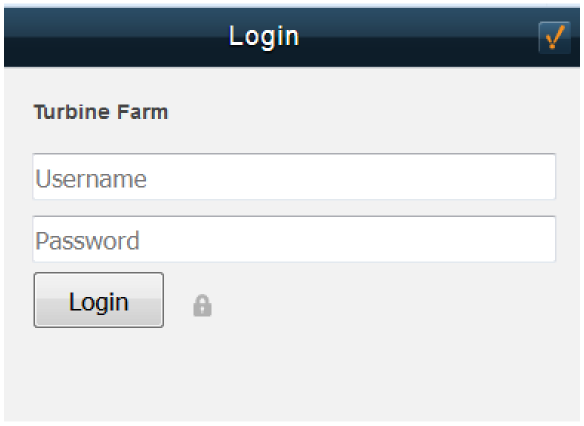
To launch a project on a mobile device, scan the barcode or navigate your device's web browser to:
http://192.168.229.142:8088/main/system/mobile

Launch a mobile project on the device by either using a barcode scanner app, or type the launch link into a browser on the device.

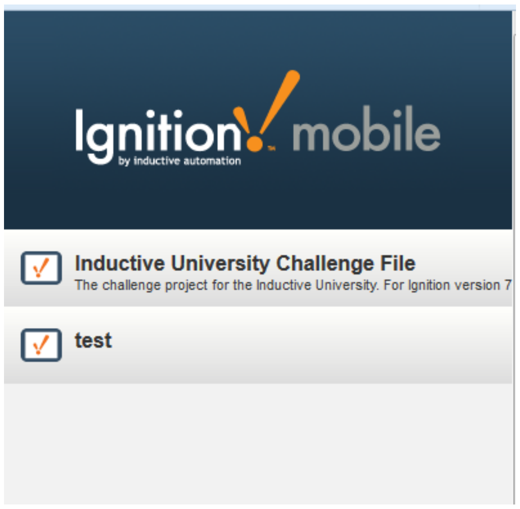
Since the mobile device has network access to the Ignition Gateway, the browser on the device can navigate to the Gateway Webpage directly. This way the project can be launched by clicking on the link from the device.


192.168.1.7:8088
192.168.1.7:8088
192.168.1.7:8088
192.168.1.7:8088
https://www.google.com
Mobile-friendly - Search the world's information, including webpages, images, videos and more. Google has many special features to ...
Google Maps
800 8
800
800 8th
Sym
EN(US)
.com
Go

If only a single mobile-enabled project exists on the Gateway, then a login page will appear.



If multiple mobile-enabled projects are present, a project listing page will appear.



Inductive University Challenge File
The challenge project for the Inductive University. For Ignition version 7

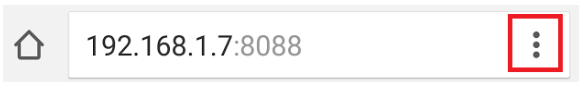
test

Step 2 - Add a shortcut to the device's home screen

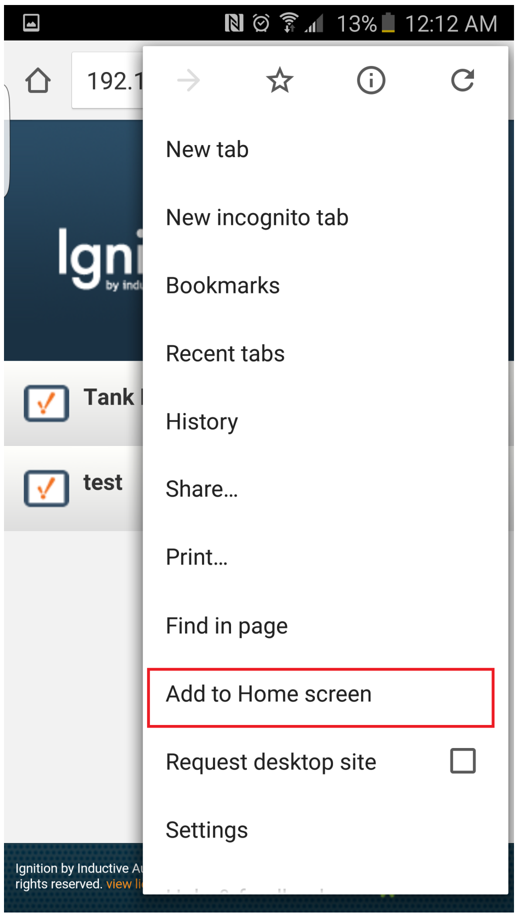
Select a device:

> The following instructions assumes Chrome will be used. If using a different browser, please see the browser's documentation for adding a link to the Home screen.

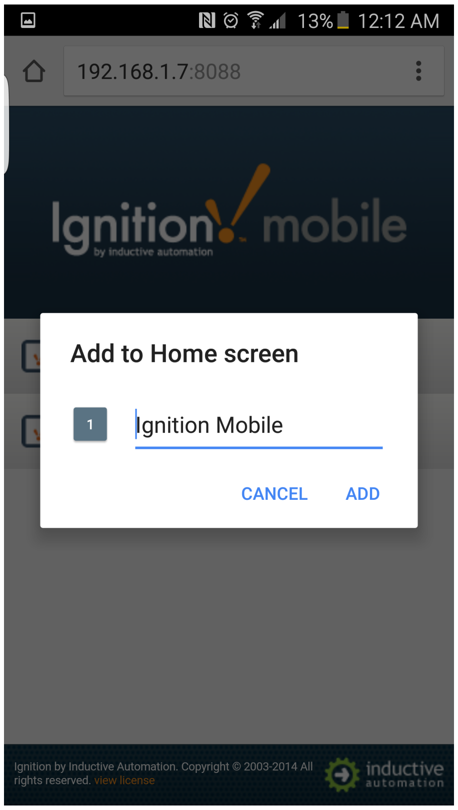
Next to the page's URL, press the More icon



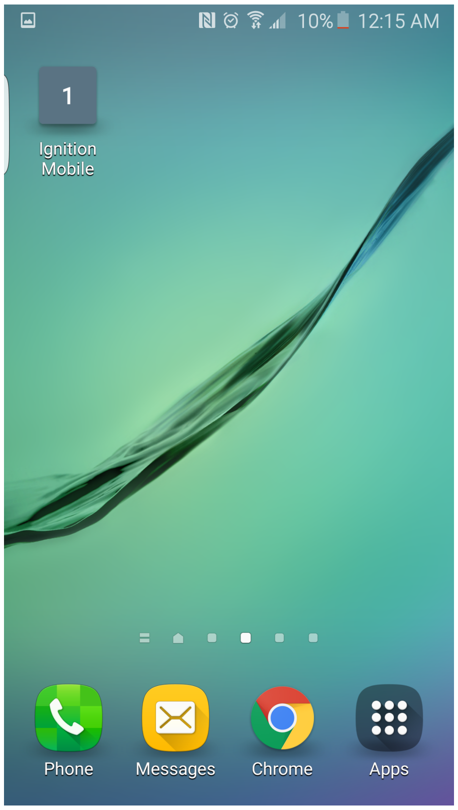
A list of options will appear. Press the 'Add to Home screen' option



Name the link that will appear on the Home screen, and press ADD.

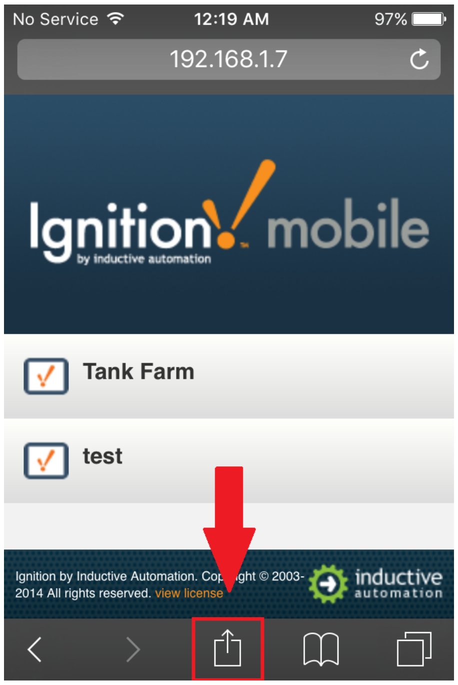


The link will now be available on the Home screen.

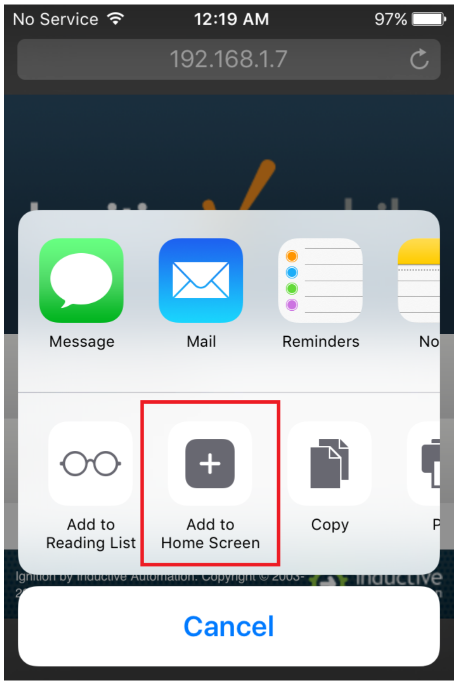


The following instructions assumes Safari will be used. If using a different browser, please see the browser's documentation for adding a link to the Home screen.

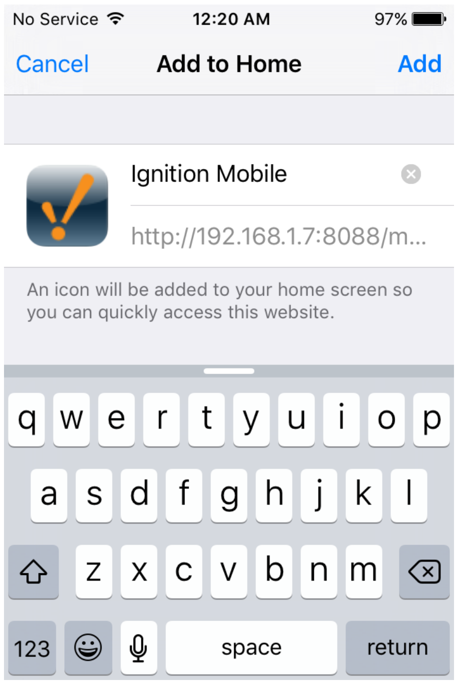
At the bottom of the browser, press the Share button.



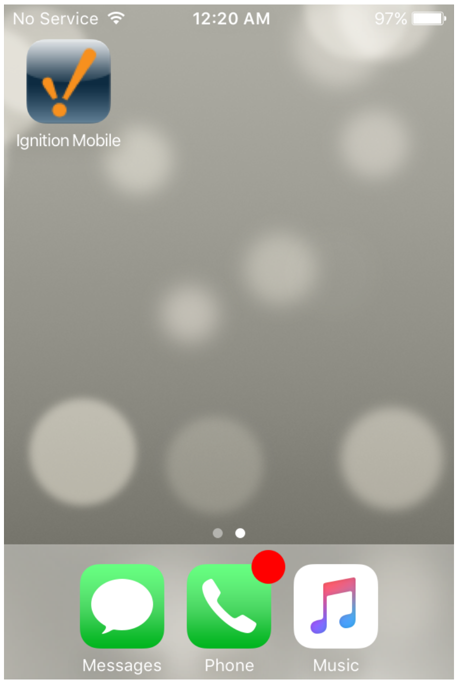
Scroll to the right until the 'Add to Home Screen' icon appears, and press it.



Name the link that will appear on the Home Screen, and press 'Add' when finished.


No Service
12:20 AM
97%
Cancel
Add to Home
Add
Ignition Mobile
http://192.168.1.7:8088/m...
An icon will be added to your home screen so you can quickly access this website.

The link will appear on the Home Screen.



Next ...

- add link here

Database Connections

Providing access to relational databases is at the heart of the Ignition platform. Ignition can connect to any SQL database that has a JDBC driver, though depending on the database's capabilities, some features may not be available.

By connecting Ignition to one or more SQL databases, you can query existing data, update data, store historical information, and more.

On this page...

Why use Databases?

Ignition can perform many tasks without the use of a database. For instance, the Vision and OPC-UA modules let you create powerful HMI status and control screens ,or Tags can be used to generate alarms that can be sent over email. However, a tightly integrated database access is a key feature that makes Ignition stand out from its competitors.

Modern relational databases offer amazing storage and querying capabilities with great performance at a price that is incomparable to older legacy historians. While it is true that historians still have a place in the industry, for most applications relational SQL databases not only suffice, but offer much more than what was previously available. Using SQL, you can store and track production information with ease. However, you can also correlate that data to who was on shift, previous runs, downtime, inventory levels and more, naturally and easily. Make the data available to more people using the Vision module's web-launch clients, or integrate the data directly into your company's internal or external website. SQL databases are at the heart of the web and modern corporate IT systems, and now thanks to Ignition, the plant floor as well.

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, requires 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, detect trends, and more. 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.

Database-driven Tags

You can use a SQL database as your Tags repository. Any other Ignition system with access to the database will be able to share and contribute tags, allowing you to create highly integrated distributed systems. For example, multiple plant sites can use Tags to report current status over a secure network connection to a central corporate headquarters.

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.

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 (sorted by our current user install base)	
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, full, and express editions. Ignition can connect to 2000, but has not been fully tested.
Oracle	10g, 11g, full, and express.
PostgreSQL	8.0+
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.

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.

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.

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 outdated, 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.

Adding a JDBC Driver

To add a new JDBC driver 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. Find the orange arrow and click on the **Create new JDBC Driver...** link.
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 core Java JAR file that contains the driver, as well as any other required JARs.
5. Use the default settings for the following properties:

Driver Defaults and Instructions	
Driver	Is the brand of database. This is used for optimizations in the Gateway, if in doubt, select GENERIC.

Type	
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.

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 orange 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.

Monitoring Connection Status

The state or status of a database can be monitored from the **Status** section of the Gateway, under **Status>Database Connections**. The 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.

In this section ...

Installing Databases

It is important to note that Ignition does not install any databases. You need to choose the database solution that is best for you and then install it. You can install it either on the server that Ignition resides on or a server that can be reached over the network from the Ignition server.

In this section ...

Installing MySQL

INDUCTIVE UNIVERSITY

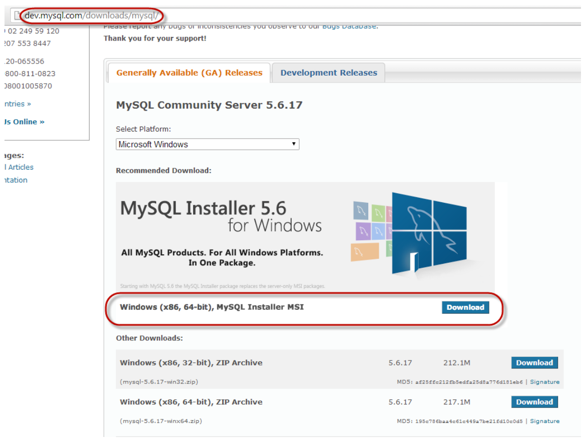
Installing MySQL

Watch the Video

To install the MySQL server and workbench

1. Go to the MySQL website at http://dev.mysql.com/downloads/mysql/

2. Scroll-down to **Windows (x86,64-bit), MySQL Installer MSI**



dev.mysql.com/downloads/mysql/

02 249 59 120
207 553 8447
20-065556
800-811-0823
08001005870
ntries »
Js Online »
ages:
l Articles
ntation

Please report any bugs or inconsistencies you observe to our Bugs Database.
Thank you for your support!

Generally Available (GA) Releases | Development Releases

MySQL Community Server 5.6.17

Select Platform: Microsoft Windows

Recommended Download:

MySQL Installer 5.6 for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the server-only MSI packages.

Windows (x86, 64-bit), MySQL Installer MSI — Download

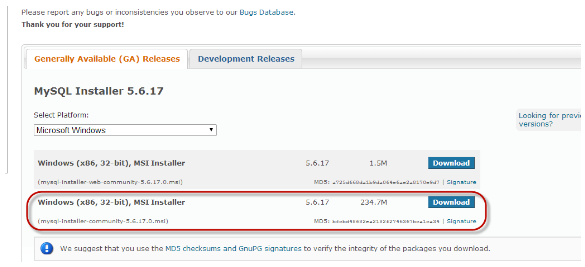
Other Downloads:

Windows (x86, 32-bit), ZIP Archive (mysql-5.6.17-win32.zip)	5.6.17	212.1M	Download MD5: af25ffc212fb5edfa25d8a776d181eb6 | Signature
Windows (x86, 64-bit), ZIP Archive (mysql-5.6.17-winx64.zip)	5.6.17	217.1M	Download MD5: 195c786baa4c61c449a7be21fd10c0d5 | Signature

3. Click on the **Download** button.

4. Scroll-down to **mysql-installer-community-x.x.xx.x.msi**

   Note: MySQL Installer is 32-bit, but will install both 32-bit and 64-bit binaries.



Please report any bugs or inconsistencies you observe to our Bugs Database.
Thank you for your support!

Generally Available (GA) Releases | Development Releases

MySQL Installer 5.6.17

Select Platform: Microsoft Windows

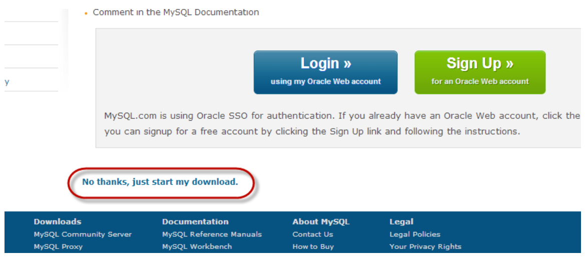
Looking for previ versions?

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-5.6.17.0.msi)	5.6.17	1.5M	Download MD5: a725d668da1b9da064e6ae2a8170e9d7 | Signature
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-5.6.17.0.msi)	5.6.17	234.7M	Download MD5: bfcbd48682ea2182f2746367bca1ca34 | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

5. Click the **Download** button.

6. On the next page, click the **No thanks, just start my download**.



• Comment in the MySQL Documentation

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the you can signup for a free account by clicking the Sign Up link and following the instructions.

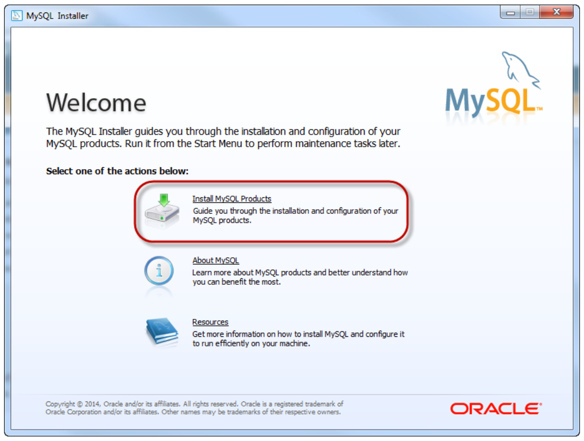
No thanks, just start my download.

   The **.msi** file starts downloading.

7. Click the **.msi** file to run it and to configure the installer.

   The **Welcome** window is displayed.

8. Select the **Install MySQL Products** action.


MySQL Installer

Welcome

The MySQL Installer guides you through the installation and configuration of your MySQL products. Run it from the Start Menu to perform maintenance tasks later.

Select one of the actions below:

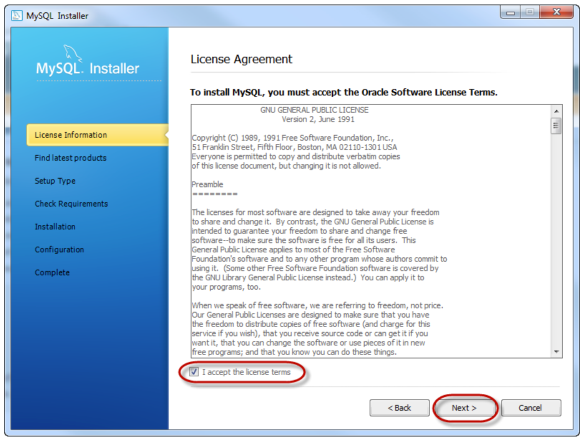
Install MySQL Products
Guide you through the installation and configuration of your MySQL products.

About MySQL
Learn more about MySQL products and better understand how you can benefit the most.

Resources
Get more information on how to install MySQL and configure it to run efficiently on your machine.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

9. On the **License Agreement** page, click the **I accept the license terms** box and click **Next**.


MySQL Installer

MySQL. Installer

License Information
Find latest products
Setup Type
Check Requirements
Installation
Configuration
Complete

License Agreement

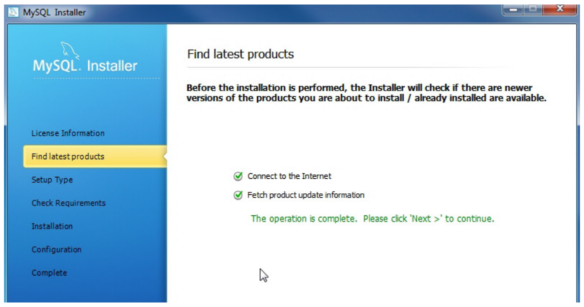
To install MySQL, you must accept the Oracle Software License Terms.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble
========

The licenses for most software are designed to take away your freedom
to share and change it.  By contrast, the GNU General Public License is
intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not price.
Our General Public Licenses are designed to make sure that you have
the freedom to distribute copies of free software (and charge for this
service if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs; and that you know you can do these things.
```

☑ I accept the license terms

< Back | Next > | Cancel

10. On the **Find latest products** page, click **Execute** to check for installer updates from MySQL. If you want to use the version you have, you can select the skip option. When checking is complete, click **Next**.


MySQL Installer

MySQL. Installer

License Information
Find latest products
Setup Type
Check Requirements
Installation
Configuration
Complete

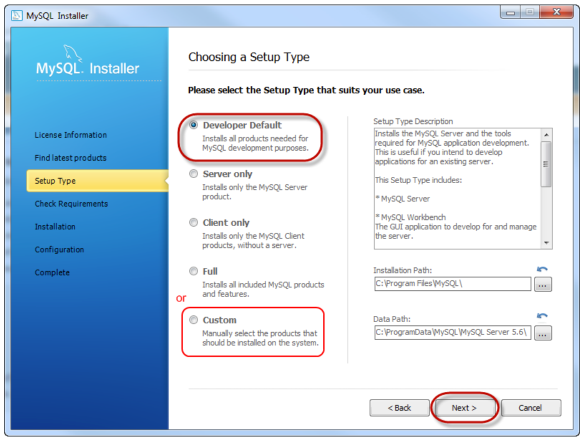
Find latest products

Before the installation is performed, the Installer will check if there are newer versions of the products you are about to install / already installed are available.

✓ Connect to the Internet
✓ Fetch product update information

The operation is complete. Please click 'Next >' to continue.

11. On the **Choosing a Setup Type** page, the simplest option is to select **Developer Default** and click **Next**. The only drawback is that the install will be much larger.


MySQL Installer

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.

or

- **Custom** Manually select the products that should be installed on the system.

Setup Type Description

Installs the MySQL Server and the tools required for MySQL application development. This is useful if you intend to develop applications for an existing server.

This Setup Type includes:

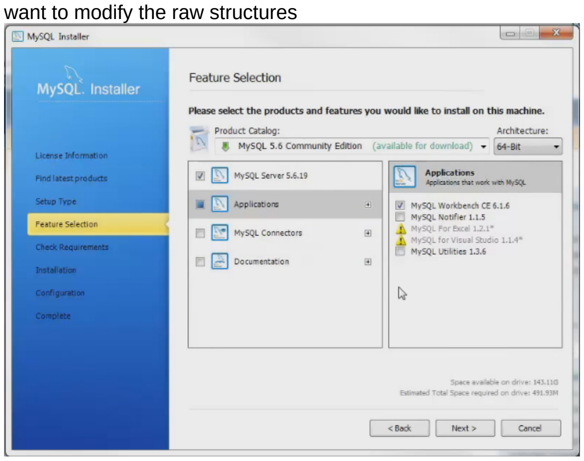
* MySQL Server

* MySQL Workbench
The GUI application to develop for and manage the server.

Installation Path: C:\Program Files\MySQL\

Data Path: C:\ProgramData\MySQL\MySQL Server 5.6\

Optionally, you can select the **Custom** option and remove everything except the **MySQL Server** object. You can include the **MySQL Workbench** if you want, it is nice to have if you ever want to modify the raw structures


MySQL Installer

Feature Selection

Please select the products and features you would like to install on this machine.

Product Catalog: MySQL 5.6 Community Edition (available for download)

Architecture: 64-Bit

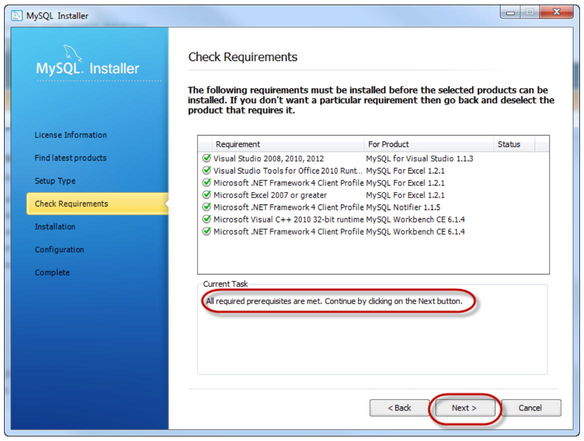
- [✓] MySQL Server 5.6.19
- ▪ Applications
- [ ] MySQL Connectors
- [ ] Documentation

Applications
Applications that work with MySQL

- [✓] MySQL Workbench CE 6.1.6
- [ ] MySQL Notifier 1.1.5
- ⚠ MySQL For Excel 1.2.1*
- ⚠ MySQL for Visual Studio 1.1.4*
- [ ] MySQL Utilities 1.3.6

Space available on drive: 143.11G
Estimated Total Space required on drive: 491.93M

12. On the **Check Requirements** page, click **Execute** to make sure all required prerequisites are met. Follow any instructions to install any missing prerequisites. Click **Next** once all prerequisites are met.


MySQL Installer

Check Requirements

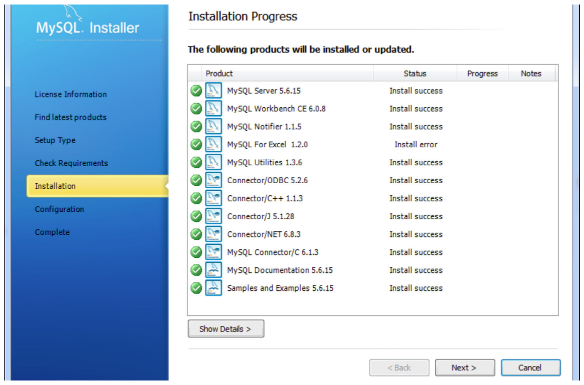
The following requirements must be installed before the selected products can be installed. If you don't want a particular requirement then go back and deselect the product that requires it.

Requirement	For Product	Status
✓ Visual Studio 2008, 2010, 2012	MySQL for Visual Studio 1.1.3	
✓ Visual Studio Tools for Office 2010 Runt...	MySQL For Excel 1.2.1	
✓ Microsoft .NET Framework 4 Client Profile	MySQL For Excel 1.2.1	
✓ Microsoft Excel 2007 or greater	MySQL For Excel 1.2.1	
✓ Microsoft .NET Framework 4 Client Profile	MySQL Notifier 1.1.5	
✓ Microsoft Visual C++ 2010 32-bit runtime	MySQL Workbench CE 6.1.4	
✓ Microsoft .NET Framework 4 Client Profile	MySQL Workbench CE 6.1.4	

Current Task

All required prerequisites are met. Continue by clicking on the Next button.

13. On the **Installation Progress** page, click **Execute** to install all of MySQL products. The **MySQL Server** and the **MySQL Workbench** will both be installed. Once all of the products are installed click **Next**.

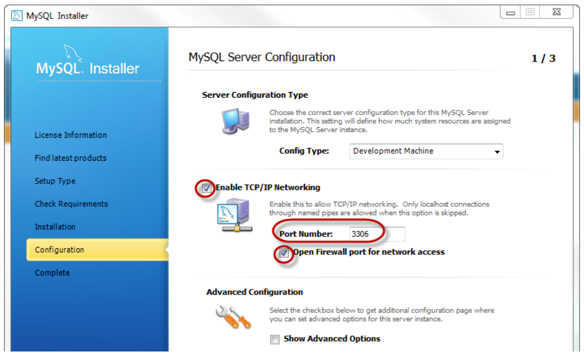

Installation Progress

The following products will be installed or updated.

Product	Status	Progress	Notes
MySQL Server 5.6.15	Install success		
MySQL Workbench CE 6.0.8	Install success		
MySQL Notifier 1.1.5	Install success		
MySQL For Excel 1.2.0	Install error		
MySQL Utilities 1.3.6	Install success		
Connector/ODBC 5.2.6	Install success		
Connector/C++ 1.1.3	Install success		
Connector/J 5.1.28	Install success		
Connector/NET 6.8.3	Install success		
MySQL Connector/C 6.1.3	Install success		
MySQL Documentation 5.6.15	Install success		
Samples and Examples 5.6.15	Install success		

14. On the **Configuration Overview** page, click **Next** to setup the configuration.

15. On the **MySQL Server Configuration** page (1/3), make sure the default **Port Number** is **3306**, keep all the defaults and click **Next**.


MySQL Server Configuration 1 / 3

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 Machine

☑ Enable TCP/IP Networking

Enable this to allow TCP/IP networking. Only localhost connections through named pipes are allowed when this option is skipped.

Port Number: 3306

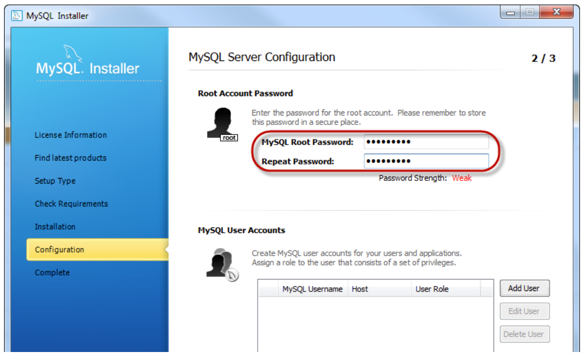
☑ Open Firewall port for network access

Advanced Configuration

Select the checkbox below to get additional configuration page where you can set advanced options for this server instance.

☐ Show Advanced Options

16. On the next **MySQL Server Configuration** page (2/3), choose a password for the root user, type it in, and write it down so you have it for future, then click **Next**.


MySQL Server Configuration 2 / 3

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: •••••••••

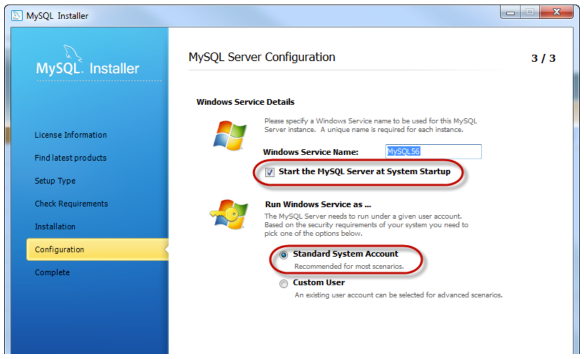
Password Strength: Weak

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 Username	Host	User Role

17. Make sure the following two items are selected by default:
• **Start the MySQL Server at System Startup**
• **Standard System Account**


MySQL Server Configuration 3 / 3

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: MySQL56

☑ Start the MySQL Server at System Startup

Run Windows Service as ...

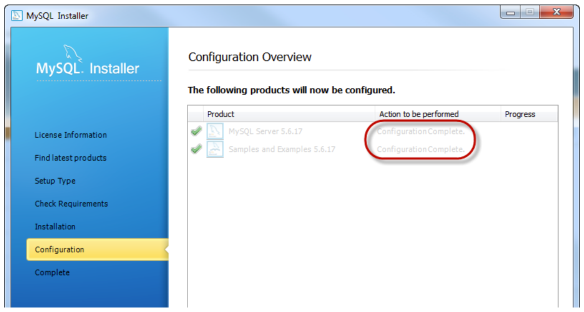
The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

● Standard System Account
Recommended for most scenarios.

○ Custom User
An existing user account can be selected for advanced scenarios.

This installs MySQL as a Service and starts MySQL automatically each time you start your computer.

18. Click **Next**. The **Configuration Overview** page is displayed and shows the MySQL server installation is complete.



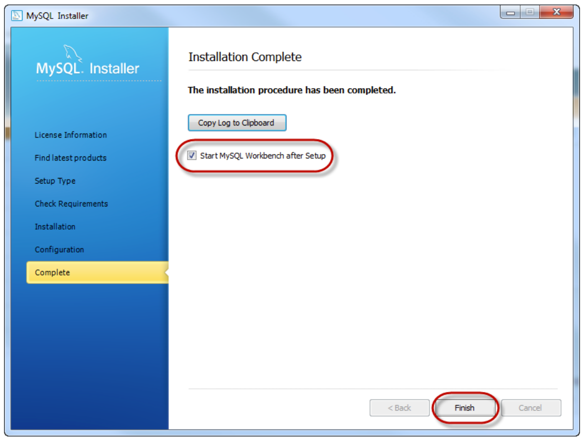
Configuration Overview

The following products will now be configured.

	Product	Action to be performed	Progress
✓	MySQL Server 5.6.17	Configuration Complete.	
✓	Samples and Examples 5.6.17	Configuration Complete.	

19. Click **Next**.

20. The **Installation Complete** window is displayed and shows the check box for **Start MySQL Workbench after Setup** is selected.



Installation Complete

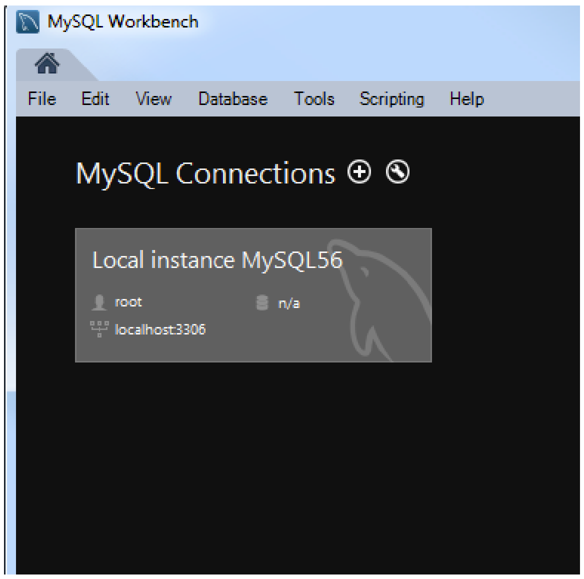
The installation procedure has been completed.

[✓] Start MySQL Workbench after Setup

21. Click **Finish** to close the Wizard and complete the installation.

The **MySQL Workbench** window is displayed. MySQL Workbench lets you to administrate the MySQL server.

22. Click on **Local instance MySQL** to connect to the newly installed MySQL server.



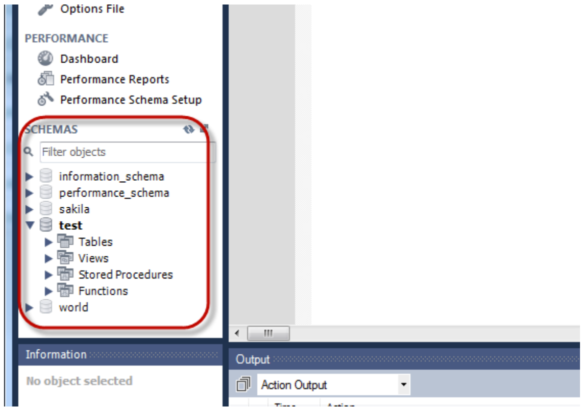
Local instance MySQL56

root n/a

localhost:3306

23. Enter the root **password** you earlier entered into the installer and click **OK**.

24. The **Local instance MySQL** graphical user interface is displayed.



You are now connected and can see, in the Navigator, the default databases. Older versions of MySQL will start with a database named **test** where new ones may not.

25. If you want to create a new schema with a new name (like **test**), right-click in the white background of the Object Browser (next to the rest of the schemas) and select **Create Schema**. Name it and click the **Apply** button in the lower right.

Next ...

- Installing Microsoft SQL Server Express

Installing Microsoft SQL Server Express

You need to download and install both the **SQL Server 2014 Express** and the **SQL Management Studio** before you connect to the database.

INDUCTIVE UNIVERSITY

Installing Microsoft SQL Server Express

Watch the Video

To download SQL Server 2014 Express

1. Go to http://www.microsoft.com/en-us/server-cloud/products/sql-server/
2. Click **Products** and under **SQL Server** click on **SQL Server Editions**.
3. Choose the **Express** tab and click on **Download SQL Server 2014 Express**.
4. Click on the **SQLEXPR_x64_ENU.exe** file to run the executable.
5. On the **SQL Server Installation Center** window, click on the **New SQL Server stand-alone installation or add features to an existing installation** link.
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 selections and click **Next**.
9. The **Instance Configuration** window shows **Named instance: SQL Express**, you can keep it or change it if you like, click **Next**.
10. On the **Server Configuration** window, choose **Automatic** from the dropdown under **Startup Type** for the **SQL Server Browser** service, and click **Next**.
11. On the **Database Engine Configuration** window, for **Authentication Mode** choose **Mixed Mode**, enter a **password**, and click **Next**.
12. On the next few windows, continue clicking **Next** until it shows installation is **Complete**.
    The **SQL Server 2014 Express** is now installed.
    Microsoft SQL Server Express installs with a **tempdb** that wipes its data whenever the computer is restarted.

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.

Next ...

- add link here

Installing Oracle Express

To download and install Oracle Express

1. Go to the Oracle website at http://www.oracle.com
2. Click on **Downloads**, and then under **Database**, click on **Oracle Database 11g Express Edition**.
3. Accept the Licensce agreement and select the download version you are interested in.
   For example, you can download the **Oracle Database Express Edition 111g Relase 2 for Windows x64** database.
   The **OracleXE112_Win64.zip** file is downloaded.
4. Unzip the **.zip** file by right-clicking on the file and selecting **Extract All...**, then click **Extract** on the next window.
   You will now see a DISK1 folder, and inside it is the **setup.exe** executable file.
5. Double-click on **setup.exe** to start the Oracle database installation.
6. On the **Welcome to Install Wizard** window, click **Next**.
7. Accept the terms in the license agreement, and click **Next**.
8. Enter a password, and click **Next**.
   This is the password to the SYSTEM account and you can use any password you wish just note it down.
9. Continue on the next windows, click **Install** to install the database, and click **Finish** once installation is complete.
10. If the completion of the previous step does not automatically take you to the database administrative window in your browser, you can manually open your browser and access the database administrative screen by typing the following URL into your browser: http://localhost:8080.

INDUCTIVE UNIVERSITY

Installing Oracle Express

Watch the Video

Next ...

- add link here

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.

INDUCTIVE UNIVERSITY

Installing PostgreSQL

Watch the Video

Next ...

- add link here

Connecting to Databases

Many of the advanced features of Ignition, such as the Transaction Groups and Tags Historian require a connection to an external database. If you don't have a database, like Microsoft SQL Server, MySQL, or Oracle installed, don't worry - you can come back to this step later.

Now that we've installed your database, lets connect to it. You can find detailed descriptions for many database connections in this User Manual (see Next on this page), however they all include the following three main steps:

1. **Add a Database Connection**
   Once you are in the Gateway **Configure** section of the Gateway's web interface, use the menu on the left to go to the **Databases > Connections** section. On at the **Database Connections** page, click on the **Create new Database Connection...** link at the bottom of the table.

2. **Pick 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.

3. **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 name, which consists of letters, numbers and underscores.
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.
Password	The password to use for the given username.
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 to use when this connection is not available.
Failover Mode	Lets you select how to handle failover and recovery. 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 dictates that the secondary connection will be used only until the primary connection is available again. • **STICKY** continues to use the secondary connection until that connection fails, or until the system is restarted.
Slow Quer	Queries that take longer than this amount of time, in milliseconds, are logged making it easier to find queries that are not performing well.

To Continue on the step-by-step simple workflow...

After you **Connect to a Database**, as described on the pages of this section, click the link below for Step 4 to continue.

Step 1. **Launch the Gateway**

Step 2. **Connect to a Device**

Step 3. **Connect to a Database**

Step 4. **Launch the Designer**

Step 5. **Create Tags**

Step 6. **Launch Clients**

Step 7. **Add History to Tags**

Step 8. **Create a Graph and View Graphs in Clients**

y Log Thre shold	

Advanced Settings
There are a 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

INDUCTIVE UNIVERSITY

Connecting to MySQL

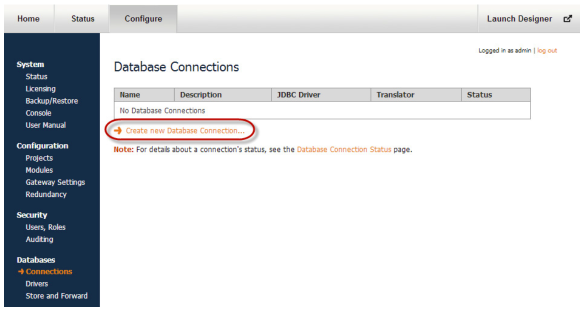
Watch the Video

To connect Ignition to MySQL database

1. From the Gateway, go to the **Configure** section.

2. From the menu on the left, select **Databases > Connections**.
   The Database Connections page is displayed.

3. Look for the orange arrow, click on **Create new Database Connection...**.



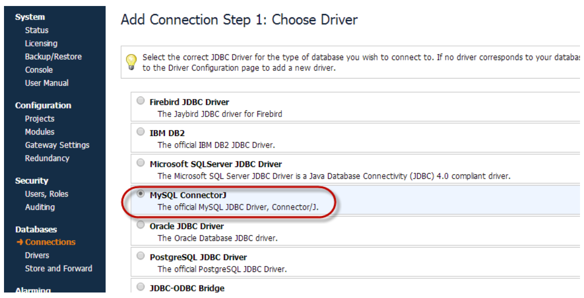
Database Connections

Name	Description	JDBC Driver	Translator	Status
No Database Connections				

Create new Database Connection...

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

4. Select the **MySQL ConnectorJ** driver and click **Next**.



Add Connection Step 1: Choose Driver

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

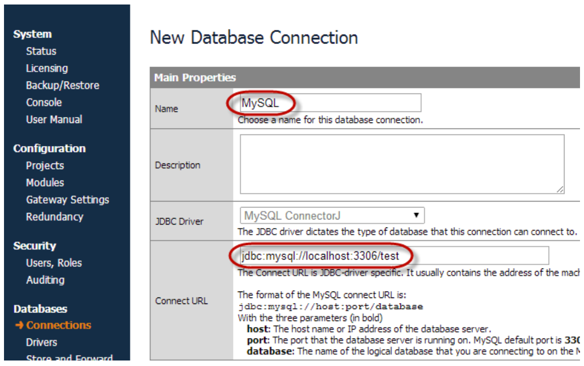
- Firebird JDBC Driver
  The Jaybird JDBC driver for Firebird
- IBM DB2
  The official IBM DB2 JDBC Driver.
- Microsoft SQLServer JDBC Driver
  The Microsoft SQL Server JDBC Driver is a Java Database Connectivity (JDBC) 4.0 compliant driver.
- [✓ MySQL ConnectorJ]
  The official MySQL JDBC Driver, Connector/J.
- Oracle JDBC Driver
  The Oracle Database JDBC driver.
- PostgreSQL JDBC Driver
  The official PostgreSQL JDBC Driver.
- JDBC-ODBC Bridge

   Database connections in Ignition are powered by JDBC drivers. Ignition ships with drivers for Microsoft SQL Server, MySQL, Oracle, and a few others.

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**)



New Database Connection

Main Properties	
Name	MySQL Choose a name for this database connection.
Description	
JDBC Driver	MySQL ConnectorJ The JDBC driver dictates the type of database that this connection can connect to.
Connect URL	jdbc:mysql://localhost:3306/test The Connect URL is JDBC-driver specific. It usually contains the address of the mach 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 33 **database**: The name of the logical database that you are connecting to on the

   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

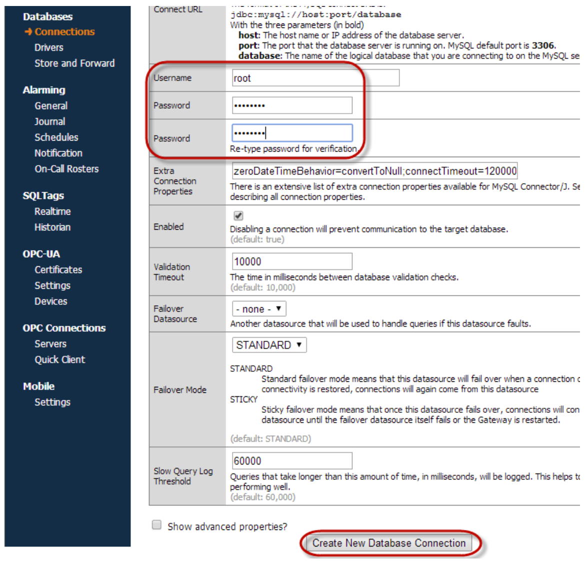
understand that a MySQL server can host many database files. The connection will target one database.

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)

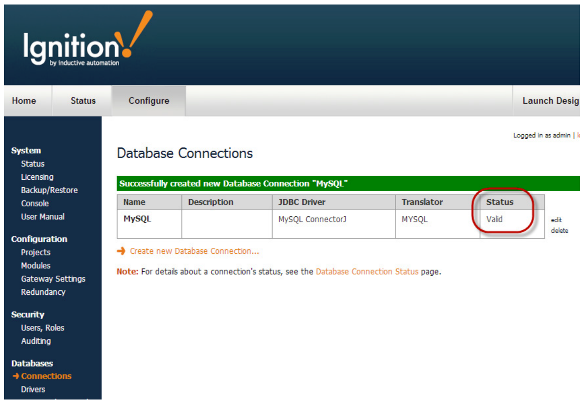
   Extra Connection Properties: Leave at the default value. By default, the **zeroDateTimeBehavior** parameter is defined and it is usually not necessary to add more parameters.



Connect URL	`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 se
Username	root
Password	••••••••
Password	•••••••• Re-type password for verification.
Extra Connection Properties	zeroDateTimeBehavior=convertToNull;connectTimeout=120000 There is an extensive list of extra connection properties available for MySQL Connector/J. Se describing all connection properties.
Enabled	✓ Disabling a connection will prevent communication to the target database. (default: true)
Validation Timeout	10000 The time in milliseconds between database validation checks. (default: 10,000)
Failover Datasource	- none - Another datasource that will be used to handle queries if this datasource faults.
Failover Mode	STANDARD STANDARD Standard failover mode means that this datasource will fail over when a connection c connectivity is restored, connections will again come from this datasource STICKY Sticky failover mode means that once this datasource fails over, connections will con datasource until the failover datasource itself fails or the Gateway is restarted. (default: STANDARD)
Slow Query Log Threshold	60000 Queries that take longer than this amount of time, in milliseconds, will be logged. This helps t performing well. (default: 60,000)

☐ Show advanced properties?

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**.



Logged in as admin |

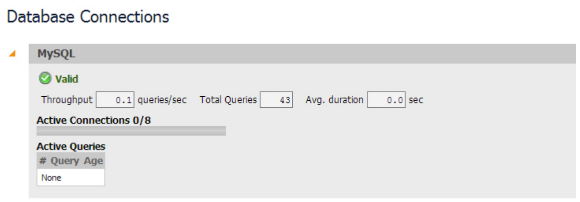
Database Connections

Successfully created new Database Connection "MySQL"

Name	Description	JDBC Driver	Translator	Status
MySQL		MySQL ConnectorJ	MYSQL	Valid

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.



Database Connections

MySQL

Valid

Throughput 0.1 queries/sec Total Queries 43 Avg. duration 0.0 sec

Active Connections 0/8

Active Queries

#	Query	Age
None		

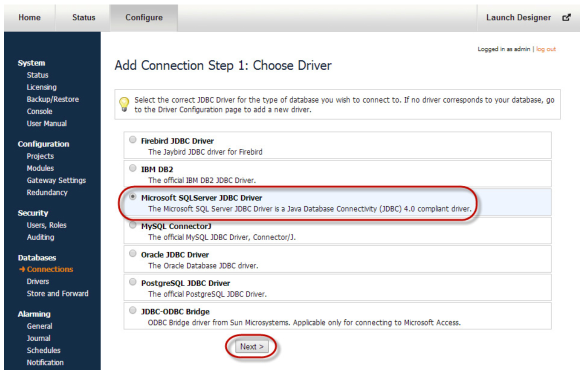
Next ...

- Connecting to Microsoft SQL Server Express
- About Store and Forward
- OPC-UA and Device Connections
- Designer

Connecting to Microsoft SQL Server Express

To connect Ignition to Microsoft SQL Server

1. From the Gateway, go to the **Configure** section.
2. From the menu on the left, select **Databases > Connections**.
   The **Database Connections** page is displayed.
3. Look for the orange arrow, click on **Create new Database Connection...**.
4. Select **Microsoft SQLServer JDBC Driver** and click **Next**.



Add Connection Step 1: Choose Driver

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.

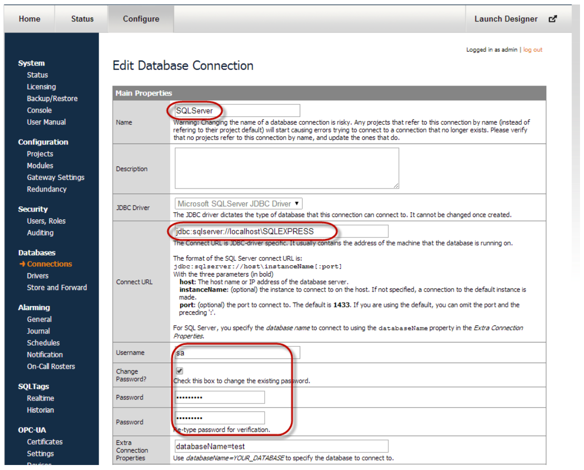
- **Firebird JDBC Driver** – The Jaybird JDBC driver for Firebird
- **IBM DB2** – The official IBM DB2 JDBC Driver.
- [✓ **Microsoft SQLServer JDBC Driver**] – The Microsoft SQL Server JDBC Driver is a Java Database Connectivity (JDBC) 4.0 compliant driver.
- **MySQL ConnectorJ** – The official MySQL JDBC Driver, Connector/J.
- **Oracle JDBC Driver** – The Oracle Database JDBC driver.
- **PostgreSQL JDBC Driver** – The official PostgreSQL JDBC Driver.
- **JDBC-ODBC Bridge** – ODBC Bridge driver from Sun Microsystems. Applicable only for connecting to Microsoft Access.

   The **New Database Connection** page is displayed.

5. On the **New Database Connection** page, enter the following information:
   Name: **SQLServer**
   Connect URL: **jdbc:sqlserver://localhost\SQLEXPRESS**

   Note: 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**)



Edit Database Connection

Main Properties	
Name	SQLServer Warning: Changing the name of a database connection is risky. Any projects that refer to this connection by name (instead of refering to their project default) will start causing errors trying to connect to a connection that no longer exists. Please verify that no projects refer to this connection by name, and update the ones that do.
Description	
JDBC Driver	Microsoft SQLServer JDBC Driver 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
Change Password?	✓ Check this box to change the existing password.
Password	•••••••••
Password	••••••••• Re-type password for verification.
Extra Connection Properties	databaseName=test Use *databaseName=YOUR_DATABASE* to specify the database to connect to.

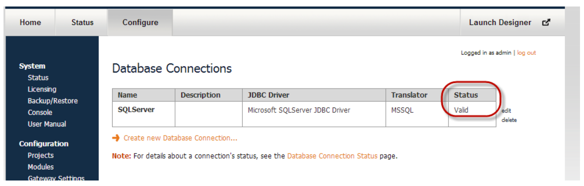
6. At the bottom of the form, click on **Create New Database Connection**.
   Your connection is now created and the **Database Connections** page is displayed showing the **Status** of your connection as **Valid**.

On this page...

INDUCTIVE UNIVERSITY

Connecting to Microsoft SQL Server Express

Watch the Video

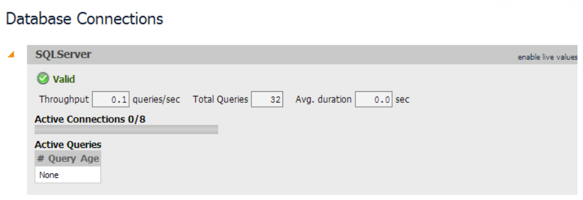


Database Connections

Name	Description	JDBC Driver	Translator	Status
SQLServer		Microsoft SQLServer JDBC Driver	MSSQL	Valid

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.



Database Connections

SQLServer

Valid

Throughput 0.1 queries/sec Total Queries 32 Avg. duration 0.0 sec

Active Connections 0/8

Active Queries

#	Query	Age
None		

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 is Enabled

Ignition connects using TCP/IP, therefore make sure your database has TCP/IP enabled.

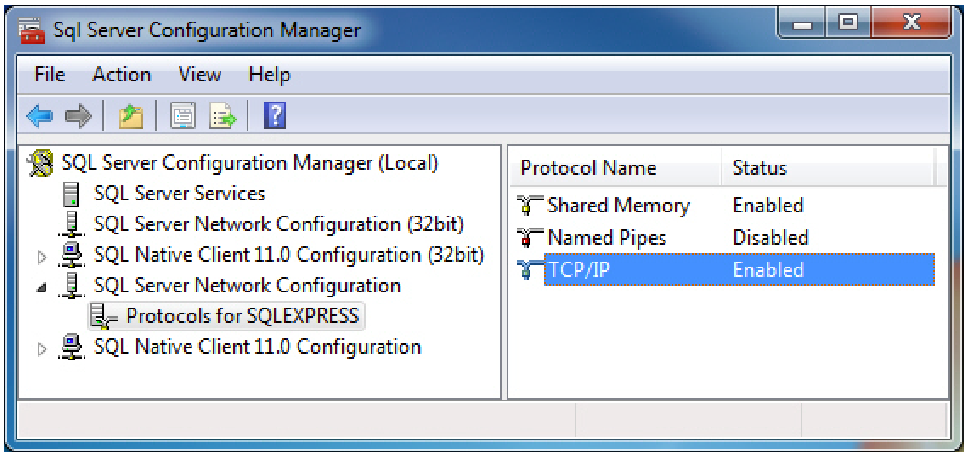
To check if TCP/IP is enabled

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.

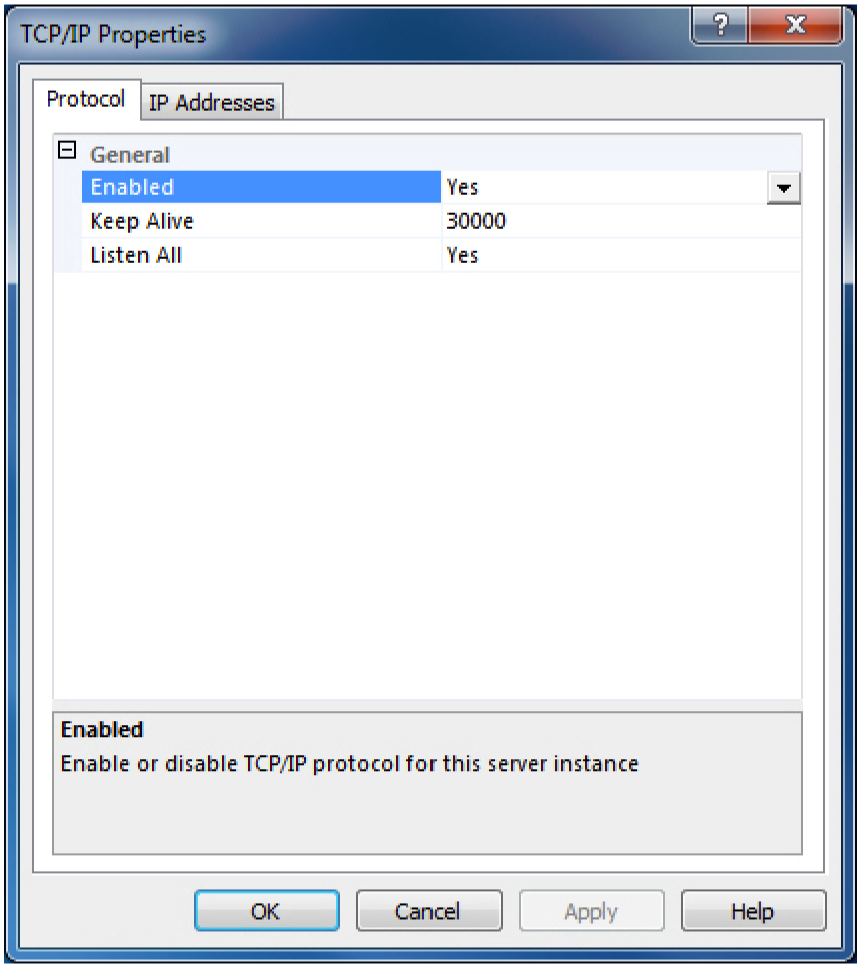
2. To see all the instances setup on that machine, expand **SQL Server Network Configuration**.

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** protoco l and select it.


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

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** .



Enabled	Yes
Keep Alive	30000
Listen All	Yes

Enabled
Enable or disable TCP/IP protocol for this server instance

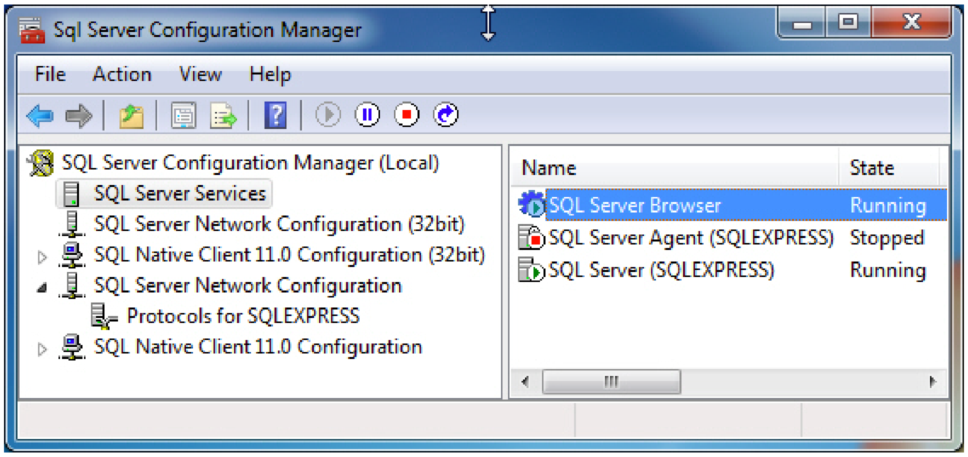
Check 2: Make Sure Microsoft SQL Server Browser is Running

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.

To check if Microsoft SQL Server Browser is running

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

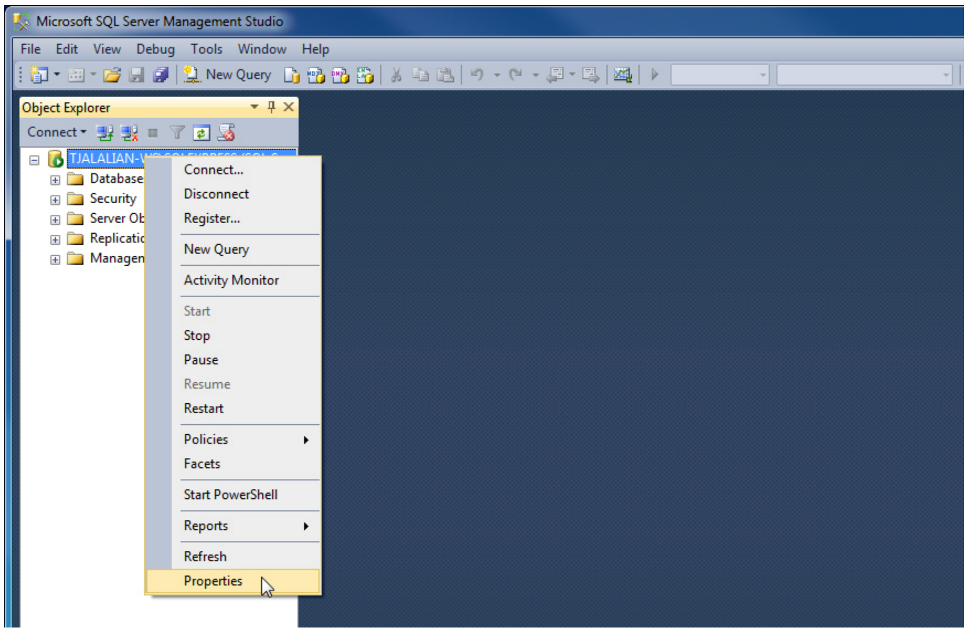
Scenario 1: Connect By Using an Instance Name and SQL Authentication

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.

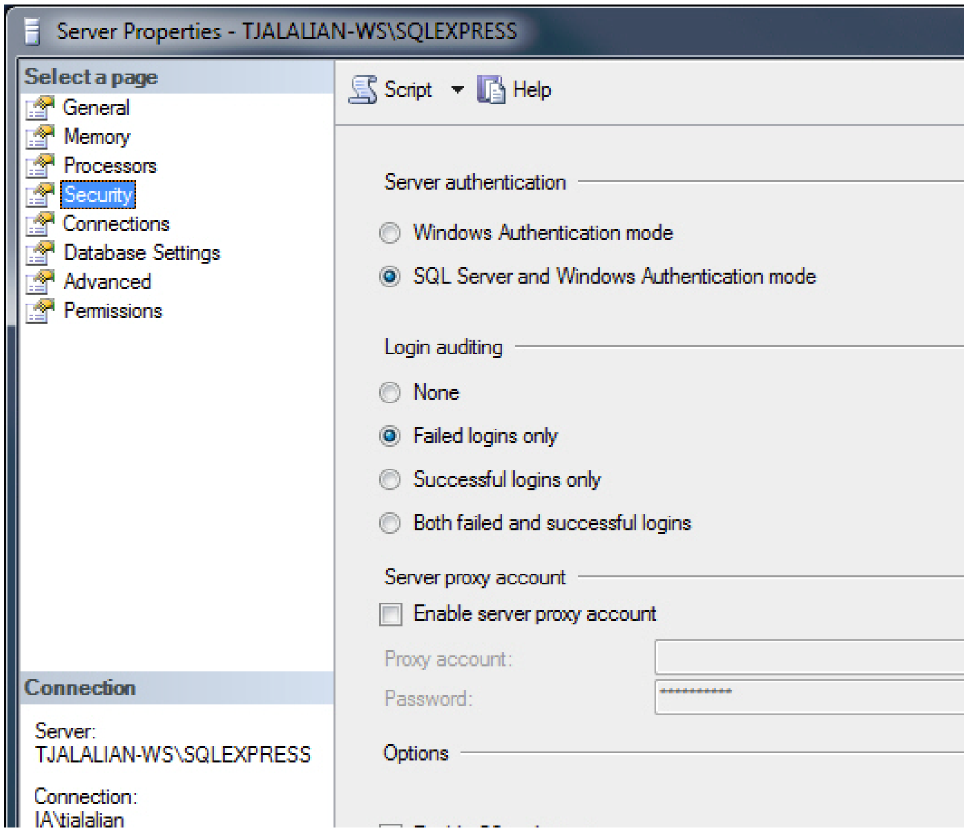
To enable SQL authentication

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

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

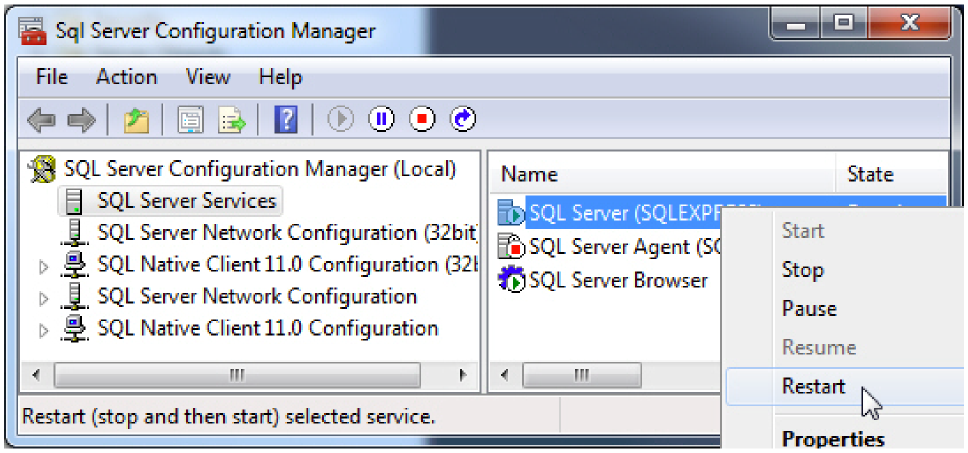
Connection
Server: TJALALIAN-WS\SQLEXPRESS
Connection: IA\tjalalian

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 (32b
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.

To configure the database connection in Ignition

1. Go to and login to the Ignition Gateway configuration 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**.
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.
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.

      Note: 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.

To set up the service to use Windows Authentication

1. You must install the Microsoft JDBC driver package (version 3) before attempting to connect using Windows Authentication. You can download the file **here**.

2. Extract the files to your desktop. Locate the `sqljdbc_auth.dll` file from the correct architecture folder (x86 for 32-bit and x64 for 64-bit) inside of the `enu/auth folders in the zip file`.

3. Copy the `sqljdbc_auth.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\

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).

To configure the database connection in Ignition

1. Go to and login to the Ignition Gateway configuration 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**.

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)

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

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 Dat abase Connection** page.

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.

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.

To enable this, open the SQL Server Management Studio and connect to the server. Right click on the instance and select **Properties**. Under **Security**, select **SQL Sever and Windows Authentication mode**.

Next ...

- Connecting to Oracle Express
- About Store and Forward
- OPC-UA and Device Connections
- Designer

Connecting to Oracle Express

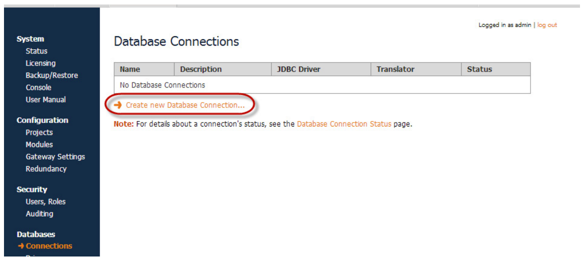
INDUCTIVE UNIVERSITY

Connecting to Oracle Express

Watch the Video

To connect Ignition to the Oracle Express database

1. From the Gateway, go to the **Configure** section.
2. From the menu on the left, select **Databases > Connections**.
   The **Database Connections** page is displayed.
3. Look for the orange arrow, click on **Create new Database Connection...**.



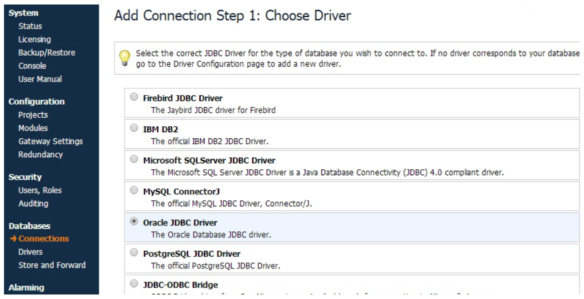
Database Connections

Name	Description	JDBC Driver	Translator	Status
No Database Connections				

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**.

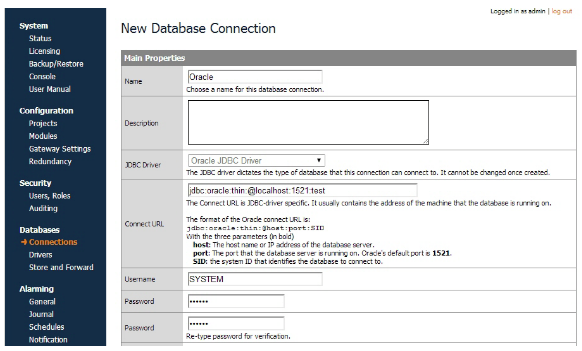


Add Connection Step 1: Choose Driver

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.

- **Firebird JDBC Driver** – The Jaybird JDBC driver for Firebird
- **IBM DB2** – The official IBM DB2 JDBC Driver.
- **Microsoft SQLServer JDBC Driver** – The Microsoft SQL Server JDBC Driver is a Java Database Connectivity (JDBC) 4.0 compliant driver.
- **MySQL ConnectorJ** – The official MySQL JDBC Driver, Connector/J.
- [✓ **Oracle JDBC Driver** – The Oracle Database JDBC driver.]
- **PostgreSQL JDBC Driver** – The official PostgreSQL JDBC Driver.
- **JDBC-ODBC Bridge**

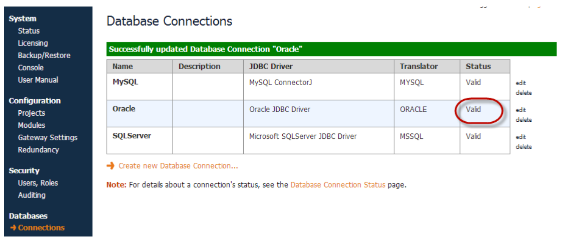
5. Enter the following information:
   Name: **Oracle**
   Connect URL: **jdbc:oracle:thin:@localhost:1521:XE** (**XE** is the System ID in Oracle)
   username: **SYSTEM**
   password: **oracle** (password is what you entered in during the Oracle server installation, in this example, the password is **oracle**)



New Database Connection

Main Properties	
Name	Oracle Choose a name for this database connection.
Description	
JDBC Driver	Oracle JDBC Driver 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.

6. Click **Create New Database Connection** at the bottom of the form.
   The Database Connections page is displayed showing the **Status** of your connection as **Valid**.



Database Connections

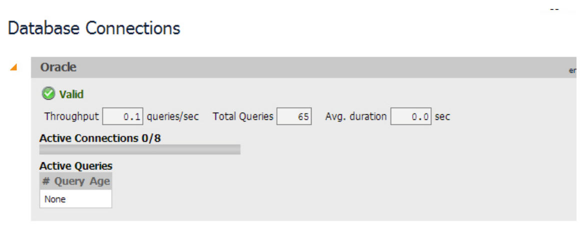
Successfully updated Database Connection "Oracle"

Name	Description	JDBC Driver	Translator	Status	
MySQL		MySQL ConnectorJ	MYSQL	Valid	edit delete
Oracle		Oracle JDBC Driver	ORACLE	Valid	edit delete
SQLServer		Microsoft SQLServer JDBC Driver	MSSQL	Valid	edit delete

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.


Database Connections

Oracle

Valid

Throughput 0.1 queries/sec Total Queries 65 Avg. duration 0.0 sec

Active Connections 0/8

Active Queries

#	Query	Age
None		

Next ...

- Connecting to PostgreSQL
- About Store and Forward
- OPC-UA and Device Connections
- Designer

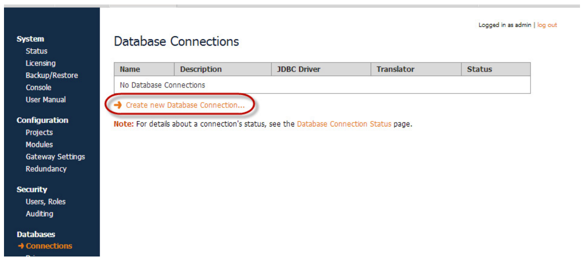
Connecting to PostgreSQL

To connect Ignition to the PostreSQL database

1. From the Gateway, go to the **Configure** section.

2. From the menu on the left, select **Databases > Connections**.
   The **Database Connections** page is displayed.

3. Look for the orange arrow, click on **Create new Database Connection...**.



Database Connections

Name	Description	JDBC Driver	Translator	Status
No Database Connections				

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

4. Select the **PostgreSQL JDBC Driver** and click **Next**.

5. Enter the following information:
   Name: **PostgreSQL**
   Connect URL: **jdbc:postgresql://localhost:5432/test**
   username: **postgres**
   password: password is what you entered in during the Postgres installation.

6. Click **Create New Database Connection** at the bottom of the form.
   The Database Connections page is displayed showing the **Status** of your connection as **Valid**.

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.

INDUCTIVE UNIVERSITY

Connecting to PostgreSQL

Watch the Video

Next ...

- About Store and Forward
- OPC-UA and Device Connections
- Designer

About 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 Transaction Groups use store-and-forward to ensure that data is not lost and is reached 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.

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.

- **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.

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.

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.

Recommended Settings

The following recommended **Engine Settings** (at **Database > Store and Forward** from the **Configure** section of Gateway) are merely a starting point, you can adjust them to fit your goals:

Buffer Settings	
Memory Buffer Size	Set to 500 or higher. It should be high enough to accommodate several bursts of data. For example, if you expect data to be logged at 100 ms burst for 10 seconds at a time, 100 records would be the minimum value. Data is forwarded as it comes in, according to the forward settings, but you should not rely on any particular throughput in order to avoid data loss.
Store Settings	
Disk Cache Enabled	Unselect this check box. Depending on your requirements, disable the disk store, or at least set to a high write size/count settings. Writing and reading from the cache is much slower than memory, so it is desirable to avoid it. Of course, the cache should only be disabled if it is ok to lose some data, should the database connection be down for a period of time.
Forward Settings	
Write Size	This value should be larger than the expected burst size. Burst data is from the same source, and therefore will benefit heavily from the optimizations in the buffer.
Write Time	This value should be balanced in order to give the buffer time to receive multiple records that can be optimized, as described in Write Size above. However, it should not be so long that too much data becomes scheduled to write, which could cause a system slowdown/back up.

In this section ...

Using Store and Forward

All the database connections in Ignition use the **Store and Forward** system. This system offers a number of benefits over other systems that log directly to the database. The main benefits are: data loss prevention and enhanced performance.

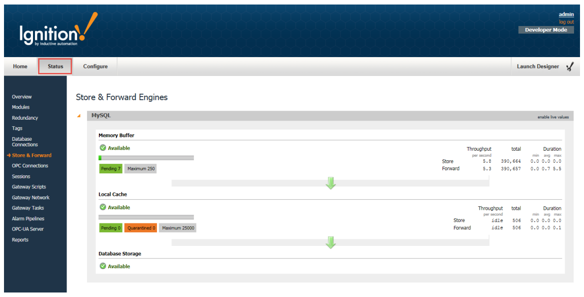
INDUCTIVE UNIVERSITY

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 Engines** page is displayed showing each store-and-forward engine along with the current throughput and capacity of its **Memory Buffer** and **Local Cache**.
3. You can click **enable live values** on the top-right of the page to refresh and update the displayed values.



	Throughput per second	total	Duration min	avg	max
Store	5.8	390,664	0.0	0.0	0.0
Forward	5.3	390,657	0.0	0.7	5.5

	Throughput per second	total	Duration min	avg	max
Store	idle	506	0.0	0.0	0.0
Forward	idle	506	0.0	0.0	0.1

The following status information are shown on the **Store & Forward Engines** page:

Memory Buffer

Memory buffering provides database performance.

- Available
  Shows the whether or not the memory buffer is available.
- Pending
  Shows the maximum number of records that can accumulate in the memory buffer before being written to the cache as you have set with the **Store Settings-Write Size** property.
- Maximum
  Shows the maximum size of the in-memory buffer as you have set it with the **Memory Buffer Size** property.

Local Cache

Local disk caching provides safe storage.

- Available
  Shows the whether or not local cache is available.
- Pending
  Shows the number of records waiting to be forwarded as you have set with the **Forward Settings-Write Size** property.
- Quarantined
  The number of quarantined records for the cache.
- Maximum
  Shows the maximum number of records that can be cached as you have set with the **Store Settings-Max Records** property.

Database Storage

- Available
  Shows the status of database storage as being available or not. If it is available, you can see that the local cache is empty. But if the database storage is not available, you need to find out how much time on your local cache you have left before it is maxed out.

Throughput, Total, and Duration

Shows the diagnostic information and statistic relating to the throughput and duration. It is important to remember how the data flows when interpreting the statistics. The number of rows that have gone to the database will be the number forwarded from the local cache, and then the number forwarded from the memory buffer, minus those that entered the cache from there.

Next ...

- Configuring Store and Forward

Configuring Store and Forward

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.

Tip: You can create multiple database connections pointing to the same database if you wish to configure multiple store-and-forward engines for different purposes.

To configure the Store and Forward engine for your database, do the following steps:

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

Configuring Store and Forward

Watch the Video

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 not be used.
Store Settings **These settings apply to the local disk storage cache.**	
Enable Disk Cache	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. 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.

3. 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.

Next ...

- 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 the issue is resolved, then you can either delete it or re-insert it into the queue.

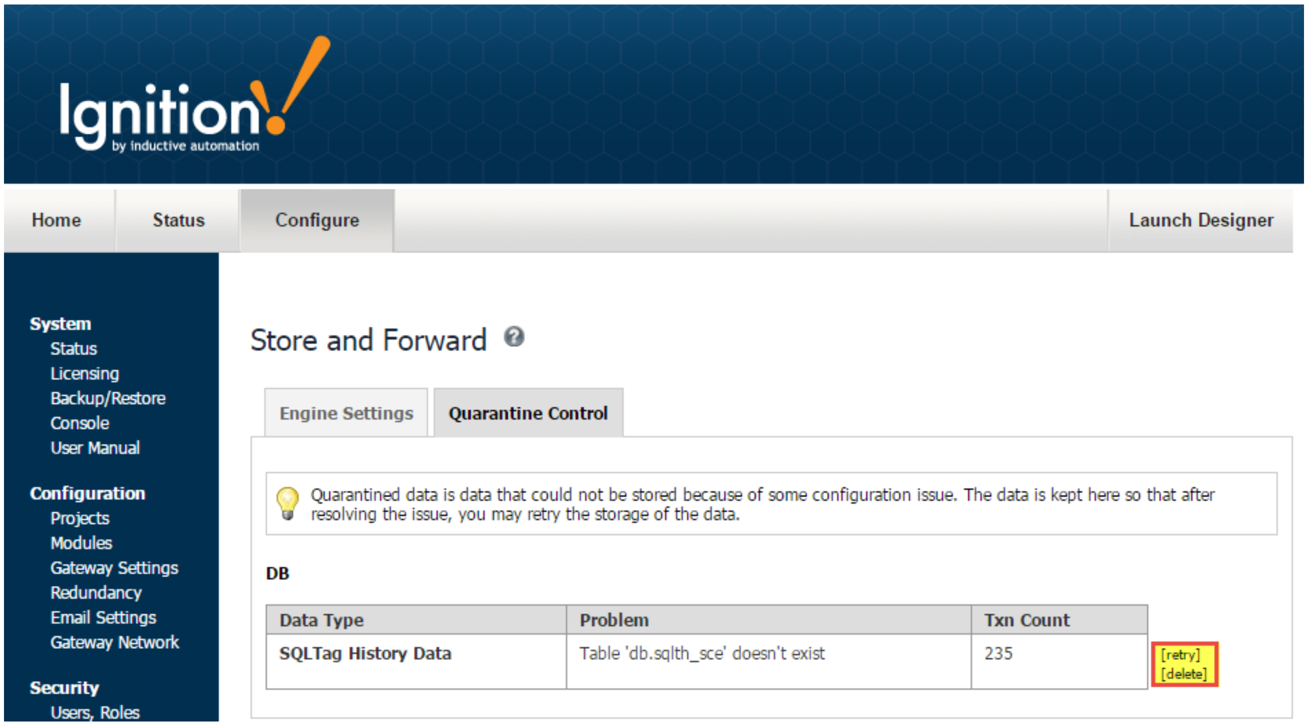
INDUCTIVE UNIVERSITY

Controlling Quarantine Data

Watch the Video

To handle the quarantined data

1. From **Configure** section of the Gateway, go to **Databases > Store and Forward**.
2. Go to the **Quarantine Control** tab and look for the problem/error.
   Here you will see any quarantined data which are listed according to store-and-forward engine and the data format, with a description, the error that caused the quarantine, and the number of quarantined records. On the right side of these records you can see **retry** or **delete**.
3. Fix the problem/error you found that caused the quarantine data.
4. Click either on **retry** or **delete.**
   retry will un-quarantine the data and place the engine back in the forward queue.
   delete will permanently delete the data in the selected row. All transactions of the selected type are deleted.



Quarantined data is data that could not be stored because of some configuration issue. The data is kept here so that after resolving the issue, you may retry the storage of the data.

DB

Data Type	Problem	Txn Count	
SQLTag History Data	Table 'db.sqlth_sce' doesn't exist	235	[retry] [delete]

Next ...

- Connecting to Databases
  or
- Connecting to a Device
  or
- Tag Historian

OPC-UA and Device Connections

Note: Connecting to a device at this point is an optional step! You can come back to it later. However, the next steps will be more interesting if you add a device now.

Now that we've installed Ignition and have logged into the Configuration section of the Gateway web interface, lets connect to a device. You can find detailed descriptions for many device connections in this User Manual, but they all include the following three main steps:

1. **Create new Device**
   From the left side configuration menu of the Gateway, you go to the **OPC-UA > Devices** section. Once at the **Devices** page, click on the **Create new Device...** link at the bottom of the table.
   A device is a named connection to an industrial device like a PLC. There are also **Simulator** devices in Ignition that you can connect to which act like a real device connection in case you don't have one available. Device connections are done over the Ethernet for those devices that have an Ignition device driver.

2. **Choose a Driver**
   A list is displayed so you can choose the driver for the device you want. If you don't have a device that matches one of the available drivers, you can add a **Simulator** device so you have some data to play with.

3. **Configure the Device**
   After you choose the driver, you give your device a name and set some options. Typically, the options are just an IP address to connect to. After configuring the device, click the **Create New Device** button to add your device. You can check the connectivity status of your device in the Gateway Status section, under Ignition OPC-UA Server.

Connecting to a Device

See the following links for the type of device connection you are interested in:

- Allen Bradley Ethernet
- Modbus
- Siemens
- UDP and TCP Driver
- Third Party OPC Servers
- Connecting to DNP3 Devices

What is OPC?

OPC is a specification for the transport and use of industrial data. It is published and maintained by the OPC Foundation, an organization comprised of hundreds of member companies that strives to ensure interoperability on the plant floor and beyond.

History

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

DCOM always proved difficult to work with, and by 2004 it was clear that a more modern solution was needed. Therefore, a new specification was developed that used common networking principals instead of DCOM, was platform independent, and combined the various separate specifications into one: OPC-UA.

OPC-UA

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

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

To Continue on the step-by-step simple workflow...

After you **Connect to a Device**, as described on this page, click the link below for Step 3 to continue.

Step 1. **Launch the Gateway**

Step 2. **Connect to a Device**

Step 3. **Connect to a Database**

Step 4. **Launch the Designer**

Step 5. **Create Tags**

Step 6. **Launch Clients**

Step 7. **Add History to Tags**

Step 8. **Create a Graph and View Graphs in Clients**

Steve Hechtman Blog

Click the link below to see the blog:

OPC-UA goes mainstream

Distributed Systems with OPC-UA

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

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

Servers and Clients

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

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

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

Technology

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

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

Connecting to a OPC Server

OPC-UA Server

An OPC-UA Connection is used to communicate with an OPC-UA compliant server, such as the one the OPC-UA module provides.

The following steps walk-through connecting Ignition (as an OPC-UA client) to a OPC-UA server:

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

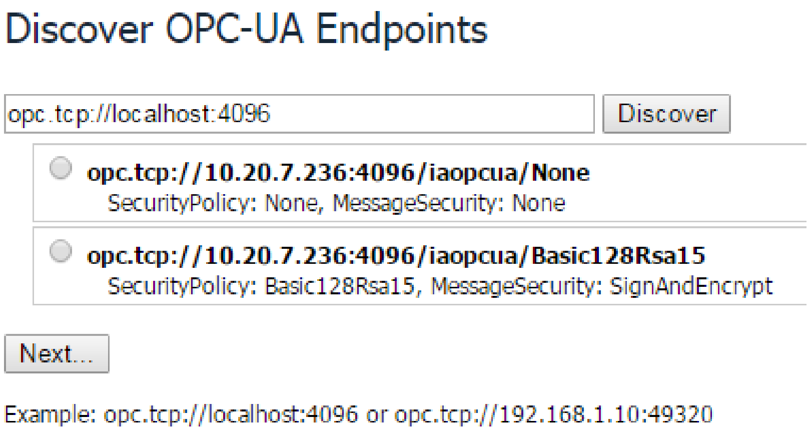
2. Click on the **Create new OPC Server Connection** link.

3. Select **OPC-UA Connection** from the list and click **Next**.
   The **Discover OPC-UA Endpoints** page will appear.

4. Type in a OPC-UA endpoint for the OPC-UA Server Ignition should connect to. The format should be as follows: `opc.tcp://IpAddress:Port`
   Ignition can connect to it's own OPC-UA server's endpoint by using localhost and the default port of 4096: `opc.tcp://localhost:4096`

   A hostname could be used instead of an IP address: `opc.tcp://myhostname:4096`

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


Discover OPC-UA Endpoints

opc.tcp://localhost:4096 Discover

opc.tcp://10.20.7.236:4096/iaopcua/None
SecurityPolicy: None, MessageSecurity: None

opc.tcp://10.20.7.236:4096/iaopcua/Basic128Rsa15
SecurityPolicy: Basic128Rsa15, MessageSecurity: SignAndEncrypt

Next...

Example: opc.tcp://localhost:4096 or opc.tcp://192.168.1.10:49320

6. Select a Security Policy and Message Security configuration to use when connecting to the endpoint. **Basic128** SecurityPolicies and **Sign AndEncrypt** MessageSecurity are more secure than **None.** If available, **Basic256** should be selected.

   Once an endpoint configuration has been selected, click the **Next** button.
7. On the **New OPC UA Connection Settings** page, give the connection a name.

   Some OPC-UA servers may require a Username and Password, but this is not always the case. Check with the OPC-UA server's documentation for more details.

   By default, Ignition's OPC-UA server requires the following credentials:

Username	opcuauser
Password	password

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

   Ignition is now connected to the OPC-UA server.

OPC-UA Server properties

Main	
Name	A name used to identify this connection.
Description	Short description of this connection.
Read-only	Puts the connection into read-only mode. All writes sent to this server will fail.
Enabled	Disable the connection to the OPC server.
Authentication	
Username and Password	If a username and password are specified, they are used as a user identity token when connecting to the specified OPC-UA server. The internal OPC-UA server provided by the OPC-UA module uses an Ignition security profile to govern who can connect to it. This can be configured in the **OPC-UA > Settings** section.
Change Password?	Enable this property to change the configured password.
Advanced	
Host Override	When specified, if the endpoint address returned by the OPC server has a different IP address or hostname than the discovered endpoint, the overridden value will be used. Expects just an IP address or hostname. Example: 192.168.1.10
Max Per Operation	Specify the maximum number of nodes to read, write, subscribe, or unsubscribe to in any given UA server request.

Failover Settings	
Failover Enabled	When enabled, if this OPC-UA server connection is faulted, requests will be sent to a failover OPC-UA server.
Failover Endpoint	The endpoint of the failover server. *Example*: opc.tcp://192.168.1.0:4096
Failover Threshold	The number of retry attempts before the failover connection is used.

The following feature is new in Ignition version **7.8.3**
Click here to check out the other new features

Failover Settings	
Backup Discovery URL	The discovery URL for the backup server's OPC-UA server. Expects the following format: `opc.tcp://hostname:port`
Backup Endpoint URL	The full endpoint URL for the backup gateway. An example endpoint would look like the following: `opc.tcp://10.20.1.100:4096/iaopcua/Basic128Rsa15`

How do I determine the Endpoint for my backup?

See step 5 in the Connecting to a OPC-UA Server guide above. Clicking the Discover button will show a list of available endpoints to connect to.

Failover Versus Backup Properties

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

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

Connecting to OPC Classic (COM)

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

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

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

To Install OPC Core Components

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

2. Download appropriate OPC Core Components Redistributable package.
   There are two packages to choose from, the 32-bit (x86) and the 64-bit (x64), make sure you get the correct one for the version of Java and Ignition you are running. 64-bit Java and Ignition needs the 64-bit Core Components package and likewise 32-bit installations needs the 32-

bit package.

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

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

Connecting to OPC-DA Server

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

To Install OPC-DA server connection

1. Go to the Ignition Gateway **Configure** section (http://localhost:8088/main/web/config).
2. Go to **OPC Connections > Servers** and then select **Create new OPC Server Connection...**.

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

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

   Unique Remote Connection Settings
   Remote connections have a few unique settings that you can specify. You can get to these settings by selecting the **Show advanced properties** check box. As of Ignition 7.4 these should all be set for you (except for the CLSID which should no longer be necessary but is still available for you to set if you wish).

   Remote Server
   Specifies that the server is remote and that a DCOM connection will be used.

   Host Machine
   The computer name or IP address of the machine on which the remote server is running.

   CLSID
   This is no longer required as of Ignition 7.4, but it is still made available for you. It can be used in place of the ProgId because the ProgId is really just used to lookup the CLSID in the registry. This id can be found in the registry of the machine hosting the server under: HKEY_CLASSES_ROOT\OPCServerName\CLSID

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

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

Connection is Faulted

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

Troubleshooting OPC-COM Connections

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

Common Problems

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

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

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

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

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

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

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

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

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

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

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

To set the OPC server to run as Interactive User

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

Faulted status with E_CLASSNOTREG error reported on OPC connections status page

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

DCOM Settings

Ignition Server DCOM Settings

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

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

2. Browse down through the **Component Services** tree until you see **My Computer**, right-click and select **Properties**.

3. We want to focus on the COM Security tab. There are two sections, **Access Permissions and Launch** and **Activation Permissions**. Each section has an **Edit Limits...** and **Edit Defaults...** button. You must add the ANONYMOUS and Everyone accounts under each of the four areas making sure that the **Allow** option is checked for each of the permission settings. If you skip adding both of these to either the limits or defaults areas under either of the two sections there is a good chance your connection will not be successful.

4. You can also try setting the **Default Authentication Level** to **None** and the **Default Impersonation Level** to **Identify** on the Default Properties tab. This isn't always necessary but it can sometimes help.

OPC Server DCOM Settings

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

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

4. Click the **Security** tab and you will see three sections, **Launch and Activation Permissions**, **Access Permissions**, and **Configuration Permissions**. There are two options to choose from for each section. If you already added the ANONYMOUS and Everyone accounts to the COM Security section from the **Ignition Server DCOM Settings** section then you can go ahead and just select the **Use Default** option for each of the three areas. The second option is to edit each of the groups that have **Customize** selected. You will have to add both the **ANONYMOUS** and **Everyone** accounts with all privileges.
5. Now select the **Identity** tab. You will notice that you can choose which account you want to run the OPC server under. Select the **Interactive User** option. This ensures that if Ignition launches an instance of the OPC server, it will run under whichever user is currently logged into the system.

Creating an OPC-HDA Connection

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

1. Navigate to the Ignition gateway configuration section, as outlined above.
2. Under Tags, History, select "Create new Historical Tag Provider..."
3. Select "OPC-HDA Provider"
4. Follow the step outlined above, for DA connections.
5. Once complete, the status on the Tags>History screen will show the state of the connection. If Connected, you should now be able to browse and query the server through the Ignition designer.

Example - Adding OPC-HDA data to a chart

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

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

OPC Quick Client

You can access the OPC Quick Client from under the **OPC Connections** section of the Ignition Gateway **Config** section. It allows for quick, simple testing of any devices connected to the server.

You can browse by expanding tree nodes and read/write to tags by clicking on the [r] and [w] buttons next to those tags.

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

Ignition OPC-UA Server

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

Authentication	
Authentication Profile	The User Source that the OPC-UA module will use to authenticate incoming connections against. By default, this is set to the **opcua-module** User Source. This profile is included in the default installation and has the following as its default settings: user: opcuauser password: password
Allowed Roles	Roles within the given User Source that are allowed to connect to the server. Multiple roles should be separated by a comma, for example, `Administrator,user,manager`.
Allow Anonymous Access	If checked, will allow anonymous connections to the server. Not checked by default.
Server	
Server Port	The port the OPC-UA module runs on.
Endpoint Address	This is the local address that the Ignition UA server will bind to. It is also the address that will be used in a GetEndpointResponse, so it is important that this be an address reachable by any clients that wish to connect. Requires a module restart to take effect. This is useful if the server machine has a VPN connection or multiple adapters and is returning the wrong address.
Minimum Sampling	The fastest rate (in milliseconds) that the server will use to sample its underlying data sources. Requires a module restart to take effect. The default is 100.

Interval	
Discovery Server	
Discovery Server Enabled	Select the check box if the local discovery server should be enabled. Requires a module restart to take effect. By default, the check box is selected or True.
Expose Configured Tags	
Expose Tag Providers	If enabled, Ignition tag providers will be exposed through the OPC-UA server, allowing third party clients to access the tags configured in the system. By default, the check box is not selected or False.
Other	
Stale Threshold	The multiplier by which the server determines that updates from a driver have become stale. This period will be calculated as the fastest sampling rate for that node multiplied by this settings value. Default is `5`.
Allow Untrusted Certificates	Whether or not to automatically accept incoming certificates or to rely upon them being placed into the trusted certificate list before allowing a connection. By default, the check box is selected or True.
Discovery Server	Address of the discovery server to register with. Default is `localhost`.
Discovery Registration Frequency	Frequency (in minutes) at which the server will attempt to register with the discovery server. Default is `10`.
Auto-cast Writes	Attempt to auto-cast incoming writes to the correct DataType before rejecting them with a `Bad_InvalidType StatusCode`. By default, the check box is selected or True.

Creating a New Device

To add a new Device go to the **Devices** section of the OPC-UA module configuration in the Ignition Gateway, go to the **Configure** section of the Gateway and then to **OPC-UA > Devices**. Once at the **Devices** page, click on the **Create new Device...** link at the bottom of the table. You will be taken to a page where you can select the driver to use. Choose your driver and click **Next**.

The **General** settings common to all devices are as follows:

General	
Name	The user-defined name for this Device. The name chosen will show up in **OPC Item Paths** and under **OPC-UA Server > Devices** of the **Status** page of the Gateway. The Device Name must be alphanumeric.
Enable Device	Only devices that are enabled appear in **OPC-UA Server > Devices** of the **Status** page of the Gateway and thus have their tags available for use.
Connectivity	
The settings under **Connectivity** are device-specific settings and unique to each driver.	
Timeout	The timeout settings refer to the communication between the device driver and the OPC-UA server and usually can be left at their default values.

Verifying Device Connectivity

Device connectivity can be verified in the following three places:

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

Drivers

Allen Bradley Drivers

ControlLogix 5500

You will see the following **Connectivity** settings on the **New Device** page accessed from the **Configure** section of the Gateway under **OPC-UA > Devices**.

Connectivity	
Hostname	This is the IP Address of the ControlLogix Ethernet module (1756-ENET) to route through to connect a ControlLogix processor. EthernetIP protocol on TCP port 44818 (0xAF12) is used to communicate to ControlLogix processors.
Timeout	After sending a request to the ControlLogix processor, the Communication Timeout setting is the amount of time in msec to wait for a response before treating it as a failure.
Slot Number	The Slot Number value is the zero based ControlLogix chassis slot number of the ControlLogix processor to connect to.
Connection Path	The Connection Path value is used to define the route of the PLC-5 processor to connect to. Currently routing through the ControlLogix Ethernet Communication Interface Module (1756-ENET) to the ControlLogix Data Highway Plus-Remote I/O Communication Interface Module (1756-DHRIO) and on to a PLC-5 processor of the DH+ network is supported. The Connection Path format contains 4 numbers separated by commas. The first number is always 1 and tells the 1756-ENET module to route through the backplane. The second number is the slot number of the 1756-DHRIO module of the DH+ network the PLC-5 processor is connected to. The third number is the channel of the 1756-DHRIO module that the PLC-5 processor is connected to. Use 2 for channel A and 3 for channel B. The final and fourth number is the DH+ node number. This number is in octal and is the same as configured in the PLC-5 processor. See the ControlLogix Ethernet Communication interface Module User Manual for more information. Connection Path Format: 1,<1756-DHRIO slot number>,<1756-DHRIO channel>,<DH+ node number> The valid range for the 1756-DHRIO slot number is between 0 and 16 but depends on the chassis size. The 1756-DHRIO channel is either 2 for channel A or 3 for channel B. The DH+ node number range is from 00 to 77 octal. For a more in depth explanation of connection paths, see Allen Bradley Connection Paths Explained. **Supported Connection Methods** ControlLogix 5500 connected through 1756-ENET/A or 1756-ENET/B.
Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt your communications with the device.

MicroLogix 1100/1400

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

Supported MicroLogix Connection Methods

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

Note: MicroLogix 1200 and 1500 are not fully supported. Browsing is not available on these devices.

PLC-5

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

More Information On Connection Path

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

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

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

For a more in depth explanation of connection paths please read: Allen Bradley Connection Paths Explained.

Supported PLC-5 Connection Methods

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

SLC 505

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

More Information On Connection Path

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

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

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

For a more in depth explanation of connection paths please read: Allen Bradley Connection Paths Explained.

Supported SLC Connection Methods

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

Allen Bradley Connection Paths Explained

Connections to ControlLogix, CompactLogix, PLC-5, MicroLogix and SLC Allen-Bradley processors through a ControlLogix Gateway require a connection path. The connection path is unique to your setup and is dependent on what modules the connection is being routed through. With there being nearly an endless number of ways to route your connection from device to device it is impossible to give an example of every possible connection path, but in general there is a pattern to how the connection path is specified.

Follow the Path

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

Steps

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

Connection Path Entries for Different Module Types

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

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

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

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

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

Some examples have been included to help illustrate the process of tracing a connection path. The first three examples illustrate how to build your connection path when going from one ControlLogix Gateway to another. The last example shows connecting through a ControlLogix Gateway to 3 different SLC 5/04 devices via DH+.

Creating an OPC-HDA Connection

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

1.	Navigate to the Ignition gateway configuration section, as outlined above.

2.	Under Tags, History, select "Create new Historical Tag Provider..."

3.	Select "OPC-HDA Provider"

4.	Follow the step outlined above, for DA connections.

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

Example - Adding OPC-HDA data to a chart

1.	Open the designer, and create or open a project.

2.	Create a window, and add an Easy Chart component.

3.	Double-click on the chart, or right click and select Customizers>Easy Chart Customizer to bring up the chart customization window.

4.	Next to the "Tag History Pens" table, select the first button, "Browse for tags". This will display a tree for browsing all historical tags.

5.	Browse through your defined HDA server. Once you find a tag, select "ok" to add it to the chart.

6.	You may edit the tag to alter its aggregation mode, though the HDA provider will select a supported mode automatically if the specified mode does not exist in the server.

7.	Once you save the configuration, the chart should update with the requested data.

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

A Frequently Asked Question

How to I get data from my PLC?

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

1. Add a device, see Connecting to a Device.
2. Add some tags, see Creating Tags.

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

Brief summary of device connection in Ignition

- Ignition can only connect directly to devices over Ethernet.
- Ignition can only connect directly to devices for which there is an Ignition device driver. Included drivers are:
  - Allen Bradley - ControlLogix 5500, CompactLogix, MicroLogix 11/1400, PLC-5, SLC 505
  - Siemens - S7-300, S7-400, S7-1200
  - Modbus - The Modbus driver connects to any ethernet enabled device that uses the Modbus protoco
- Ignition can connect to third party OPC servers via OPC-UA or OPC-DA (using the OPC-COM module) for devices that do not have a supported driver.

Adding a Device to Ignition

Ignition Supported OPC-UA Device

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

1. Click "Add a device..."
2. Select the driver for the device you wish to add
3. When adding a device you will notice that there are some common settings that are shared by all devices. You can find an explanation of these settings here: Adding a New Device
4. Specify any of the required device specific settings for the device (For example, hostname, etc.)
5. Check the status of your device to see if it is connected.

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

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

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

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

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

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

Adding tags for Allen Bradley devices

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

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

Simulator Drivers

Generic Simulator

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

There are no configurable settings for the generic simulator.

Simulator tags

ReadOnly - static values that do not change for read only purpose.

ReadOnlyBoolean1 - false
ReadOnlyBoolean2 - true
ReadOnlyShort1 - 1
ReadOnlyShort2 - 2
ReadOnlyInteger1 - 1
ReadOnlyInteger2 - 2
ReadOnlyLong1 - 1
ReadOnlyLong2 - 2
ReadOnlyFloat1 - 1.1
ReadOnlyFloat2 - 1.2
ReadOnlyDouble1 - 1.1
ReadOnlyDouble2 - 1.2
ReadOnlyString1 - "ABCDEFG"
ReadOnlyString2 - "ZYXWVUT"

Writeable - static values that you can read/write to, initial values below.

WriteableBoolean1 - false
WriteableBoolean2 - false
WriteableShort1 - 0
WriteableShort2 - 0
WriteableInteger1 - 0
WriteableInteger2 - 0
WriteableLong1 - 0
WriteableLong2 - 0
WriteableFloat1 - 0
WriteableFloat2 - 0
WriteableDouble1 - 0
WriteableDouble2 - 0
WriteableString1 - "" (empty string)
WriteableString2 - "" (empty string)

Random - Random values updating at some rate, they follow Java Random(rate) - rate is the seed.

RandomBoolean1 - 10 sec
RandomShort1 - 5 sec
RandomInteger1 - 1 sec
RandomLong1 - 2 sec
RandomFloat1 - 10 sec
RandomDouble1 - 10 sec

Sine - Different sine waves with frequency, amplitude and offset (listed in that order)

Sine1 - 0.1, 100.0, 0.0
Sine2 - 0.01, 50.0, -25.0
Sine3 - 0.02, 10.0, 10.0
Sine4 - 0.04, 100.0, 0.0
Sine5 - 0.08, 100.0, 0.0

Ramp - Ramp signals starting from 0 going up to some value at the specified rate. When they reach their upper limit, they are reset to zero.

Ramp1 - 0 - 100 @ 10 ms
Ramp2 - 25 - 75 @ 100 ms
Ramp3 - 0 - 100 @ 50 ms
Ramp4 - 0 - 100 @ 25 ms
Ramp5 - 0 - 100 @ 12.5 ms

Realistic - Values determined by adding a random number (between -1 and 1) to the current value.

Realistic1 - -50 - 50 @ 500 ms
Realistic2 - -50 - 50 @ 1000 ms
Realistic3 - -50 - 50 @ 1500 ms
Realistic4 - -50 - 50 @ 2000 ms
Realistic5 - -50 - 50 @ 2500 ms

Allen Bradley SLC Simulator

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

In this section ...

Allen Bradley Ethernet

Ignition contains several Allen Bradley drivers, they are: ControlLogix, CompactLogix, PLC-5, MicroLogix, and SLC Allen-Bradley. These drivers can connect directly through the Gateway to devices that support Ethernet communications. This section contains step-by-step instructions for connecting to these drivers.

In this section ...

Connecting to CompactLogix

To connect Ignition to an Allen-Bradley CompactLogix device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley CompactLogix**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **CompactLogix**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.55
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **CompactLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, go to **OPC Connections > Quick Client** in the **Configure** section, on the **OPC Quick Client** page expand the **CompactLogix** folder and in the **Global** folder you can see all the tags.

INDUCTIVE UNIVERSITY

Connecting to CompactLogix

Watch the Video

Next ...

- add link here

Connecting to ControlLogix

To connect Ignition to an Allen-Bradley ControlLogix device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley ControlLogix**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **ControlLogix**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.50
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **ControlLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, go to **OPC Connections > Quick Client** in the **Configure** section, on the **OPC Quick Client** page expand the **ControlLogix** folder and in the **Global** folder you can see all the tags.

Connecting to ControlLogix

Watch the Video

Next ...

- add link here

Connecting to ControlLogix v21

To connect Ignition to an Allen-Bradley ControlLogix v21 device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley Logix5000 Driver**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **ControlLogix**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.57
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **ControlLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, go to **OPC Connections > Quick Client** in the **Configure** section, on the **O PC Quick Client** page expand the **ControlLogix** folder and in the **Global** folder you can see all the tags.

INDUCTIVE UNIVERSITY

Connecting to ControlLogix v21

Watch the Video

Next ...

- add link here

Connecting to MicroLogix

To connect Ignition to an Allen-Bradley MicroLogix device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley MicroLogix**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **MicroLogix**
   Hostname: type the **IP address** for the PLC, for example 10.20.7.77
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **MicroLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, go to **OPC Connections > Quick Client** in the **Configure** section, on the **OPC Quick Client** page expand the **MicroLogix** folder which contains all the folders with the individual tags.

Connecting to MicroLogix

Watch the Video

Supported MicroLogix Connection Methods

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

Important

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

Additionally, the MicroLogix driver can not access Input Parameters.

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

Next ...

- add link here

Connecting to PLC5

To connect Ignition to an Allen-Bradley PLC5 device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley PLC5**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **PLC5**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.56
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **PLC5** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, go to **OPC Connections > Quick Client** in the **Configure** section, on the **OPC Quick Client** page expand the **PLC5** folder which contains all the folders with the individual tags.

Connecting to PLC5

Watch the Video

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

Next ...

- add link here

Connecting to SLC

To connect Ignition to an Allen-Bradley SLC (5/05 and 5/04) device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley SLC**, and click **Next**.
   Note that these PLCs do not have a native Ethernet connection, therefore another device like a Net ENI or an ENBT must be used for the connection.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **SLC**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.56
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **SLC** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, go to **OPC Connections > Quick Client** in the **Configure** section, on the **OPC Quick Client** page expand the **SLC**folder which contains all the folders with the individual tags.

Connecting to SLC

Watch the Video

Next ...

- add link here

ControlNet Example

Before we show you how to connect to a PLC5 through a ControlLogix Gateway using ControlNet, you need to understand what a **connection path** is and how to build one.

On this page ...

ControlNet Example

Watch the Video

Connection Paths

Connections to ControlLogix, CompactLogix, PLC-5, MicroLogix and SLC Allen-Bradley processors through a ControlLogix Gateway require a connection path. The connection path is unique to your setup and is dependent on what modules the connection is being routed through.

Follow the Path

A connection path is exactly what the name implies. It is a path that when followed leads to a processor residing in a numbered slot of a chassis somewhere on site. You only have to follow the path and build the connection path as you go.

The first connection point between Ignition and the device is a ControlLogix Ethernet module such as an ENET, ENBT, or EN2T module. The slot number of this module doesn't matter and you don't need to specify it in the connection path.

Building the Connection Paths Entries

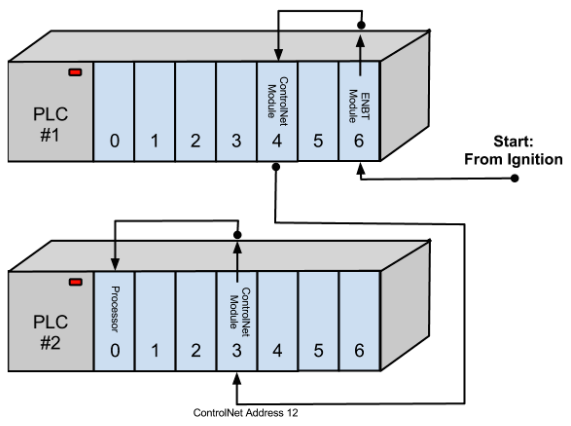
You need to have 6 numbers/entries to specify the connection path. The way you find each number is described in the following table:

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

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

Here is an example, as pictured below, that has a **connection path** of **1,4,2,12,1,0** going from **PLC 1** to **PLC 2**. The following table shows the walk-through for these numbers:

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



Connecting to a PLC5 using ControlNet

To connect to a PLC5 through a ControlLogix Gateway using ControlNet in Ignition, do the following steps:

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley PLC5**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **PLC5**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.62
   Connection Path: **1,3,2,14** (**1** means going to ControlLogix and going to the backplane in PLC 1, **3** is the slot number of the ControlNet module to move to in this example, **2** is the exit port of the ControlNet module, **14** is the address of entry point to the ControlNet module in PLC 2)
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **PLC5** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, go to **OPC Connections > Quick Client** in the **Configure** section, on the **OPC Quick Client** page expand the **PLC5** folder which contains all the folders with the individual tags.

Connection Path Entries for Different Module Types

How you specify your exit point from a module is slightly different depending on which module type you are using. You can only move in two directions once you are **in** a module, either out to the back plane, or out through the module port/channel.

Ethernet modules have ethernet ports and an IP address. ControlNet modules have ControlNet Ports and ControlNet addresses. DHRIO modules have channels and station numbers.

Below is a list of different kinds of modules and what numbers you specify in the connection path when you are exiting or entering those modules. When in a module, an entry of 1 will always take you to the back plane.

ENET, ENBT, and EN2T

Exiting

1 = Backplane
2 = Ethernet Port

Entering

IP Address

CNB

Exiting

1 = Backplane
2 = ControlNet Port

Entering

ControlNet Address

DHRIO

Exiting

1 = Backplane
2 = DH+ Channel A
3 = DH+ Channel B

Entering

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

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

Next ...

- An ENBT Example

An ENBT Example

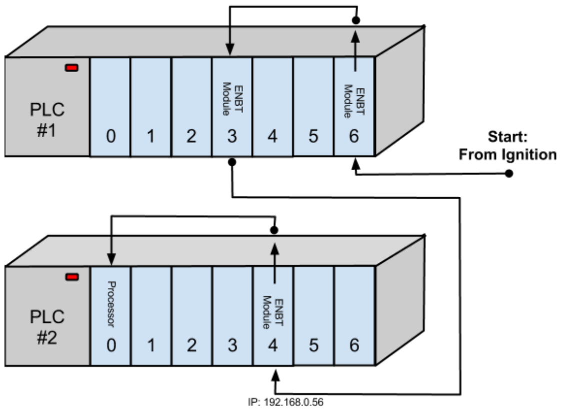
An ENBT Example

Watch the Video

Before we show you how to connect to a ControlLogix through a ControlLogix Gateway using ENBT, you need to understand what a **connection path** is and how to build one, see the Connection Paths , Follow the Path, and Building the Connection Paths Entries sections.

Here is an example, as pictured below, that has a **connection path** of **1,3,2,192.168.0.56,1,0** going from **PLC 1** to **PLC 2**. The following table shows the walk-through for these numbers:

Example	
Connection Path	**Walk-through from PLC 1 to PLC 2**
1	Move out of ENBT Module to the backplane
3	Move to ENBT Module in Slot 3
2	Move out Ethernet Port
192.168.0.56	Move in EBNT Module at IP Address 192.168.0.56
1	Move out of EBNT to the backplane
0	Move to the Processor in Slot 0



To connect to a ControlLogix through a ControlLogix Gateway using ENBT

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley ControlLogix**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **ControlLogix**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.62
   Connection Path: **1,2,2,10.21.4.52,1,0** (**1** means going to ControlLogix and going to the back plane in PLC#1, **2** is the slot number of the EBNT module to move to in this example, **2** is the exit port of the ControlNet module, **10.21.4.52** is the IP address of entry point to the ControlNet module in PLC#2)
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **ControlLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, go to **OPC Connections > Quick Client** in the **Configure** section, on the **OPC Quick Client** page expand the **ControlLogix** folder and in the **Global** folder you can see all the tags.

Next ...

- A DH+ Example

A DH+ Example

Before we show you how to connect to an SLC through a ControlLogix Gateway using Data Highway Plus (DH+), you need to understand what a **connection path** is and how to build one, see the Connection Paths, Follow the Path, and Building the Connection Paths Entries sections.

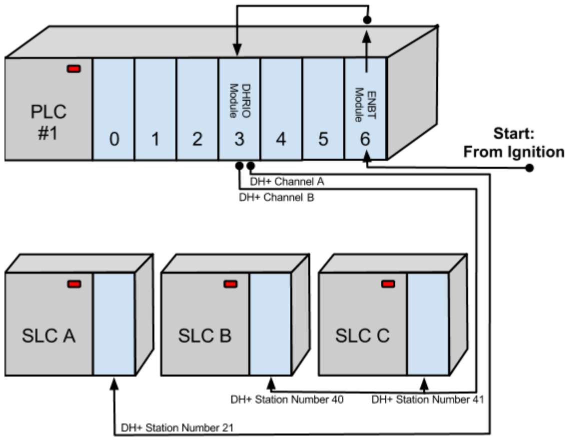
The example here, as pictured below, shows connecting through a ControlLogix Gateway to 3 different SLC 5/04 devices via DH+.

The **connection paths** are:

SLC A	**1,3,3,21**
SLC B	**1,3,2,40**
SLC C	**1,3,2,41**

The following table shows the walk-through for these numbers:

Example	
Connection Paths	**Walk-through from PLC 1 to SLC A, B, & C**
1	Move out of ENBT Module to the backplane
3	Move to DHRIO Module in Slot 3
2 (for SLC B&C) 3 (for SLC A)	Move out DH+ Channel A, B, or C
21 (for SLC A) 40 (for SLC B) 41 (for SLC C)	Move in DH+ Station Number



A DH+ Example

Watch the Video

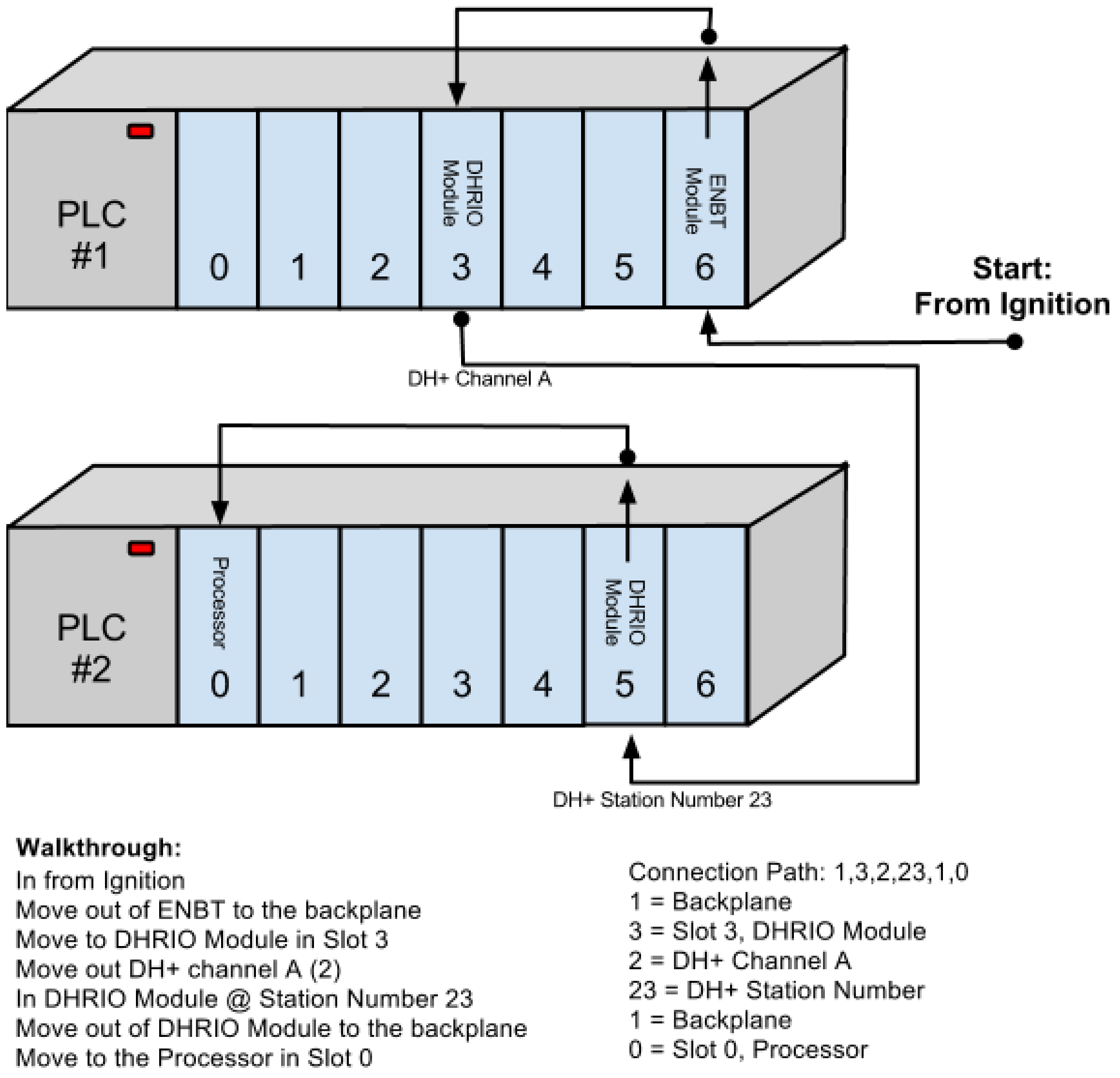
To connect to an SLC through a ControlLogix Gateway using Data Highway Plus (DH+)

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Allen-Bradley SLC**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **SLC**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.62
   Connection Path: **1,6,2,1** (**1** means going to the back plane, **6** is the slot number of the DHRIO Module to move to, **2** is the DH+ Channel A moving out of DHRIO Module, **1** is the SLC station number)

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

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

DHRIO Example



Walkthrough:
In from Ignition
Move out of ENBT to the backplane
Move to DHRIO Module in Slot 3
Move out DH+ channel A (2)
In DHRIO Module @ Station Number 23
Move out of DHRIO Module to the backplane
Move to the Processor in Slot 0

Connection Path: 1,3,2,23,1,0
1 = Backplane
3 = Slot 3, DHRIO Module
2 = DH+ Channel A
23 = DH+ Station Number
1 = Backplane
0 = Slot 0, Processor

Next ...

- add link here

Modbus

The generic Modbus driver allows the Ignition OPC-UA server to communicate with any device that supports Modbus TCP protocol.

The Modbus driver can connect directly to devices that support Ethernet communications. It can also connect to Modbus devices through the Gateway.

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

The Connecting to Modbus Device section contains step-by-step instructions on how to connect to a Modbus device.

In this section ...

Connecting to Modbus Device

To connect Ignition to a Modbus device

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

2. Scroll down and select **OPC-UA > Devices**.

3. On the **Devices** page, find the orange arrow and click on **Create new Device**.

4. On the **Add Device Step 1: Choose Type** page, select **Modbus TCP**, and click **Next**.

   There are two modules to choose from

   - **Modbus RTU over TCP**
     Which connects to devices that implement the Modbus RTU protocol over TCP.
   - **Modbus TCP**
     Which connects to devices that implement the Modbus TCP protocol.

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

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

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

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

Connecting to Modbus Device

Watch the Video

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

Modbus Drivers using Ethernet

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

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

Supported Functions Codes

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

Function Name	Code	Hex
Read Discrete Inputs	02	0x02
Read Coils	01	0x01
Write Single Coil	05	0x05
Write Multiple Coils	15	0x0F
Read Input Register	04	0x04
Read Holding Register	03	0x03
Write Single Register	06	0x06
Write Multiple registers	16	0x10
Read/Write Multiple Registers	23	0x17

Driver Properties

Connectivity	
Hostname	Is the IP Address of the Modbus device.
Port	Is the port to use when connecting to a Modbus device. The Modbus TCP port specified in the Modbus specification is 502, but it can be changed to a different port.
Communication Timeout	Is the amount of time in milliseconds to wait for a response before treating it as a failure, after sending a request to the Modbus device. When working with a Modbus RTU over TCP connection, each "device" would be an individual Modbus device on the Modbus network.
Advanced	
Max Holding Registers per Request	Is the the maximum number of Holding Registers the device can handle. Because some Modbus devices cannot handle the default of requesting 125 Holding Registers in one request, to accommodate this limitation you can change this setting.
Max Input Registers per Request	Is the the maximum number of Input Registers the device can handle. Because some Modbus devices cannot handle the default of requesting 125 Input Registers in one request, to accommodate this limitation you can change this setting.
Max Coils per Request	Is the the maximum number of Coils the device can handle. Because some Modbus devices cannot handle the default of requesting 2000 Coils in one request, to accommodate this limitation you can change this setting.
Max Discrete Inputs per Request	Is the the maximum number of Discrete Inputs the device can handle. Because some Modbus devices cannot handle the default of requesting 2000 Discrete Inputs in one request, to accommodate this limitation you can change this setting.
Reverse Word Order	When reading and writing 32bit values from/to a Modbus device, the high word comes before the low word. By checking this option, the low word comes before the high word. The Modbus specification does not include a section for reading and writing 32bit values and as a result device manufacturers have implemented both methods.
One-based Addressing	When this option is checked, the address range for each area starts at 1. If unchecked, the range starts at 0. The Modbus specification states that Modbus addresses are to be zero based. Meaning Modbus addresses start at 0 instead of 1. To read a value from Modbus address 1024, 1023 is sent to the device. When connecting to devices that do not adhere to zero based addressing, make sure this option is not selected. This will cause 1024 to be sent to the device to read Modbus address 1024.
Span Gaps	When this option is checked, it spans address gaps when optimizing requests, reducing the number of requests but increasing the amount of data requested at once. If unchecked, it does not span the address gaps.
Allow Write Multiple Registers Request	Enable or disable Modbus function code 0x10, Write Multiple Registers. Some devices may not support this function code. **Caution:** disabling this option will break the ability to write 32-bit and String values correctly to registers.
Force Multiple Register Writes	Force the use of Modbus function code 0x10, Write Multiple Registers, on write requests.
Allow Write Multiple Coils Request	Enable or disable Modbus function code 0x0F, Write Multiple Coils. Some devices may not support this function code.
Allow Read Multiple Registers Request	If disabled all registers will be read in individual read requests. Disable with caution.
Allow Read Multiple Coils	If disabled all coils will be read in individual read requests. Disable with caution.
Allow Read Multiple Discrete Inputs	If disabled all discrete inputs will be read in individual read requests. Disable with caution. Note: Function code 0x02 is always used to read Discrete Inputs, regardless what this property is set to.
	When checked forces a reconnect after 3 consecutive timeouts.

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

Next ...

- About Modbus Addressing

About Modbus Addressing

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

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

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

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

About Modbus Addressing

Watch the Video

To manually specify each address

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

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

2. In the **Tag Editor** window, as an example, you can set the following values:
   Name: Temp
   Data Type: Int4
   OPC Server: choose **Ignition OPC-UA Server** from the dropdown
   OPC Item Path: **[Modbus]HR1**, the **Modbus** device name goes in the square brackets then you give the address to PLC which in this case is the **HR** designator plus **1** as the Modbus address. The Modbus Specific Addressing section below, explains how your can construct these addresses.

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

Modbus Specific Addressing

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

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

To manually create an address for a single tag

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

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

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

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

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

Designator	Description

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

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

Description	**Holding Register Designator**	**Input Register Designator**
Float/Double		
2 consecutive Registers with Float conversion.	**HRF**	**IRF**
4 consecutive Registers with Double conversion.	**HRD**	**IRD**
Integer		
Holding Registers with 16 bit unsigned integer conversion.	**HRUS**	**IRUS**
2 consecutive Registers with 32 bit integer conversion.	**HRI**	**IRI**
2 consecutive Registers with 32 bit unsigned integer conversion.	**HRUI**	**IRUI**
4 consecutive Registers with 64 bit integer conversion.	**HRI_64**	**IRI_64**
4 consecutive Registers with 64 bit unsigned integer conversion.	**HRUI_64**	**IRUI_64**

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

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

Note: There are 2 characters for each word and the order of which character comes first is controlled by the **Reverse String Byte Order** device setting as described in the Connecting to Modbus Device section. Because two characters are stored in a word, the string length must be an even number of characters.

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

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

Unit ID

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

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

Bit-level addressing

You just append a period and the bit number you want to read and write to a bit, your Modbus device must support the `Mask Write` command, and your device documentation should specify if it does.

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

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

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

Next ...

- About Modbus Address Mapping

About Modbus Address Mapping

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

Another benefit of address mapping is that the addresses inside a device can have a different numbering scheme than the Modbus address. The Direct Automation DL240 is a perfect example of this. Address V2000, capable of holding a 16 bit integer, is Modbus Holding Register 1024. In addition, the DL240 addressing is in octal meaning there are no 8 or 9s. The sequence of addresses are: V2000, V2001, V2002, V2003, V2004, V2005, V2006, V2007, V2010, V2011.... V3777. This is not very straight forward.

On this page ...

About Modbus Address Mapping

Watch the Video

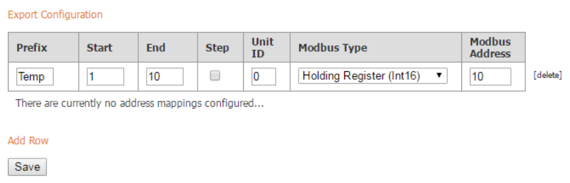
Address Mapping Properties

Name	Description
Prefix	A prefix applied to each mapped address as they appear in the OPC browser. Must compose of letters, numbers, and underscore characters. The following values are reserved, and may not be used: **HR**, **IR**, **C**, or **DI**
Start and End	Numerical values will be assigned to each mapped addresses. These properties determine the range of the numerical assignments. Follows the Prefix. The difference between these two values determines how many mapped addresses will be created.
Step	When enabled, adjacent addresses will be combined. This is commonly used to combine two words (16-bit addresses) into a double word (32-bit addresses). Please see the Floating Point or 32-bit Address Mapping for more details.
Radix	The base number of unique digits for modbus addresses. Determines what format addresses in the device are incremented and labeled (HR0, HR1,...HR9, HR10, HR11). Common values are 10 (decimal system) or 16 (hexadecimal).
Unit ID	The Unit Id for the device to use. When several Modbus devices are connected to a single IP address, the step determines which device the mapping should be applied against. A value of 0 means the first device, and should be used when only a single Modbus device is connected. See Address Mapping Multiple Devices for more details.
Modbus Type	The table each mapped address should run against, as well as the type and size of each address.
Modbus Address	The address in the device that mapping will begin at. Note that since the table has already been selected by the Modbus Type property, this number should range between 1-9999.

Simple Mapping Demonstration

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

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

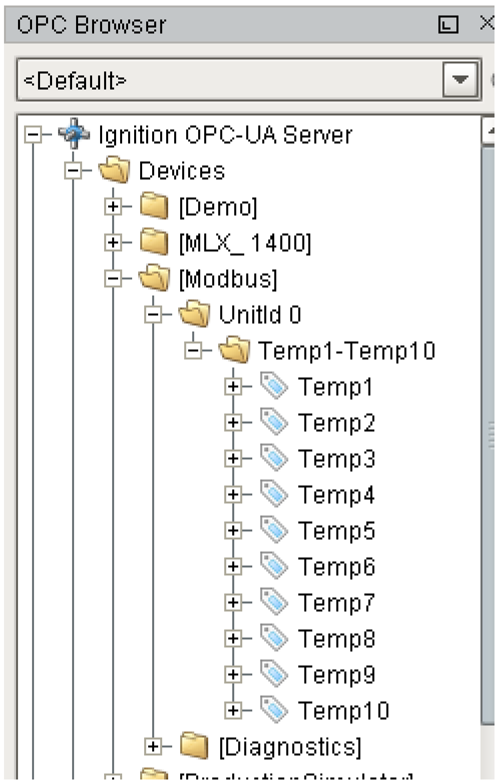


Export Configuration

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

There are currently no address mappings configured...

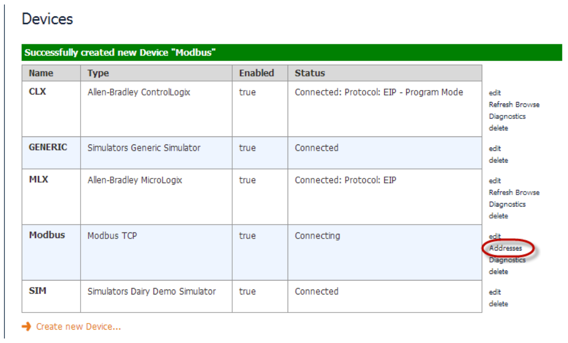
The above configuration would result in the following items appearing in the OPC Browser:



```
OPC Browser
<Default>
Ignition OPC-UA Server
  Devices
    [Demo]
    [MLX_1400]
    [Modbus]
      UnitId 0
        Temp1-Temp10
          Temp1
          Temp2
          Temp3
          Temp4
          Temp5
          Temp6
          Temp7
          Temp8
          Temp9
          Temp10
      [Diagnostics]
```

To specify the address mapping

1. Click on **Addresses** to the right of your Modbus device.



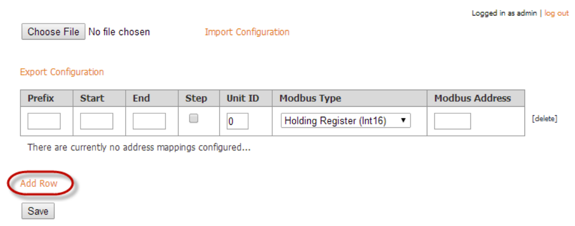
Devices

Successfully created new Device "Modbus"

Name	Type	Enabled	Status	
CLX	Allen-Bradley ControlLogix	true	Connected: Protocol: EIP - Program Mode	edit Refresh Browse Diagnostics delete
GENERIC	Simulators Generic Simulator	true	Connected	edit delete
MLX	Allen-Bradley MicroLogix	true	Connected: Protocol: EIP	edit Refresh Browse Diagnostics delete
Modbus	Modbus TCP	true	Connecting	edit Addresses Diagnostics delete
SIM	Simulators Dairy Demo Simulator	true	Connected	edit delete

Create new Device...

2. Click on **Add Row**.



Logged in as admin | log out

Choose File No file chosen   Import Configuration

Export Configuration

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

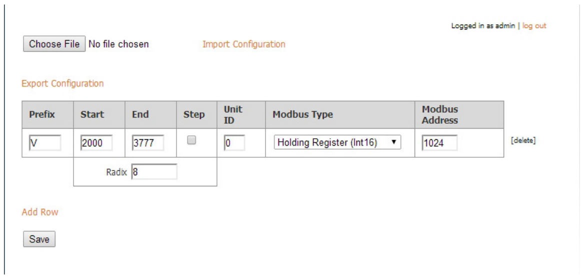
There are currently no address mappings configured...

Add Row

Save

3. Enter the mapping as follows:

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


Logged in as admin | log out

Choose File No file chosen   Import Configuration

Export Configuration

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

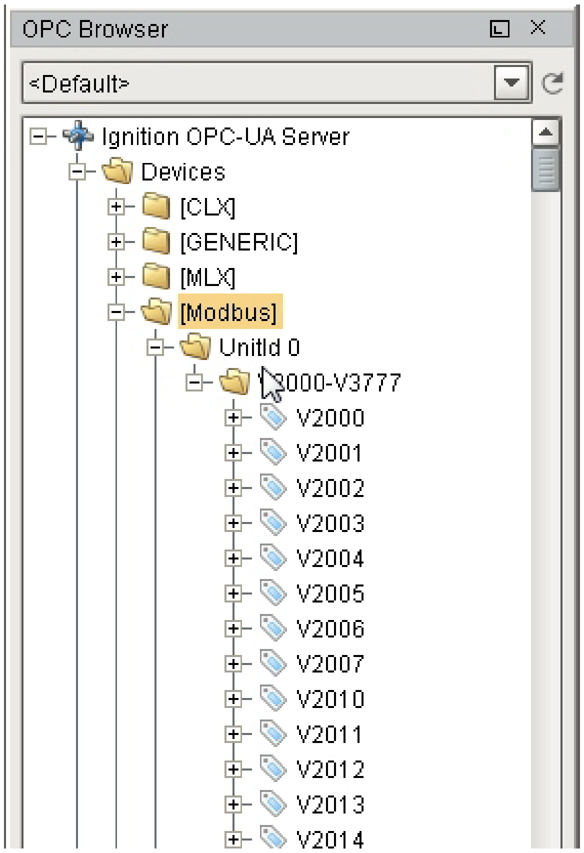
Add Row

Save

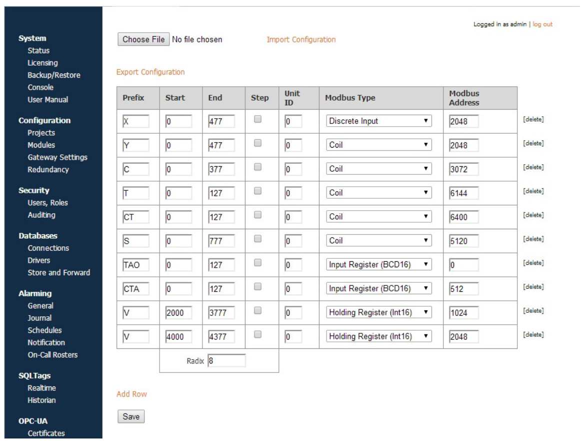
These settings map the **Modbus address range V2000 to V3777** in octal to **Modbus Holding Register addresses 1024 to 2047**.
Note: The mappings for string data types cannot be entered. Strings can only be read or written using Modbus Specific Addressing.

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

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



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



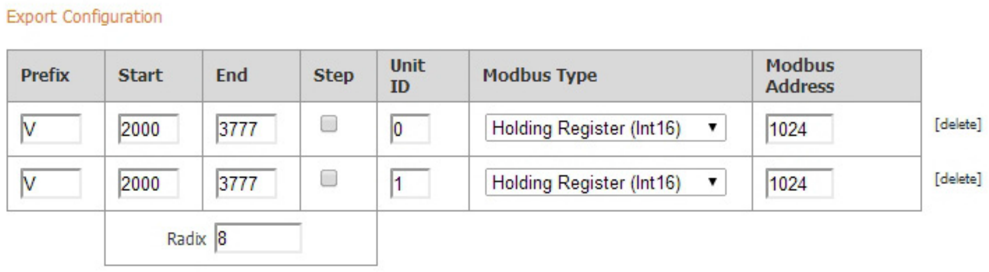
No file chosen

Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address	
X	0	477	☐	0	Discrete Input	2048	[delete]
Y	0	477	☐	0	Coil	2048	[delete]
C	0	377	☐	0	Coil	3072	[delete]
T	0	127	☐	0	Coil	6144	[delete]
CT	0	127	☐	0	Coil	6400	[delete]
S	0	777	☐	0	Coil	5120	[delete]
TAO	0	127	☐	0	Input Register (BCD16)	0	[delete]
CTA	0	127	☐	0	Input Register (BCD16)	512	[delete]
V	2000	3777	☐	0	Holding Register (Int16)	1024	[delete]
V	4000	4377	☐	0	Holding Register (Int16)	2048	[delete]
	Radix 8						

Address Mapping Multiple Devices

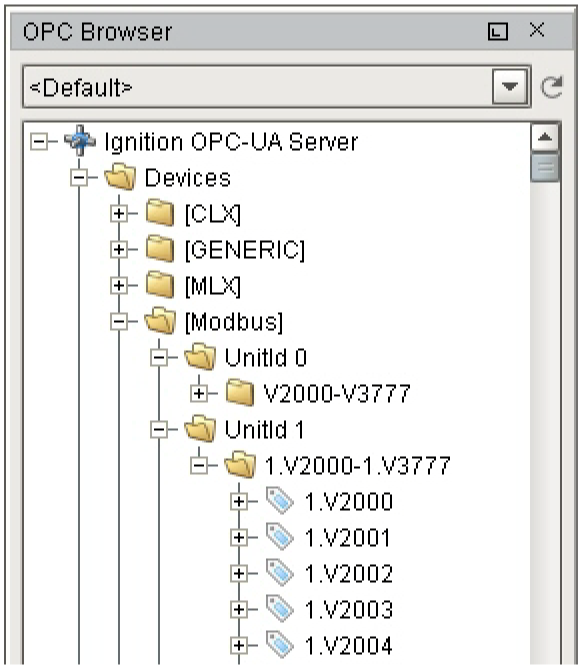
It is not recommended to communicate to multiple Modbus devices through a Modbus Gateway where Gateway has the same address. Therefore, do not add multiple Modbus devices with the same IP address.

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



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

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



Floating Point or 32-bit Address Mapping

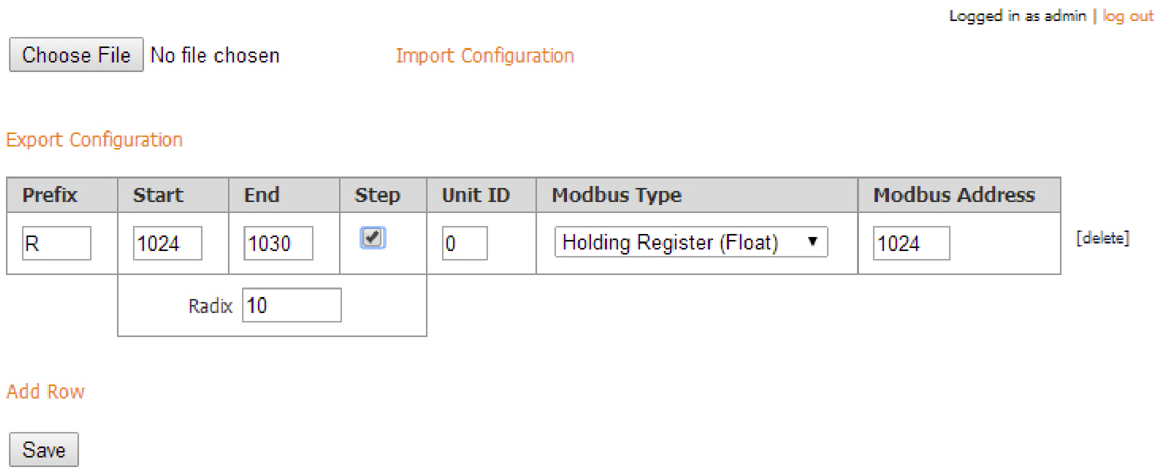
Modbus only supports reading and writing to memory types of bits and 16-bit words. This is not very useful when reading from or writing to float point or 32-bit integers.

To workaround this problem, the Modbus driver is designed to read 2 consecutive 16-bit words and encode it into the desired data type.

To map float point addresses

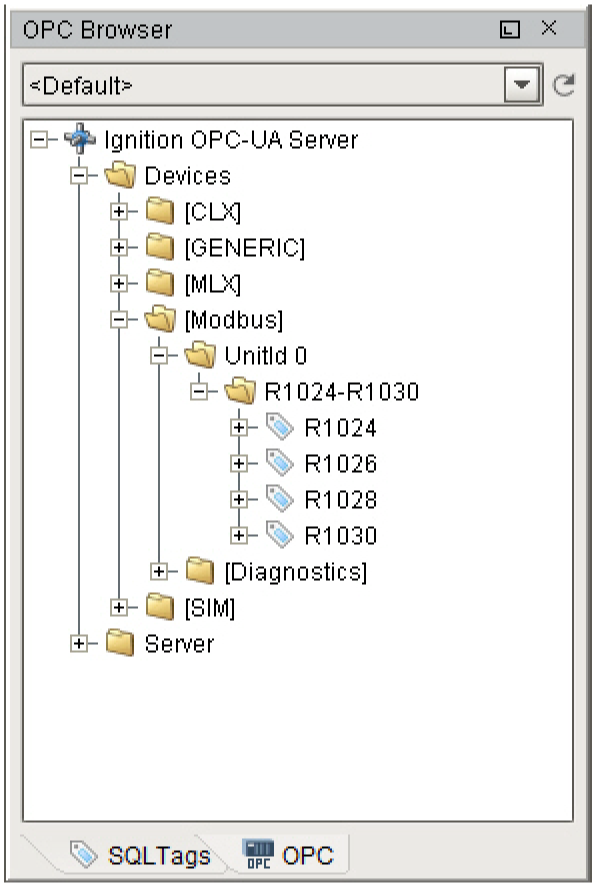
The Modbus address mapping below shows how to map float point addresses starting at 1024 and ending at 1030. With the box in the **Step** column checked, the addresses on the Ignition side will index by 2. In this case, R1024, R1026, R1028 and R1030 will be created.

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



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

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



Import / Export Address Mapping

The mapping configuration can be exported to a comma separated values (CSV) file. The CSV file can later be imported in other Ignition installations or similar devices.

You can find a few examples of CSV files on our website. To see the examples, go to:

http://inductiveautomation.com/downloads/extras

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

Next ...

- add link here

Siemens

The Siemens drivers in Ignition support connections to S7-300, S7-400, and S7-1200 PLCs via TCP/IP using the S7 protocol. For step-by-step instructions to connect to a Siemens device, see Connecting to S7 Devices.

In this section ...

Connecting to S7 Devices

To connect Ignition to a Siemen S7-1500, S7-1200, S7-400, or S7-300 device

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

2. Scroll down and select **OPC-UA > Devices**.

3. On the **Devices** page, find the orange arrow and click on **Create new Device**.

4. On the **Add Device Step 1: Choose Type** page, select **Siemens S7-1200**, and click **Next**.

   There are four modules to choose from

   - **Siemens S7-1500**
     Which connects to Siemens S7-1500 PLCs over Ethernet.
   - **Siemens S7-1200**
     Which connects to Siemens S7-1200 PLCs over Ethernet.
   - **Siemens S7-300**
     Which connects to Siemens S7-300 PLCs over Ethernet.
   - **Siemens S7-400**
     Which connects to Siemens S7-400 PLCs over Ethernet.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **S71200**
   Hostname: type the **IP address**, for example 10.20.4.71

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

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

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

Connecting to S7 Devices

Watch the Video

Next ...

- About Siemens Addressing

About Siemens Addressing

INDUCTIVE UNIVERSITY

About Siemens Addressing

Watch the Video

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

To manually specify each address

1. From the Designer, in the **Tag Browser**, right-click on **Tags** folder, and then go to **New Tag > OPC Tag**.
2. In the **Tag Editor** window, as an example, you can set the following values:
   Name: **Tag**
   Data Type: **Int4**
   OPC Server: choose **Ignition OPC-UA Server** from the dropdown
   OPC Item Path: **[S71200]IW0**, the **S71200** device name goes in the square brackets then you give the address to PLC which in this example is **IW0** (Word at Offset 0 in the Inputs area). The Configuring Siemens Addressing section below, explains how your can construct these addresses.
3. Click **OK**.
   Now you can see the **Temp** tag in the **Tag Browser**.

Configuring Siemens Addressing

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

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

	Area Syntax	
DataBlocks	DBn,	
Inputs	I	
Outputs	Q	
Flags	M	
Timers	T	
Counters	C	
	Data Type Syntax	**Signedness**
Bit	X	N/A
Byte	B	Unsigned
Char	C	Signed
Word	W	Unsigned
Int	I	Signed
DWord	D	Unsigned
DInt	DI	Signed
Real	REAL	Signed
String	STRING or STRING.LEN	N/A

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

Examples	

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

Offsets

It is important to note that offsets are absolute. **IW0** and **IW1** share a byte. To get 2 consecutive, non-overlapping words you need to address **IW0** and **IW2**.

Bits

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

Strings

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

Timers

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

Counters

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

Next ...

- add link here

UDP and TCP Driver

The TCP driver allows Ignition to communicate to barcode scanners, scales, and more. They are strictly passive listeners and are configured to connect to one or more ports on a given IP address and any ASCII characters that are brought in as tags. For step-by-step instructions to connect to a TCP Driver, see Connecting to TCP Device.

UDP and TCP Drivers

The UDP and TCP drivers are strictly passive listeners. These drivers are configured to connect and **listen** to one or more ports on a given IP address. Rules are configured that dictate how the incoming data is interpreted.

Structure in the Address Space

A device using the UDP or TCP driver appears in the **Devices** folder of the OPC-UA server with the name it was configured to use. Browsing the device will yield one folder per port configured to listen on. Browsing the port folder will yield 1 variable node containing the entire message received as well as an additional variable node per field configured. A device configured with a field count of 4 would have 5 nodes total -1 for the message and 4 for the fields.

In this section ...

Connecting to TCP Device

To connect to a barcode scanner or scale

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

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, scroll all the way down and select **TCP Driver**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **Scale**, the name you specify here will appear under **Devices** folder on the **Quick Client** page in the **Gateway.**
   Port(s): **12345**, as an example
   Hostname: type the **IP address**, for example 10.20.6.108
6. You can check the box for **Show advanced properties?** to see the additional settings, but you can keep all the defaults.
7. Click **Create New Device**.
   The **Devices** page is displayed showing the **Scale** device is successfully created and added to Ignition. The **Status** will show as **1/1 Connected**.
8. Go to the **OPC Connections > Quick Client** in the **Configure** section of the Gateway, under the **Devices>[Scale]>12345** folder you will see the **Last Receive Time** and the **Message** folder s.
   Next to each tag, under the **Action** column, you will see `[s][r][w].`
9. Click on `[s]` which means **Subscription**. You will be able to see the **Value** of the tag displayed on this **OPC Quick Client** page.

INDUCTIVE UNIVERSITY

Connecting to TCP Device

Watch the Video

Device Properties

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

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

	BAD_CONFIG_ERROR.
Field Delimiter	This is the character(s) that are used as field delimiters. For example, the message **a|b|c|d** with a field delimiter of **|** would be split into four fields: **a**, **b**, **c**, and **d**. The field count would have to be set at 4.

Next ...

- add link here

DNP3

The DNP3 driver can connect directly through the Gateway to devices that support Ethernet communications. The DNP3 section contains step-by-step instructions for connecting using this driver.

In this section ...

Connecting to DNP3 Devices

The DNP3 module allows the Ignition OPC-UA server to communicate with other devices that support the DNP3 protocol (DNP3 outstations).

On this page...

To connect Ignition to a DNP3 device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **DNP3 Driver**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:

   Name: **DNP3**
   Hostname: type the IP address, for example 10.20.8.51
6. You can check the box for **Show advanced properties?** to see the additional settings, but you can keep all the defaults.
7. Click **Create New Device**.
   The **Devices** page is displayed showing the **DNP3** device is successfully created and added to Ignition. The **Status** will show as Disconnected and then Connected or Idle, depending on the status of the device.

INDUCTIVE UNIVERSITY

Connecting to DNP3 Devices

Watch the Video

Connection Settings

General	
Name	The name of this DNP3 device connection.
Description	Device connection description (optional). Can be used to provide any useful information / comments about this connection.
Enabled	If True (checked), the connection is enabled; if False (unchecked), the connection is disabled.
Main	
Hostname	The IP Address of the Modbus device.
Port	The port to use when connecting to a DNP3 device. The default port is 20000.
Source Address	The address of the master station, default is 3.
Destination Address	The address of the outstation, default is 4.
Integrity Poll Interval	The interval at which to perform an integrity poll, in millis, default is 3,600,000.
Direct Operate Enabled	When true, the Direct-Operate function code is used on a write, otherwise Select-Operate is used, default is true.
	When true, the outstation may send unsolicited messages for Class 1, 2, and 3 data, default is false.

Unsolicited Messages Enabled	
Advanced	
Message Fragment Size	The maximum size of a message fragment in the application layer, default is 249.
Message Timeout	The amount of time to wait for a message response from the outstation, default is 5,000.
Retries	The number of retries on a message timeout, default is 0.
Link Layer Confirmation	When true, a link layer confirmation will be required from the outstation when sending messages, default is false

Notes

- **Source Address and Destination Address**: These addresses are assigned to the computers and should be the same across all settings, because of this the settings in Ignition and the settings in the device are the opposite of each other. For example, if the device is configured with an address of 4 and Ignition has an address of 3:
  - the settings in Ignition should have the Source Address set to 3 and the Destination Address set to 4.
  - the settings in the Device should have the Source Address set to 4 and the Destination Address set to 3.
- **Unsolicited messages enabled** property: setting this property to True (checking the box) allows the outstation to send unsolicited messages to Ignition. This means that Ignition will connect to the outstation, but not request any data from it. Ignition waits for the outstation to send data. Not all devices support this option; those that do need to be configured to use it. Please refer to your device's documentation for more information.

Internal Indicators

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

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

Terminology

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

Next...

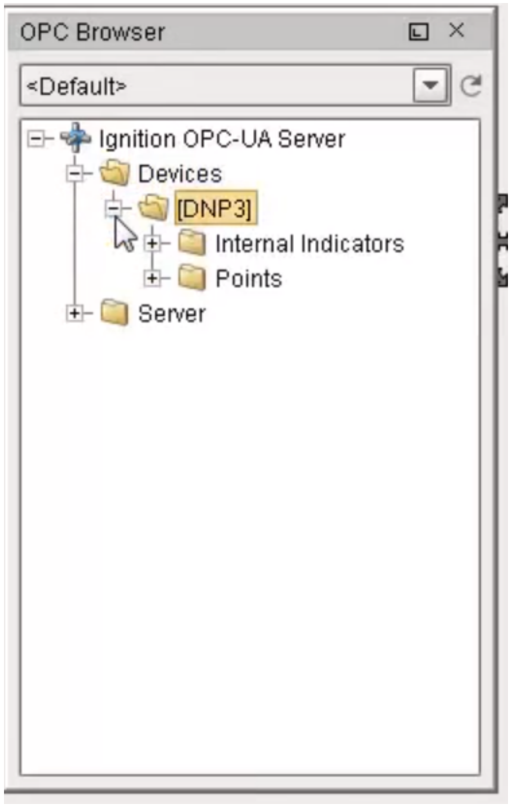
About DNP3 Addressing

About DNP3 Addressing

Browsing DNP3 Points

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

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

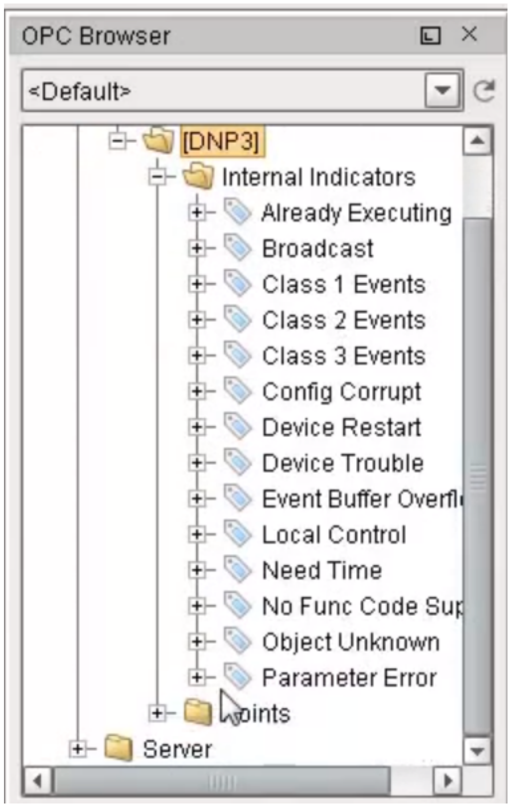


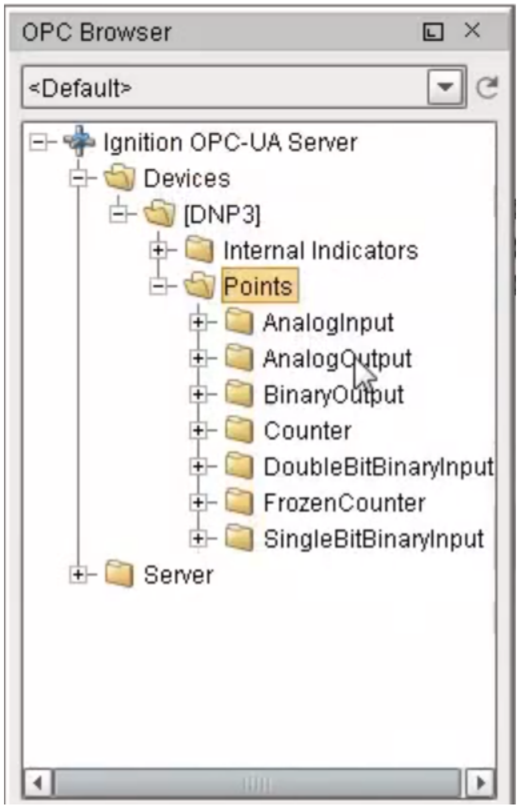
On this page...

INDUCTIVE UNIVERSIT

About DNP3 Addressing

Watch the Video





Point Types

Type Name	Group	Supported Variations
SinglBitBinaryInput	1	1 - Packed format 2 - With flags

DoubleBitBinaryInput	3	1 - Packed format 2 - With flags
BinaryOutput	10	1 - Packed format 2 - With flags
Counter	20	1 - 32-bit with flags 2 - 16-bit with flags 5 - 32-bit 6 - 16-bit
FrozenCounter	21	1 - 32-bit with flags 2 - 16-bit with flags 5 - 32-bit with flags and time 6 - 16-bit with flags and time 9 - 32-bit 10 - 16-bit
AnalogInput	30	1 - 32-bit with flags 2 - 16-bit with flags 3 - 32-bit 4 - 16-bit 5 - Float with flags 6 - Double with flags
FrozenAnalogInput	31	1 - 32-bit with flags 2 - 16-bit with flags 3 - 32-bit with time of freeze 4 - 16-bit with time of freeze 5 - 32-bit 6 - 16-bit 7 - Float with flags 8 - Double with flags
AnalogOutput	40	1 - 32-bit with flags 2 - 16-bit with flags 3 - Float with flags 4 - Double with flags
OctetString	110	0 - 255

Aliased Points

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

Point Address	The group, variation, and index that fully describe a point. A full address consists of all three parts: • Group – An integer prefixed with **`g`**. For example, **`g40`** • Variation – An integer prefixed with **`v`**. For example, **`v2`** • Index – An integer prefixed with **`i`**. For example, **`i5`**

	Example: **`g30v1i20`**
Path	A “/” separated folder hierarchy in which to create the aliased point. Example: **`Facility1/Voltage`**
Description	A user-defined description of the point mapping.

Omron

To Connect Ignition to an Omron Device

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

2. Scroll down and select **OPC-UA > Devices**.

3. On the **Devices** page, find the orange arrow and click on **Create new Device**.

4. On the **Add Device Step 1: Choose Type** page, select **Omron Driver**, and click **Next**.

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

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

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

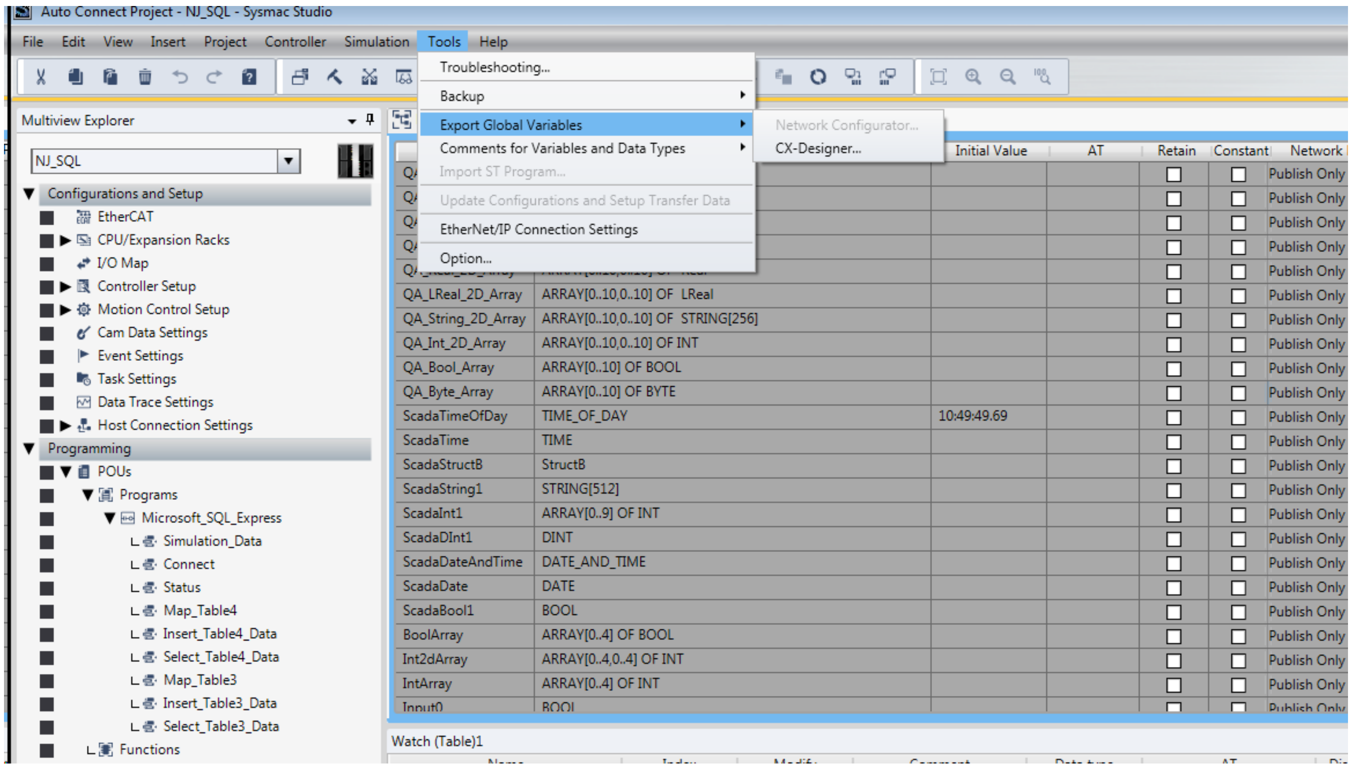
7. On the **Devices** page, click the tags link next to the newly created device.
   The **Manage Tags** page is displayed, allowing you to configure which variables in the device will show up as tags in Ignition.

Gateway Settings

General	
Name	The name of the device connection, as Ignition will reference it.
Enabled	Determines whether or not to enable this device connection.
Main	
Hostname	The hostname or IP address of the device.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.
Concurrency	The number of concurrently issued requests allowed. There is a 1:1 correlation between concurrency and the number of CIP connections used.
Advanced	
Connection Size	The CIP connection size to use. The default (and maximum) is 1,994 bytes.
Slot Number	The slot number in the backplane in which the CPU is located.

Exporting from the Device

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

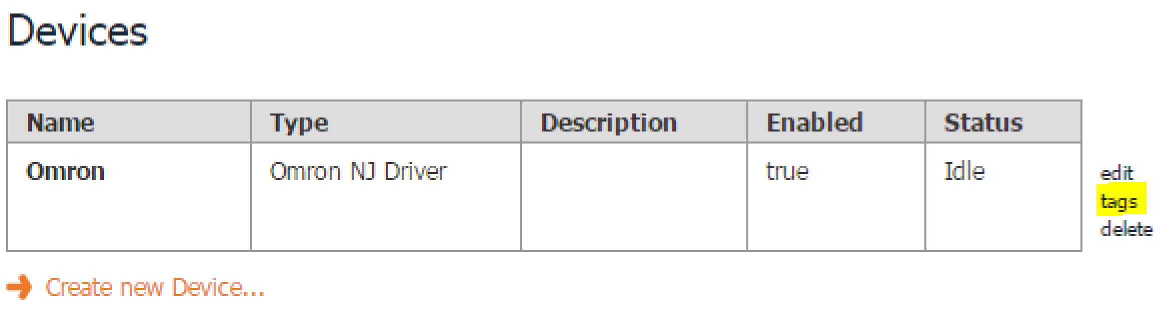


		Initial Value	AT	Retain	Constant	Network
QA_LReal_2D_Array	ARRAY[0..10,0..10] OF LReal			☐	☐	Publish Only
QA_String_2D_Array	ARRAY[0..10,0..10] OF STRING[256]			☐	☐	Publish Only
QA_Int_2D_Array	ARRAY[0..10,0..10] OF INT			☐	☐	Publish Only
QA_Bool_Array	ARRAY[0..10] OF BOOL			☐	☐	Publish Only
QA_Byte_Array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

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

Managing Tags

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

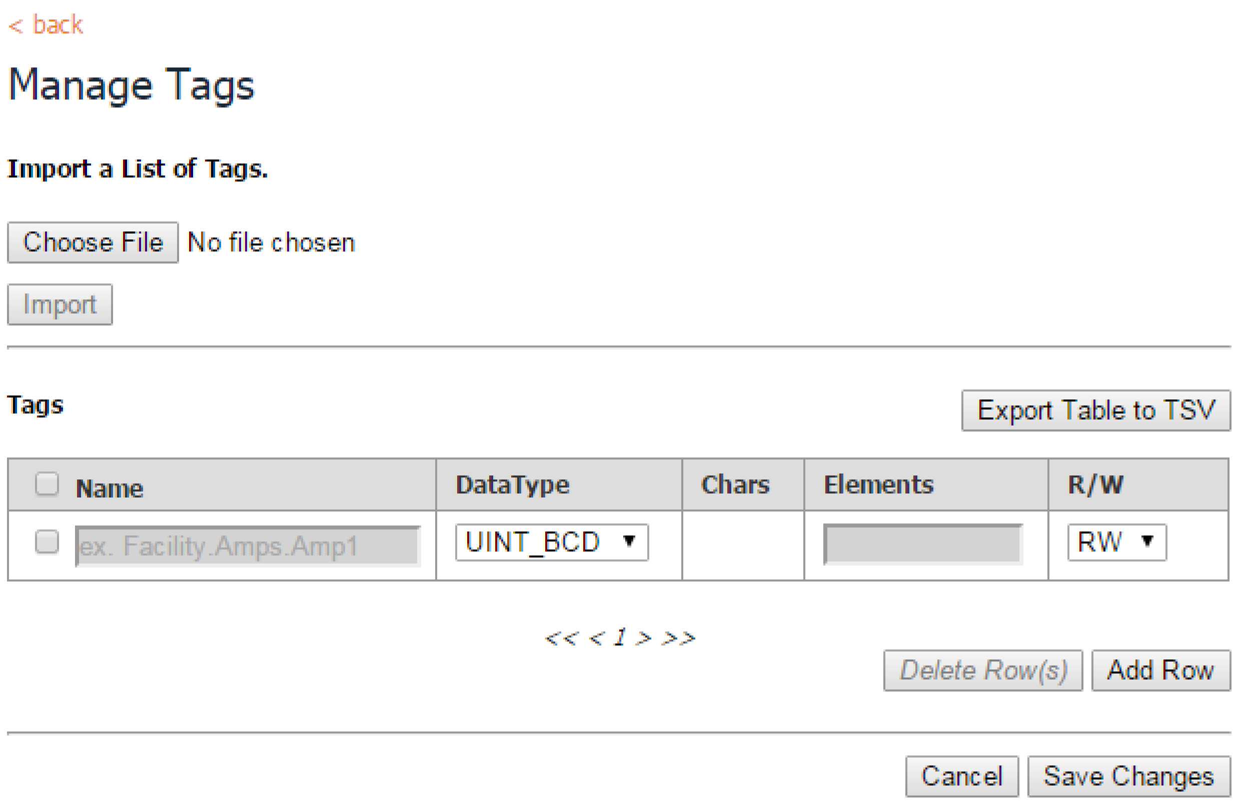

Devices

Name	Type	Description	Enabled	Status	
Omron	Omron NJ Driver		true	Idle	edit tags delete

Importing Tags

Once on the *Manage Tags* page, you can manually enter the tags, or import them from a tab-separated file.

When importing, first choose a file, then click the *Import* button. The default option when importing is to replace the *Tags* table with tags from the import. Select the append option to append tags to the table.


< back

Manage Tags

Import a List of Tags.

Choose File No file chosen

Import

Tags

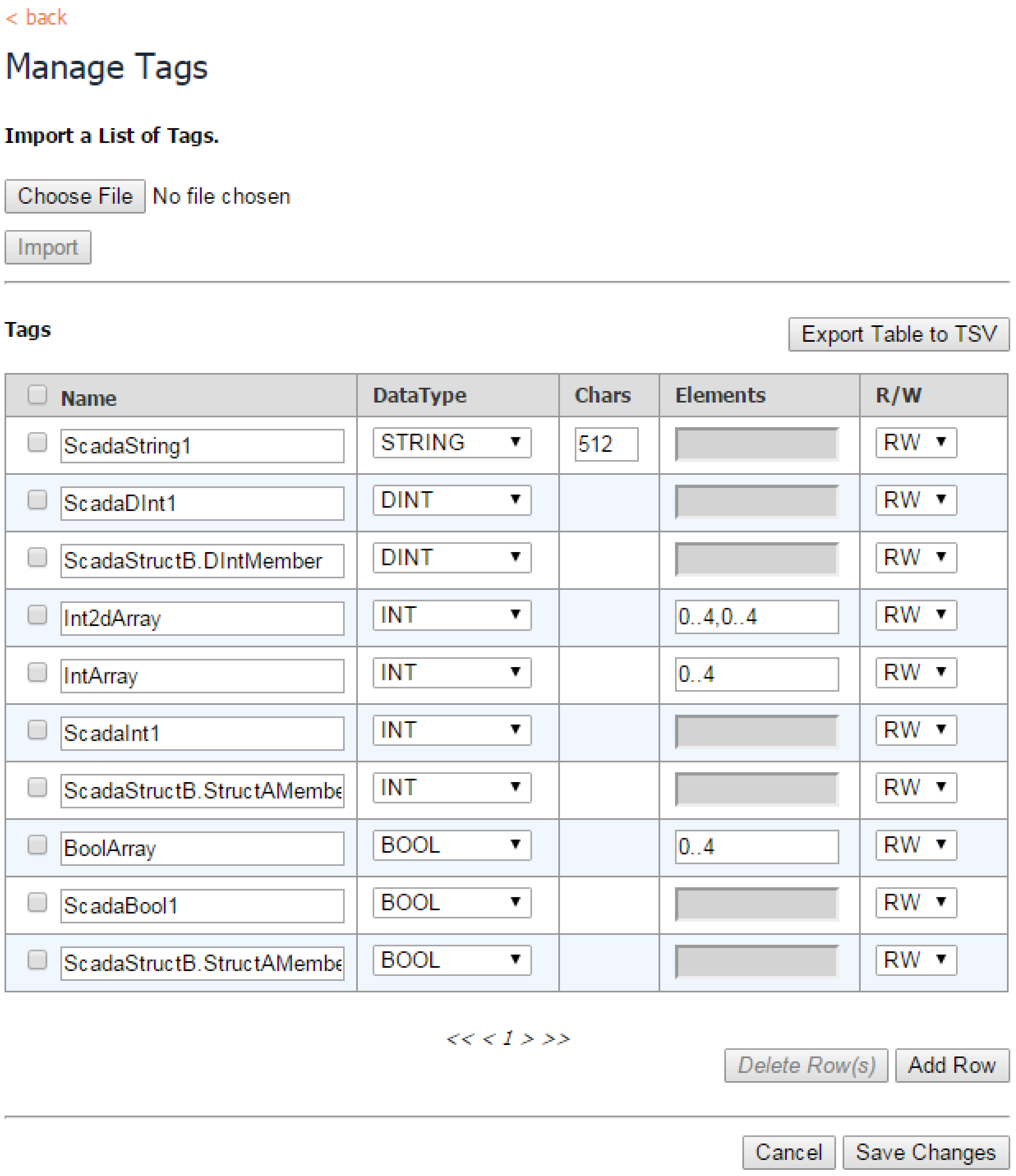
Export Table to TSV

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

<< < 1 > >>

Delete Row(s) Add Row

Cancel Save Changes


< back

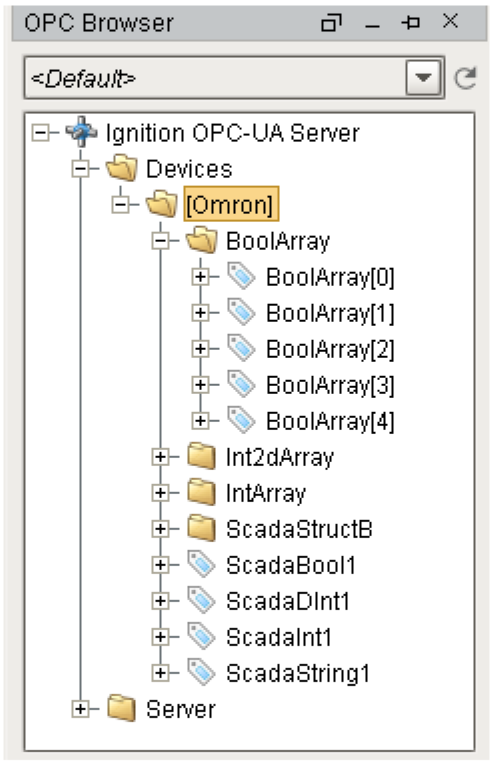
Manage Tags

Import a List of Tags.

Tags

Name	DataType	Chars	Elements	R/W
ScadaString1	STRING	512		RW
ScadaDInt1	DINT			RW
ScadaStructB.DIntMember	DINT			RW
Int2dArray	INT		0..4,0..4	RW
IntArray	INT		0..4	RW
ScadaInt1	INT			RW
ScadaStructB.StructAMembe	INT			RW
BoolArray	BOOL		0..4	RW
ScadaBool1	BOOL			RW
ScadaStructB.StructAMembe	BOOL			RW

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


OPC Browser

<Default>

- Ignition OPC-UA Server
  - Devices
    - [Omron]
      - BoolArray
        - BoolArray[0]
        - BoolArray[1]
        - BoolArray[2]
        - BoolArray[3]
        - BoolArray[4]
      - Int2dArray
      - IntArray
      - ScadaStructB
      - ScadaBool1
      - ScadaDInt1
      - ScadaInt1
      - ScadaString1
  - Server

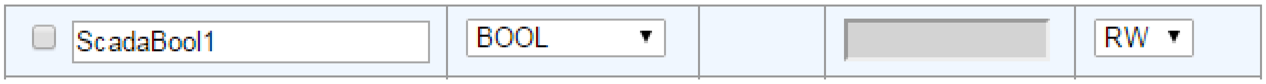
Addressing

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

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

Scalars

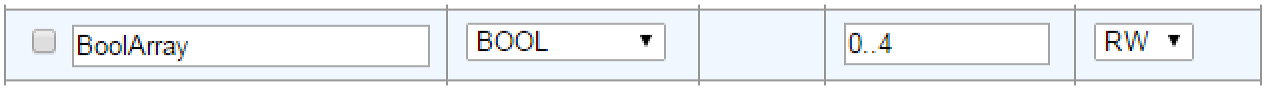
Leaving the *Elements* column blank will result in a scalar tag. When reading from the device, only one element will be requested.



☐	ScadaBool1	BOOL			RW

Arrays

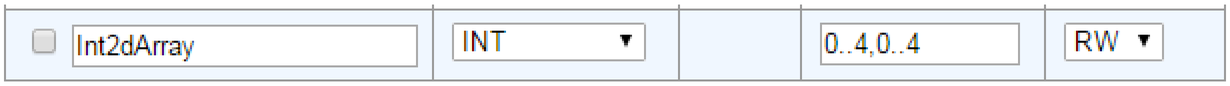
Specify the number of elements in an array in the form of **0..N** . The initial index 0 is always included, so an array mapped with 0..4 elements is a 5 element array.



☐	BoolArray	BOOL		0..4	RW

Multi-Dimensional Arrays

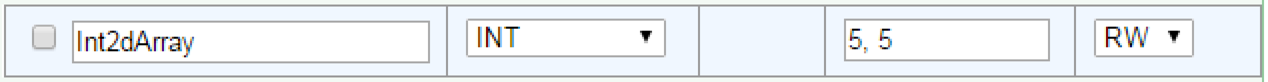
Multi-dimensional arrays are specified in the same way as arrays, with each group of indices separated with a comma.



☐	Int2dArray	INT		0..4,0..4	RW

Optional Format

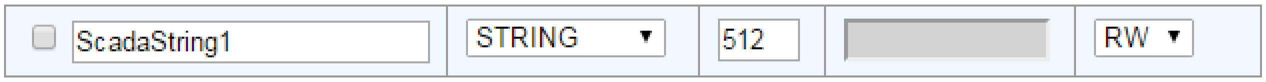
Array elements may also be specified with a single number equaling the total number of elements.



☐	Int2dArray	INT ▼		5, 5	RW ▼

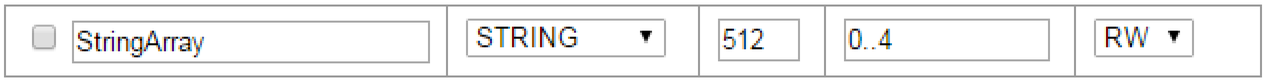
Strings

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



☐	ScadaString1	STRING ▼	512		RW ▼

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



☐	StringArray	STRING ▼	512	0..4	RW ▼

Third Party OPC Servers

Ignition can easily and quickly connect to third party OPC servers via OPC-UA or OPC-DA (using the OPC-COM module) for devices that do not have a supported driver. For step-by-step instructions to connect to a Siemens device, see Connecting to Kepware OPC-UA.

In this section ...

Connecting to Kepware OPC-UA

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

INDUCTIVE UNIVERSITY

Connecting to Kepware OPC-UA

Watch the Video

To connect to KEPServerEX from Ignition using OPC-UA

1. In the **Configure** section of the Gateway, go to **OPC Connections > Servers**.
   The OPC Server Connection page is displayed showing the OPC-UA servers you Ignition is connected to.
2. Find the orange arrow and click on **Create new OPC Server Connection…**.
   The **Add OPC Server Connection Step 1: Choose Type** page is displayed.

3. Choose **OPC-UA** as the connection type and click **Next**.

The next steps depends on the version of Ignition being used.

Ignition 7.8 and up

1. On the **Discover OPC-UA Endpoints** page, type in the endpoint of the OPC-UA server Ignition should connect to. The format should be as follows:

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

   #Examples

   #If KepServer is installed on the same server as Ignition and uses
   the default port, the following endpoint could be used:
   opc.tcp://localhost:49320

   #If KepServer is installed on a different server from Ignition, then
   the IP address would be used:
   opc.tcp://10.1.1.10:49320
   ```

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

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

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

3. On the **New OPC UA Connection Settings** page, give the connection a name and click the **Create New OPC Server Connection** button. A Username and Password do not need to be given, and should be left blank.
4. The connection will appear as **Faulted**. This is expected because KEPServerEX is denying access to the Ignition OPC-UA Client. The next step is to have KEPServerEX trust the Ignition OPC-UA Client.

   Right-click on the **KEPServerEX** icon on the desktop KEPServerEx is installed on and from the menu select **OPC UA Configuration**.
   The **OPC UA Configuration Manager** is will appear.
5. On the **OPC UA Configuration Manager** window, go to the **Trusted Clients** tab, click on **Ignition OPC-UA Client**, click the **Trust** button, and click **Close**.
   Now the **OPC Server Connections** page shows the **Status** of **Kepware** to be **Connected**.

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

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

Ignition 7.7 and prior

1. In the **Configure** section of the Gateway, go to **OPC Connections > Servers**.
   The OPC Server Connection page is displayed showing the OPC-UA servers you Ignition is connected to.

2. Find the orange arrow and click on **Create new OPC Server Connection…**.
   The **Add OPC Server Connection Step 1: Choose Type** page is displayed.

3. Choose **OPC-UA** as the connection type and click **Next**.

4. On the **New OPC Server Connection** page, leave all the default values and type in the following fields:
   Name: **Kepware**, the name you specify here will appear under **Devices** folder on the **Quick Client** page in the **Gateway.**
   Host: **localhost**, if the Kepware server is hosted on your own machine otherwise use the actual IP address of the machine
   Port: **49320**, which is the default port

5. Right-click on the **KEPServerEX** icon on your desktop and from the menu select **OPC UA Configuration**.
   The **OPC UA Configuration Manager** is displayed.

6. Go to the **Server Endpoints** tab, check the URL to ensure it shows your localhost and the 49320 port, and click **Close**.

7. Go back to the **New OPC Server Connection** page, make sure you have the following default settings:
   Security Policy: **Basic128Rsa15**
   Message Security Mode: **SignAndEncrypt**

8. Leave the **Password** fields empty and click **Create New OPC Server Connection**.
   The **OPC Server Connections** page shows the **Status** of **Kepware** to be **Faulted** and not connected. The reason for this is that Kepware is denying access to the Ignition OPC-UA Client.

9. To make the Ignition OPC-UA Client a trusted client, go back to the **KEPServerEX** icon on your desktop and from the menu select **OPC UA Configuration**.

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

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

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

Other UA Servers

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

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

To download a client certificate manually

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

2. Select **OPC-UA > Certificate** from the left side of the page.
   The **Manage Certificate** page is displayed.

3. In the **This Gateway** tab, click the download link under **Ignition OPC-UA Client**, and save the certificate somewhere to disk.
   This certificate is then supplied to your third-party OPC server in a way specific to that server. For more information, check respective server's documentation.

Next ...

- add link here

Vision Windows

Every Ignition project contains a collection of Windows. The windows are the fundamental building blocks for projects using the Vision module.

In this section ...

About Vision Windows

Windows are the top-level design units for Vision projects. Windows are made up of components and come in different types such as main, popup, and docked windows.

Creating Windows

Creating windows is as easy as pressing the **New Window** button in the toolbar, or by navigating to the **File > New > Window** menu. There are three types of windows you can create: a **main** window, a **popup** window, or a **docked** window. These three windows are described in the Window Types page.

To create a Window

Lets create a window so we can use our tags for some basic status and control.

1. Click on the **New Window** icon in the toolbar or use the **File > New > Main Window**.
   Tags are used in windows to power property bindings on components. The easiest way to make some components that are bound to tags is to simply drag and drop some tags onto your window.
2. Drag a tag onto a window.
   You'll get a popup menu asking you what kind of component to make. You can Display the tag with some components, and control the tag with other components.
3. Drag a few more tags onto the screen to experiment with the different options.
4. As you're editing your project, you can click the **Save** icon on the toolbar to save your changes.
   In Ignition, you're not editing a file, your Designer is linked to the Ignition Gateway. When you click save, the project is saved back on the central Gateway. All running Clients are notified that there is a new version of the project available.

Naming and Renaming a Window

In the Project Browser, right-click on the window and choose **Rename** or press **F2**. Enter a new name for the window. The name must be unique in its folder.

Organizing Windows

You can also create folders to organize your windows. A window's name must be unique among the windows in its folder. A window's name and folder path is very important - it will be how other windows reference it.

Accessing windows from the Client

In the Project Browser, right-click on the window and select the check box next to **"About" Window**. Once checked, the window will be accessible from the **Help > About This Application** menu in the Client, which opens the appropriate window.

Adding Notes on Windows

Through the right-click menu on a window in the **Project Browser,** you can access the window's **Notes**. This free-form text field is provided to let the designer document the purpose and any technical information about how the window works.

Importing and Exporting a Window

You can import and export windows to external files by using the right-click menu in the **Project Browser**. Simply select the windows in the export wizard that you'd like to export, and choose a path for the resulting `*.vwin` file.

Setting the Window's Security

You can configure security settings that control who can and who can't open a window. While the window is open, right-click on a window in the **Project Browser** and select **Security.** Window Security is configured the same way that Component Security is configured.

In this section ...

Anatomy of a Window

Name and Path

Windows are the top-level unit of design for Vision projects. A window is identified by its path, which is the name of all its parent folders plus its name, with forward slashes (/) in between. For example, the path to a window in the top level called **MainWindow** would simply be its name, whereas the path to a window named **UserOptions** under a folder called **OptionsWindows** would be: **OptionsWindows /UserOptions**.

Root Container

Inside a window is always the Root Container. This is a normal container component except that it cannot be deleted or resized - its size is always set to fill the entire window. The Root Container is where you will place all of your components in the window.

Titlebar and Border

A window may display a titlebar and/or a border. The titlebar allows the user to drag the window around, and houses the window's close and maximize/restore buttons. The border of a window can be used to resize the window when it is floating or docked. Whether on not the titlebar and border are displayed depends on the values of the window's titlebar and border display policy properties, and its current state. Commonly, a window will display both a titlebar and border when it is floating, but only a titlebar when maximized. It is often desirable to remove titlebars and borders on maximized windows.

INDUCTIVE UNIVERSIT

Anatomy of a Window

Watch the Video

Next ...

- Window Types
- Window Properties

Window Types

Windows come in three flavors: main screens, popups, and docked windows.

INDUCTIVE UNIVERSITY

Window Types

Watch the Video

Main Windows

A **Main Window** is one that is set to start maximized, and has its border and titlebar display policies set to When Not Maximized or Never. This will make the window take up all available space (minus space used by any "docked" windows). This makes the window act much like a typical "HMI screen." There can by many main windows in a project, but only one should be open at any time.

Popup Windows

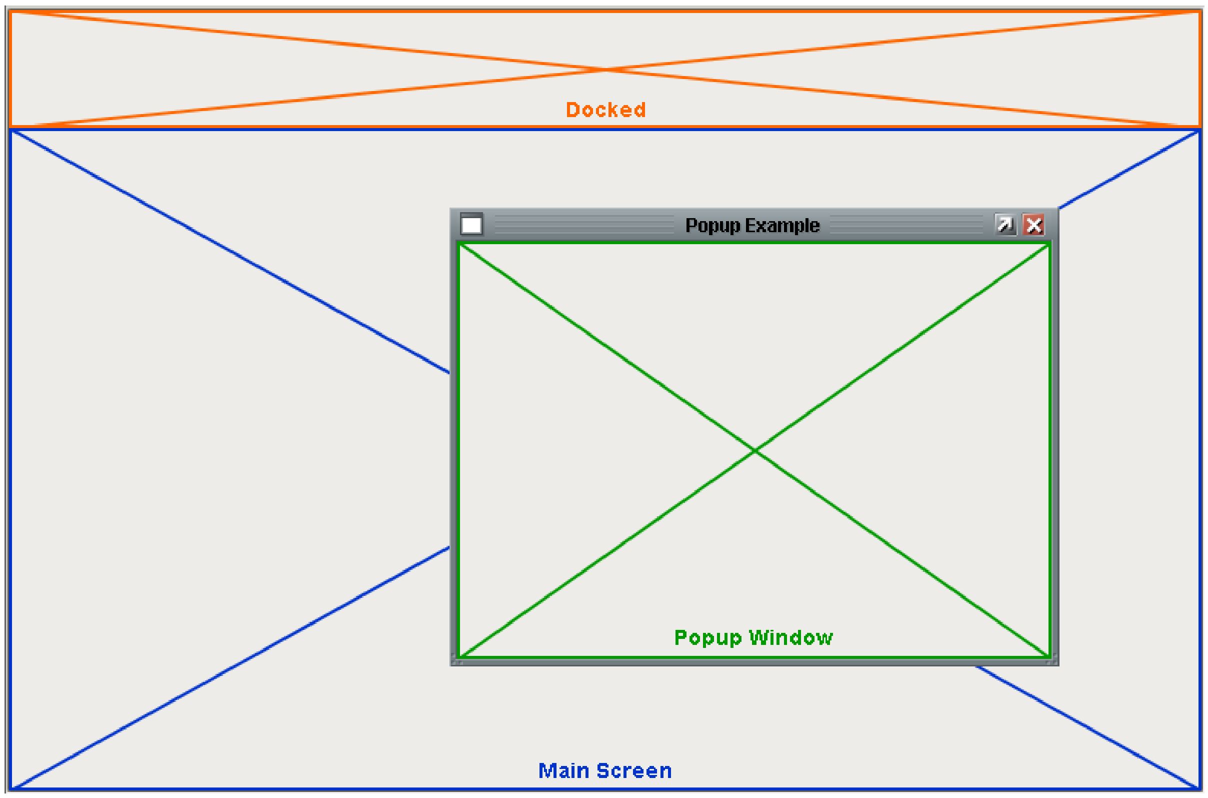
A **Popup Window** is a window whose Dock Position is set to Floating and is not maximized. Its border and titlebar display policies are usually set to When Not Maximized or Always, so that they can be manipulated by the end-user. These windows are often opened by components in a main window, and are meant to be on top of the screen. To this end, they may have their Layer property set to a number higher than zero so they don't get lost behind the main window. To get a window to pop-up at a specific position, edit the Window's Starting Location property.

Popup windows are often parameterized so they can be re-used.

Docked Windows

A **Docked Window** is one whose Dock Position is set to anything but Floating. This will make the window stick to one side of the screen, and nothing can overlap it. It will also typically have its border and titlebar display policies set to Never. This makes the "docked" window appear to be joined seamlessly with the current "screen" window.

These screens are usually tall and skinny or short and wide, depending on the side they're docked to. The purpose of a docked window is to make some information always available; typically navigation controls and overall status information. Using docked windows can help eliminate repetitive design elements from being copied to each screen, making maintenance easier.



Next ...

- Navigation Strategies
- Parameterized Pop-up Windows
- Window Properties

Window Properties

There are a number of window properties that change the way the windows look and behave. These properties are: Closable, Maximizable, Resizable, Start Maximizable, Title, Border Display Policy, Titlebar Display Policy, Dock Position, Dock Index, and Layer.

You can modify these window properties, just like a component's property. Simply select the window, either by clicking on its title bar, or clicking on the window's node in the Project Browser, and while it is open, set the property in the **Property Editor**.

Window Properties

Watch the Video

Behavior		
Dock Position	Determines the position this window is docked to, or if it is floating.	
	Scripting name	**`dockPosition`**
	Data type	int
	Values	0 Floating 3 West 4 South 2 East 1 North
Closable	Closable Determines whether or not to draw the close (X) button in the upper right corner.	
	Scripting name	**`closable`**
	Data type	boolean
Maximizable	Determines whether or not to draw the maximize button in the upper right corner.	
	Scripting name	**`maximizable`**
	Data type	boolean
Resizeable	Determines whether or not to let the user resize the window.	
	Scripting name	**`resizable`**
	Data type	boolean
Start Maximized	When set to true, the window will become maximized when it is opened.	
	Scripting name	**`startMaximized`**
	Date type	boolean
Cache Policy	By default this property is set to **`Auto`**, which keeps a window in a memory cache for a while after it is closed, so that if it is opened again it will be quick. The window isn't "active" while it is closed: all of its bindings and scripts are shut down. Setting this property to **`Never`** causes a fresh copy of the window to be deserialized every time it is opened. This is a performance hit, but it also is a convenient way to "clear out" the values of the window from the last time it was opened, which can be helpful in data-entry screens. Setting the property to **`Always`** will trade memory for higher performance, causing the window to always remain cached after the first time it is opened. This means the window will open very fast, but your Client will need lots of memory if you do this to a large amount of windows.	
	Scripting name	**`cachePolicy`**
	Data type	int

	Flags	expert
	Values	0 Auto 1 Never 2 Always
Appearance		
Title	The title to be displayed in this window's titlebar.	
	Scripting name	**`title`**
	Data type	String
Border Display Policy	Determines if window's border is shown in various window states.	
	Scripting name	**`borderDisplayPolicy`**
	Data type	int
	Values	0 Always 1 Never 2 When Not Maximized
Titlebar Display Policy	Determines if window's titlebar is shown in various window states	
	Scripting name	**`titlebarDisplayPolicy`**
	Data type	int
		0 Always 1 Never 2 When Not Maximized
Titlebar Height	The height of the window's titlebar.	
	Scripting name	**`titlebarHeight`**
	Data type	int
Titlebar Font	The font of the window title in the titlebar.	
	Scripting name	**`titlebarFont`**
	Data type	Font
Layout		
Location	The location that this window will open up at. Only applicable to floating windows that are not set to start maximized. Also, you must un-check the **Center Window** checkbox on the open-window navigation action in order for this location to take effect	
	Scripting name	**`startingLocation`**
	Data type	Point
Size	The dimensions of the window. This can be manipulated by selecting the window and dragging the resize handles along the windows right and bottom edges.	
	Scripting name	**`size`**
	Data type	Dimension
Minimum Size	The minimum size that this window will allow itself to be resized to.	
	Scripting name	**`minimumSize`**

	Data type	Dimension
	Flags	expert
Maximum Size	The maximum size that this window will allow itself to be resized to.	
	Scripting name	**`maximumSize`**
	Data type	Dimension
	Flags	expert
Layer	Sets the layer that this window is in. Default layer is 0, which is the bottom layer. Windows in higher layers will always be shown on top of windows in layers beneath them.	
	Scripting name	**`layer`**
	Data type	int
	Flags	expert
Dock Index	Determines the order of windows docked to the same side.	
	Data type	int
	Scripting name	**`dockIndex`**

Next ...

- Open Static Window(s) on Startup
- Open Dynamic Window(s) on Startup

Open Static Window(s) on Startup

You can have a static set of windows automatically open when a client is opened.

To open a static window on startup

Lets say you want the Main Window and the Navigation Window to open in the Client on Startup.

1. In the Designer, right-click on a window and choose **Open on Startup**.
   An icon is displayed next to the window so you can see visually which ones will be opened on start up.

2. Go to your Client, logout and log back in.
   Now you can see the Main Window and the Navigation Window open automatically.

Open Static Window (s) on Startup

Watch the Video

Next ...

- Creating Components
- Tag Binding - Drag and Drop
- Open Dynamic Window(s) on Startup

Open Dynamic Window(s) on Startup

Sometimes a project needs to change its startup windows depending on who logged in, what security roles they have, or what computer the Client is launched on. In these cases, rather than setting a static startup window, you can write a Client Startup Script that uses the `system.nav` library to open a dynamic set of windows based on hostname, IP address, user who logged in, and more.

INDUCTIVE UNIVERSITY

Open Dynamic Window(s) on Startup

Watch the Video

Next ...

- Docked Windows - Axis Precedence and Dock Index

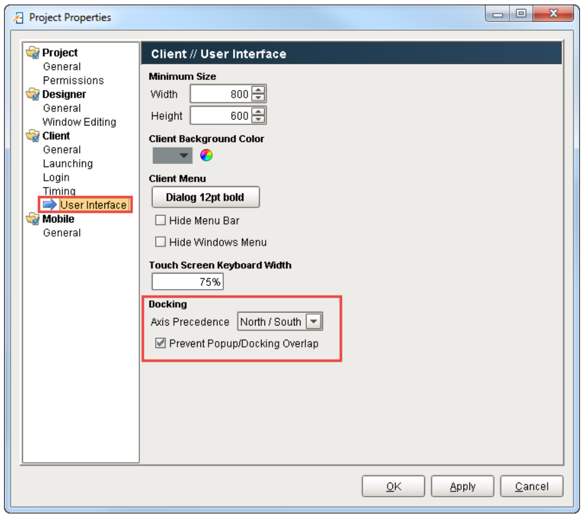
Docked Windows - Axis Precedence and Dock Index

Docked windows fill up the edges of the Client. When you have more than one docked window that share a corner, you can specify which window will take control over that corner.

To set the docked window order precedence

1. Go to **Project > Properties**.
2. The Project Properties window is displayed.

3. Go to **Client > User Interface** page.
4. Select a value for the **Docking > Axis Precedence** drop-down. For example, if you choose **Nor th/South** then North and South docked windows will take precedence or over the East and West windows.


Project Properties

Project
General
Permissions
Designer
General
Window Editing
Client
General
Launching
Login
Timing
User Interface
Mobile
General

Client // User Interface

Minimum Size
Width 800
Height 600

Client Background Color

Client Menu
Dialog 12pt bold
☐ Hide Menu Bar
☐ Hide Windows Menu

Touch Screen Keyboard Width
75%

Docking
Axis Precedence North / South
☑ Prevent Popup/Docking Overlap

OK Apply Cancel

5. **Save and Publish** your project.

6. Open your project in the **Client**.

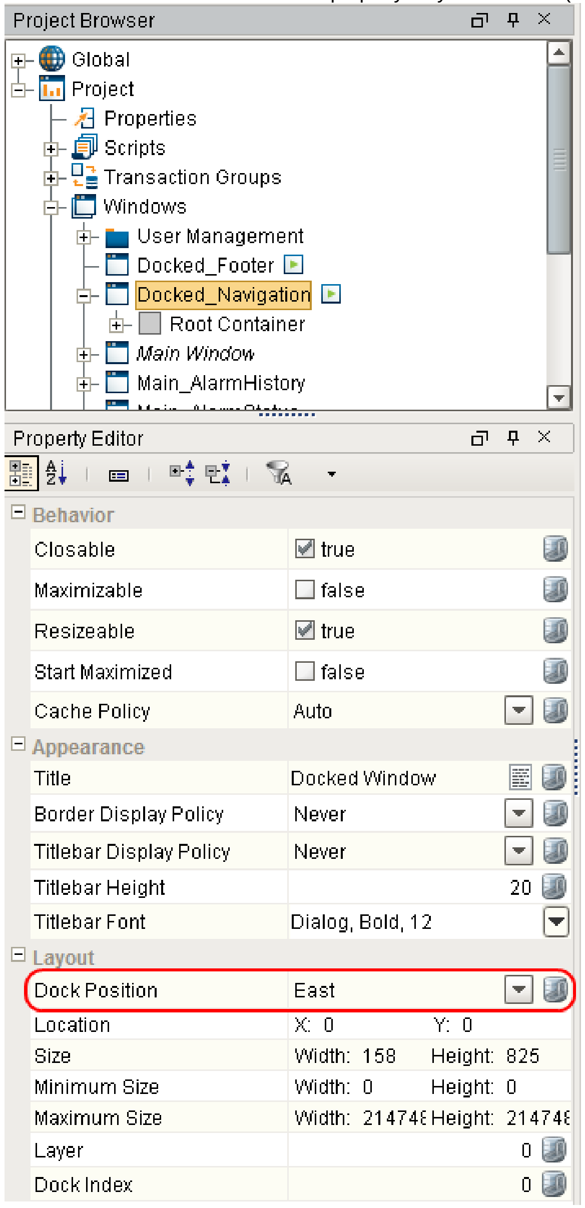
For Testing

INDUCTIVE UNIVERSITY

Docked Windows - Order Precedence

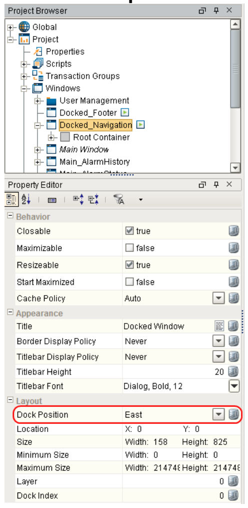
Watch the Video

- Make sure to set the **Dock Position** property on your windows (North/South/East/West)



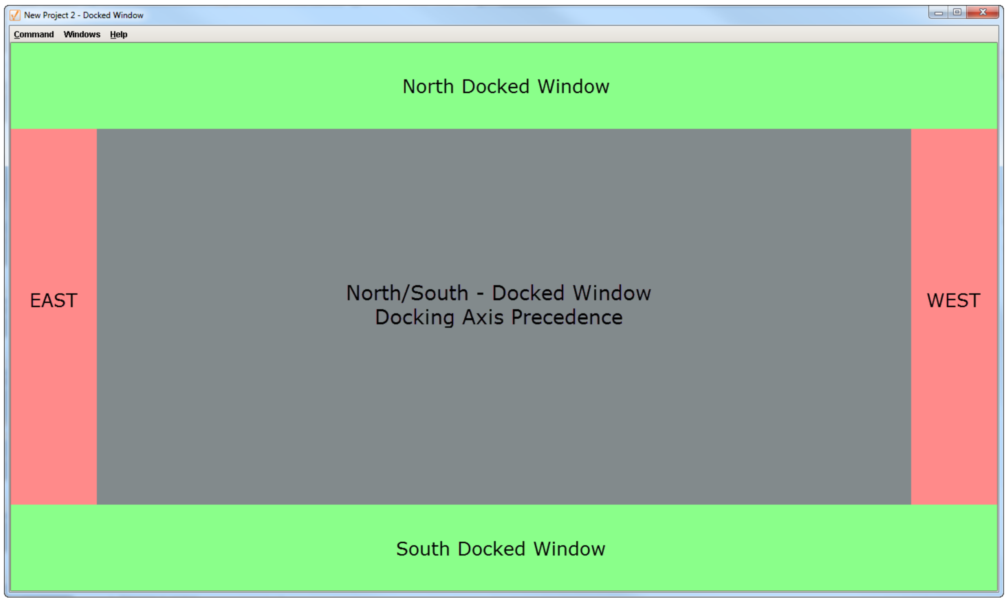
Behavior	
Closable	✓ true
Maximizable	☐ false
Resizeable	✓ true
Start Maximized	☐ false
Cache Policy	Auto
Appearance	
Title	Docked Window
Border Display Policy	Never
Titlebar Display Policy	Never
Titlebar Height	20
Titlebar Font	Dialog, Bold, 12
Layout	
Dock Position	East
Location	X: 0 Y: 0
Size	Width: 158 Height: 825
Minimum Size	Width: 0 Height: 0
Maximum Size	Width: 21474 Height: 21474
Layer	0
Dock Index	0

- Set the **Open on Startup** option for each docked window. **Note:** logging in is the only way to activate the **Open on Startup** settings.

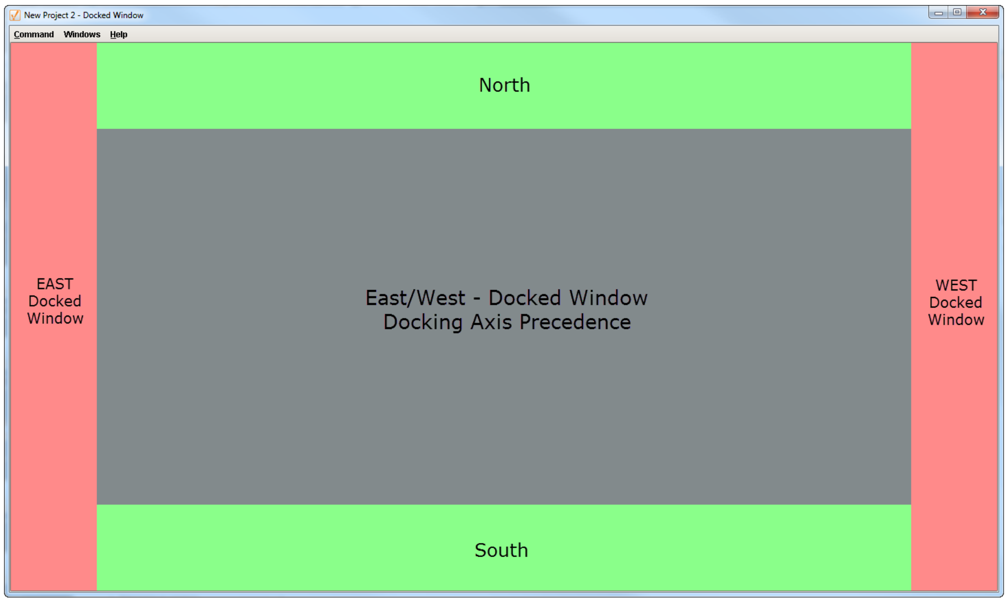

Behavior	
Closable	✓ true
Maximizable	☐ false
Resizeable	✓ true
Start Maximized	☐ false
Cache Policy	Auto
Appearance	
Title	Docked Window
Border Display Policy	Never
Titlebar Display Policy	Never
Titlebar Height	20
Titlebar Font	Dialog, Bold, 12
Layout	
Dock Position	East
Location	X: 0 Y: 0
Size	Width: 158 Height: 825
Minimum Size	Width: 0 Height: 0
Maximum Size	Width: 21474 Height: 21474
Layer	0
Dock Index	0

Examples of Docked Windows

North/South Docking Axis Precedence



East/West Docking Axis Precedence



Next ...

- Locate All Opened Windows In Client

Locate All Opened Windows In Client

You can locate all of the currently opened windows in a Client from the **Windows** menu command.

Open Windows and Performance

While a window is open, its query bindings are running, its tag bindings are keeping tags subscribed, and its event scripts are being executed. This means that an open window is actively using system resources, both on the Client's host machine, and on the Gateway's server machine as its queries and tag subscriptions must be handled. For these reasons, it is important that you properly implement a navigation strategy that prevents windows that are no longer being used from being held open.

The most common mistake that will cause windows to stay open unintentionally is to implement a swapping navigation system using the **swapTo** function on windows that are not maximized. When you do this, the **swapTo** function cannot calculate the window to swap from, thereby simply opening the window, and not closing any windows. It is easy to check the Windows menu to see what windows are currently open. If there are more windows listed there than you can currently see, there is a problem in your navigation logic that is failing to close windows properly.

Locate All Opened Windows In Client

Watch the Video

Next ...

- add link here

Navigation Windows

Setting up a navigation strategy allows you to navigate between different windows in the runtime Client. There are several different navigation strategies you can use and choose from.

In this section ...

Navigation Strategies

There are several different types of runtime navigational strategies you can use when designing your Ignition project. A typical navigation strategy for a Vision project is as follows:

1. Have a **Docked** window or two, usually docked North and/or West.
2. Have a single **Main** window visible at a time.
3. Use swap navigation to swap between the **Main** windows. This ensures that only one main window is open at a time.
4. Use standard open navigation to open various **Popup** windows as necessary.

This style of project is so common, that the default operation of the Tab Strip component expects it. When it is in its default automatic operation, it expects that each tab represents a "screen" window, and will automatically swap from the current screen to the desired screen.

Furthermore, the `[System]/Client/User/CurrentWindow` tag is calculated based on this strategy: its value is the name of the current maximized window. This navigation strategy is used in the Ignition Online Demonstration that you can download from our website.

Navigation Strategies

Watch the Video

Next ...

- Swapping vs. Opening

Swapping vs. Opening

The two primary window navigation operations are: swapping and opening.

To set the Open or Swap option

1. In the Designer, right-click on a component and choose **Scripting**.
   The Component Scripting window is displayed.

2. Go the to **Navigation** tab and click the **Open/Swap** button.

3. Select **Open** or **Swap** as your Navigation strategy.
   Typically you want to use the Open option for Popup and Docked windows and the Swap for the Main windows that will be in the background as you can only view one at a time.

Swapping vs. Opening

Watch the Video

Swapping

In general, swapping involves closing one window, and then opening another window in its place. This operation can be performed on windows in any state: docked or floating, maximized or not. The **Start Maximized** and **Dock Position** properties of the window that is being swapped in will be ignored - it will take the dock and maximized state of the window that it is replacing.

This operation is so common in the typical navigation strategy that there is even a version of the swapping function dedicated to it, the **swapTo** function. This function eliminates the need to specify the window to swap from - you only need to specify the window to swap to. It will take the current window - that is, the current maximized window - as the window to swap from.

See the following functions for more information:

- system.nav.swapWindow
- system.nav.swapTo

Opening

Opening and closing are the basic window navigation options. Opening a window opens the window at the same size it was in the Designer, unless the **Start Maximized** property is **true** or the **Dock Position** is **not Floating**.

To make a floating popup window open at a specific location, make sure to set the **Location** property of the window in the Designer. If the window was recently open, it will open in its last state due to window caching. See the Window's Cache Policy property for more information.

See the following functions for more information:

- system.nav.openWindow

Next ...

- Navigation - Tab Strip

Navigation - Tab Strip

It is very common to use tabs for navigation in Ignition.

To configure the Tab Strip in the Designer

1. Go to Project Browser, expand the **Navigation** window, to see the **Tabs** component inside of it. The **Tabs** component, has a **Navigation Mode** property which is set to **Swap to window**.
2. Right-click on the tab component, choose **Customizers > Tab Strip Customizer**. Here you can specify which window to open with each tab.
3. Save you project and open it in the runtime to see that each tab now goes to a different window.

Navigation - Tab Strip

Watch the Video

Next ...

- Navigation - 2 Tier

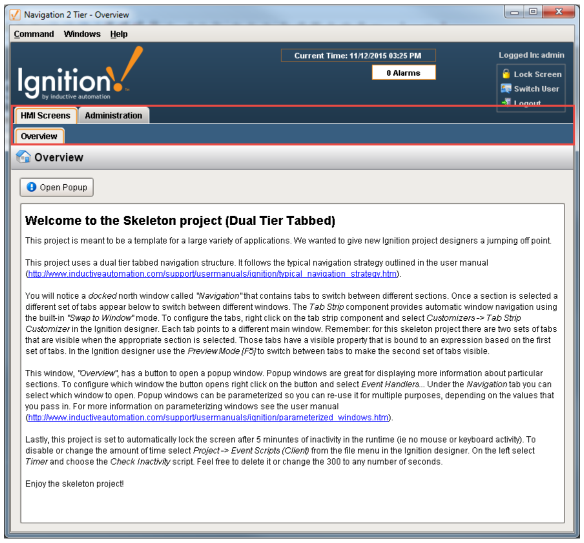
Navigation - 2 Tier

2 Tier Navigation involves using 2 sets of tabs to navigate around various areas of your project. Ignition has a built-in skeleton project that you can start from to build your new project in the Designer and customize it to fit your needs. It uses a typical navigation strategy. There are 2 windows that open up on startup: a docked north window that contains tabs that are always open to do navigation, and the main window at the bottom which fills in the rest of the space.

INDUCTIVE UNIVERSITY

Navigation - 2 Tier

Watch the Video



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.

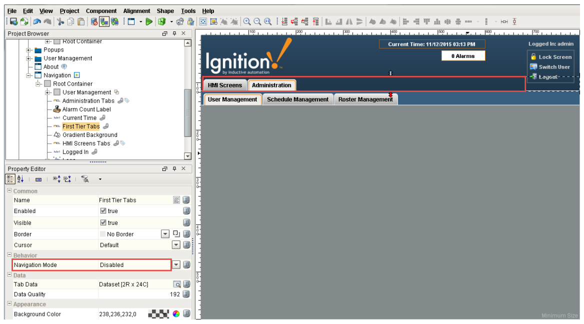
This window, *"Overview"*, has a button to open a popup window. Popup windows are great for displaying more information about particular sections. To configure which window the button opens right click on the button and select *Event Handlers*... Under the *Navigation* tab you can select which window 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!

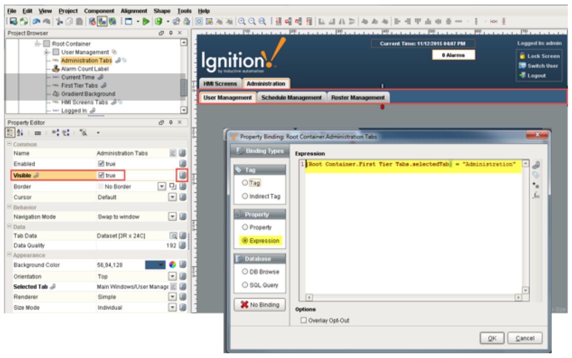
In the docked north window there are 2 sets of tabs. The first tier component has the '**HMI Screens**' and '**Administration**' tabs which contain multiple main windows. The HMI Screens tab has one main window on the second tier called Overview, and the Administration tab contains 3 main windows. Each second tier tab represents a main window and when clicked will swap out the bottom window ensuring only one main window is visible at a time. This 2 Tier Navigation strategy lets you organize all your windows into different groupings by having the first layer of tabs.

In the **Project Browser**, you will see multiple Tab Strip components. In this example, there are two tabs named '**HMI Screens**' and '**Administration**'. Click on the **first tier tab**. You will notice that the **Navigation Mode** property is set to **Disabled** and does not have any navigation functionality.

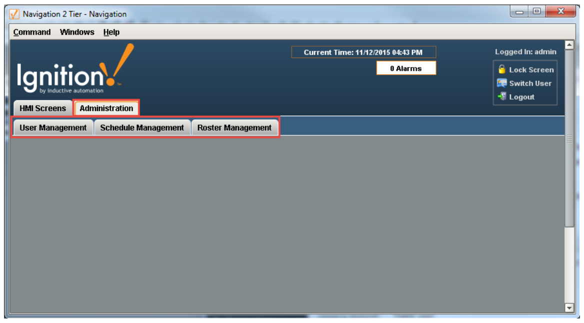


Common	
Name	First Tier Tabs
Enabled	✓ true
Visible	✓ true
Border	No Border
Cursor	Default
Behavior	
Navigation Mode	Disabled
Data	
Tab Data	Dataset [2R x 24C]
Data Quality	192
Appearance	
Background Color	238,236,232,0

In the same example, the **Adminisration Tab** component has 3 tabs named '**User Management**', '**Schedule Management**' and '**Roster Management**'. Click one of these **second tier tab strip components**. The **Visible** property for the second tier tabs is bound to a simple expression looking at the first tier tab. You can see the expression by clicking on the binding icon for the **Visible** property.

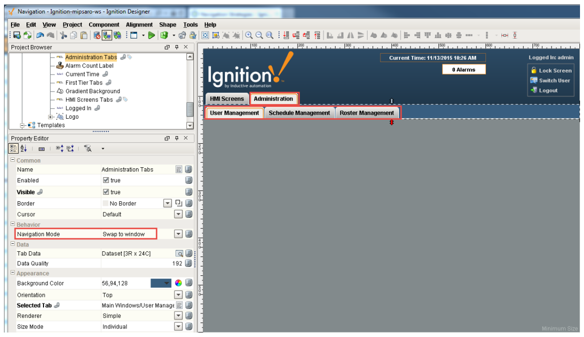

Property Binding: Root Container.Administration Tabs
Expression
Root Container.First Tier Tabs.selectedTab = "Administration"
OK
Cancel

The second tier tabs will only be visible based on which first tier tab is selected as shown in the example below. When the **Administration tab** is selected, Ignition displays the **Administration Tab** components: User Management, Schedule Management and Roster Management.



In the **Project Browser**, click on the **Administration Tab** component. You will notice that the **Navigation Mode** property is set to **Swap to window**. Each tab on a second tier tab strip represents a main window, and when clicked it, will swap out the main window ensuring only one main window is visible at a time.

You can also go into the Navigation Tab Strip Customizer to configure tabs on both tiers that best fit your requirements.



The 2 Tier Navigation strategy lets you organize your main windows into different sections making navigation easy for users.

Next ...

- Navigation - Tree View

Navigation - Tree View

You can navigate through your project using the multi-tier **Tree View** component. In fact there is a built-in project template that you can use to get started with this navigation strategy. The key is the Tree View component that provides the navigation for the project.

Navigation - Tree View

Watch the Video

To use a Tree View Component for Navigation

1. Select the **Tree View** component from the Navigation window or drag a **Tree View** component onto a docked window.

2. From Property Editor, find the **Items** property and click on the **Dataset Viewer** icon.
   The **Dataset Viewer** is displayed which contains information about the tree nodes and folders.

   You can also manage the Dataset by right-clicking on the Tree View component and choosing **C ustomizers>Tree View Customizer**, however in this example we use the Dataset icon which is located next to the Items property's binding icon.

   If you are using the existing Navigation window from the Template Project, there will be a **windo wPath** column on the left that will determine the folder organization of the **Tree View** component, while the **path** column will determine the name of the node on the Tree View component. If you are using a new **Tree View** component, add a column names windowPath that is a string type.

3. The **Tree View** component needs to have a script that executes when the user double-clicks on one of the nodes in the **Tree View** component. Here is the script that is executed on the **Tree View**'s `mouseClicked` event handler, copy it in if needed.

```
#If the user performs a double click, open the appropriate window.
if event.clickCount == 2:
        row = event.source.selectedItem
        data = event.source.data
        if row != -1:
                #Grab the window path value out of the trees view's
items dataset
                windowPath = data.getValueAt(row, "windowPath")
                system.nav.swapTo(windowPath)
```

Next ...

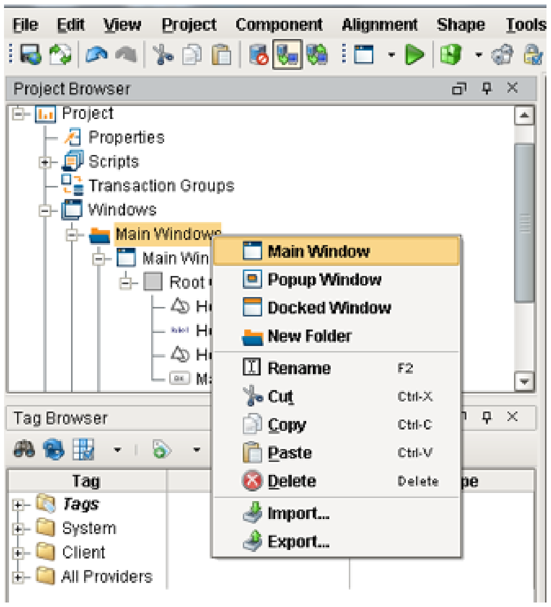
- Navigation - Back and Forward Buttons

Navigation - Back and Forward Buttons

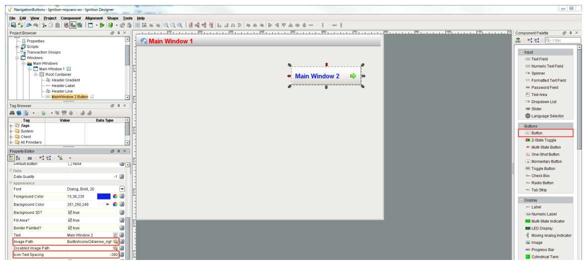
Another navigation strategy in Ignition is to have Forward and Back Buttons on your main screen to navigate between different windows. This navigation strategy does not have a docked window, tree view or tabs to navigate around. It is one big main window and has buttons to navigate forward and back from one Main Window to another.

This navigation strategy is very simple to setup in Designer. You can start by creating your project using a blank template.

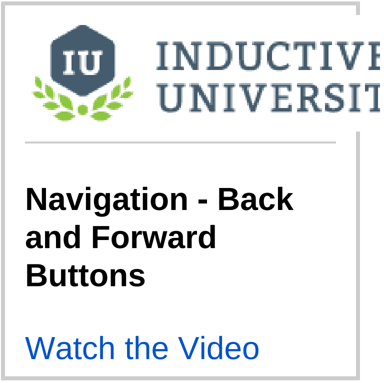
1. In the **Project Browser**, right click on the Main Windows folder and select **Main Window**. You have the option of renaming your Main Window to whatever best describes your window. In this example, it's called Main Window 1. You don't need to have a header on each window, but for this example, a label was added for clarity and called **Main Window 1**. Repeat this step to create Main Window 2 and Main Window 3.



2. From the **Component Palette** in Designer, select and drag a **Button** component to your Main Window 1. Label the button, **Main Window 2**. In the **Property Editor**, add a right green arrow to your button component from the **Image Path folder**. To get the arrow to move to the right side of the button, click the **Icon-Text Spacing property** and enter **-200**. The Icon-Text Spacing property is used to move the location of the green arrow within the button. Select a desired value and the green arrow will move within the button component.

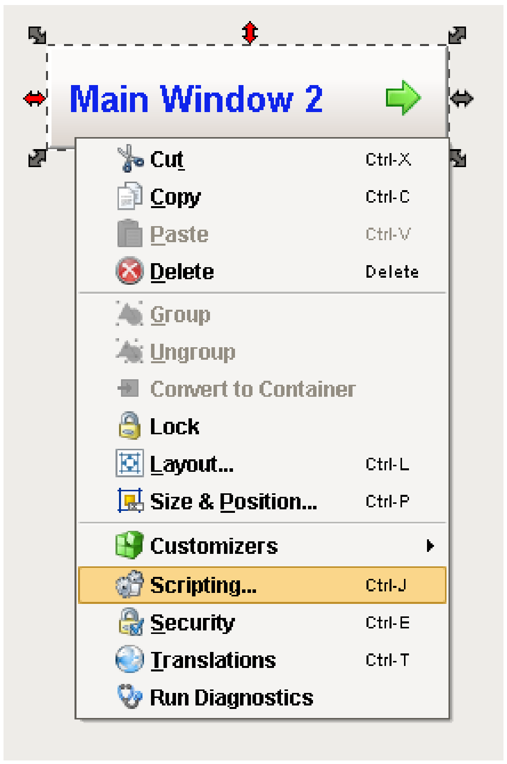


3. Now you need to tell the Main Window 2 button what to do when someone clicks on it. Right click on the Main Window 2 button and select **Scripting**.



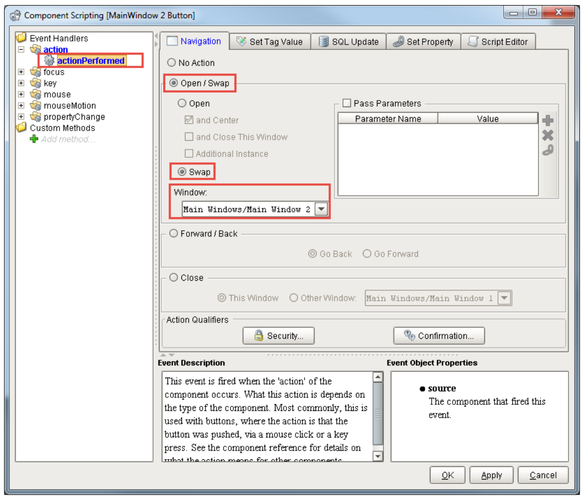
Navigation - Back and Forward Buttons

Watch the Video



The component scripting dialogue box will open. Under the Event Handlers folder, open the action folder and select **actionPerformed**. Click the **Open / Swap** and **Swap** radio buttons. The Swapping function is used to put buttons on windows for easily building simple scripts to go back and forth between different main windows. The Swap function ensures only one main window is open at a time.

From the **Window dropdown box**, select **Main Windows/Main Window 2**. Click **OK**. **Save and Publish** your project.



Component Scripting [MainWindow 2 Button]

- Event Handlers
  - action
    - actionPerformed
  - focus
  - key
  - mouse
  - mouseMotion
  - propertyChange
- Custom Methods
  - Add method...

Navigation | Set Tag Value | SQL Update | Set Property | Script Editor

No Action
[✓ Open / Swap]
Open
and Center
and Close This Window
Additional Instance
[✓ Swap]
Window: Main Windows/Main Window 2

Pass Parameters

Parameter Name	Value

Forward / Back
Go Back Go Forward

Close
This Window Other Window: Main Windows/Main Window 1

Action Qualifiers
Security... Confirmation...

Event Description

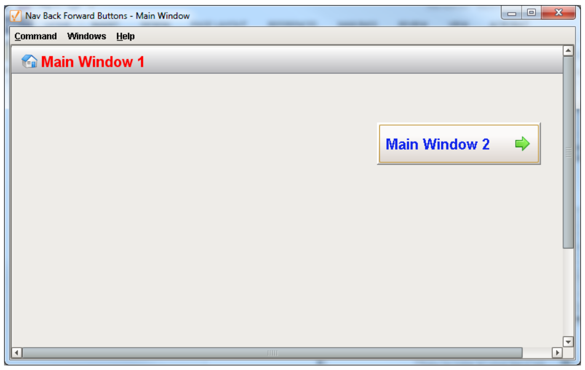
This event is fired when the 'action' of the component occurs. What this action is depends on the type of the component. Most commonly, this is used with buttons, where the action is that the button was pushed, via a mouse click or a key press. See the component reference for details on what the action means for other components.

Event Object Properties

- **source**
  The component that fired this event.

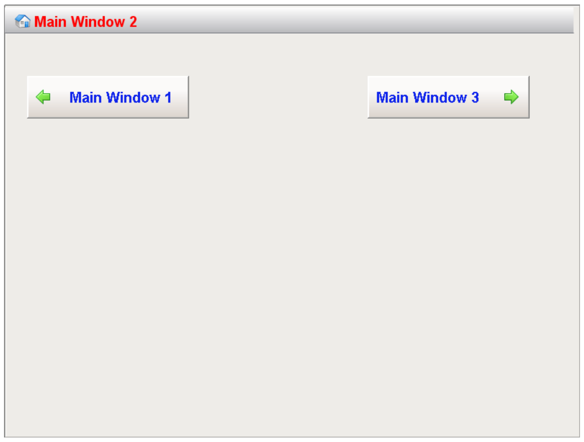
OK Apply Cancel

Open a Client, click on the **Main Window 2** button and Ignition swaps the Main Window 1 with Main Window 2.



4. Repeat Steps 2 and 3 to create Main Windows 2 and 3 respectively. Make sure for Main Window 2, you have a button for both Main Windows 1 and 3. Your window path for the Main Window 1 button is set to Main Windows/Main Window 1. For the Main Window 3 button, set the window path to Main Windows/Main Window 3.

5. In the following example, each button navigates you to the next main window. Click on the **Main Window 1** button and Ignition navigates you to Main Window 1. It swaps out Main Window 2 with Main Window 1 so you always have 1 main window open. Click on the Main Window 3 button and Ignition swaps out Main Window 2 with Main Window 3.



Navigating between all 3 windows is easy using the Back and Forward Buttons navigation strategy in Ignition.

Next ...

- Navigation - Drill Down

Navigation - Drill Down

You can navigate through your project using a map to drill into different areas of your project. A popular option is to have a main window that has an image representing the facility. The image has various boxes that overlay the image. When the user selects one of the boxes, the Client swaps windows with a window that displays the information that pertains to the selected area.

To create a navigation drill down

1. Create a box on the Main window.
2. Right-click the box and select **Scripting**.
3. Select the `mousePressed` event handler and with the **Navigation** tab selected.
   a. Select the **Open/Swap** radio button.
   b. From the **Window** drop-down box, select the window that relates to the selected area.
4. Click **OK**.

A back button on the area window would navigate you back to the main window.

INDUCTIVE UNIVERSIT

Navigation - Drill Down

Watch the Video

Next ...

- add link here

Popup Windows

Popup windows are often opened by components in a main window, and are meant to be on top of the screen. They are used to view setpoints or zoom into a specific area and more.

In this section ...

Open Popup Window

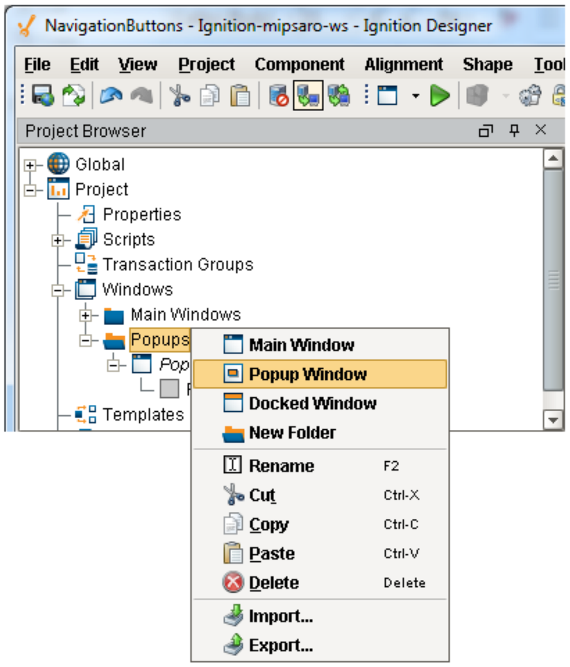
INDUCTIVE UNIVERSITY

Open Popup Window

Watch the Video

In order to open a Popup window, you first have to create a **Popup Window**. You can add a Popup window to any folder in the Designer.

1. In this example, a Popup folder already exists so we will right click on the Popup folder to create a new '**Popup**' window. Right click on the **Popup window** then rename it to '**Popup'** or give it an appropriate name.

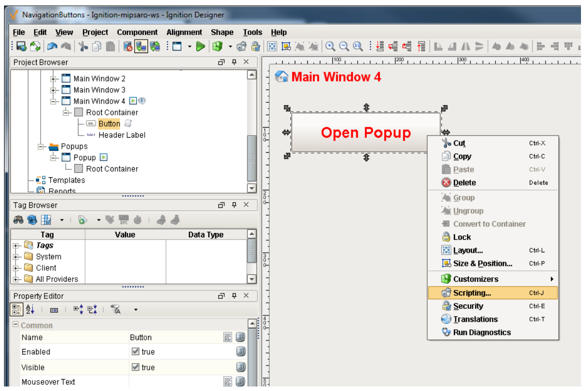


You can make the Popup window as big or small as you want. You can also set properties in the Property Editor by making the window Closable, Resizable, and giving it a Title. You can even setup the Border or Title Bar to show-up on the window.

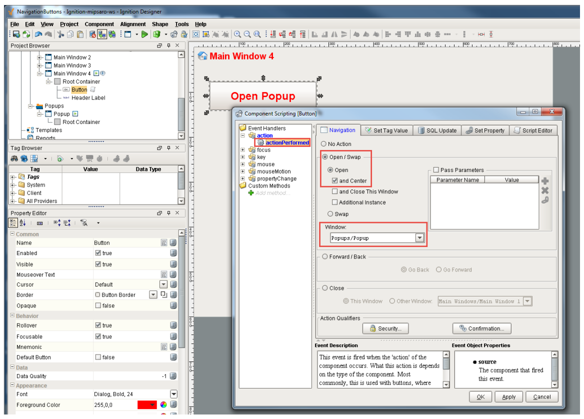
Once the window is created, you can open up the Popup window from any other window. For example, you can open it up from any Main Window. You can open a Popup window from a button, a label, an image or any component you like.

2. In any Main Window, drag a **Button** component from the **Component Palette** to your window. Do a slow double click on the word '**Button**' and change the text to '**Open Popup**.'

   Now you need to tell the Open Popup button what you want it do when you click it. Select the **O pen Popup button**, right click and select **Scripting** from the dropdown.



The Component Scripting dialogue box will open. Select **actionPerformed**, and click the **Open / Swap** then **Open and Center** options on the Navigation tab. From the **Window** dropdown box, set the window path to **Popups/Popup.** Click **OK**. **Save and Publish** your project.


Component Scripting [Button]

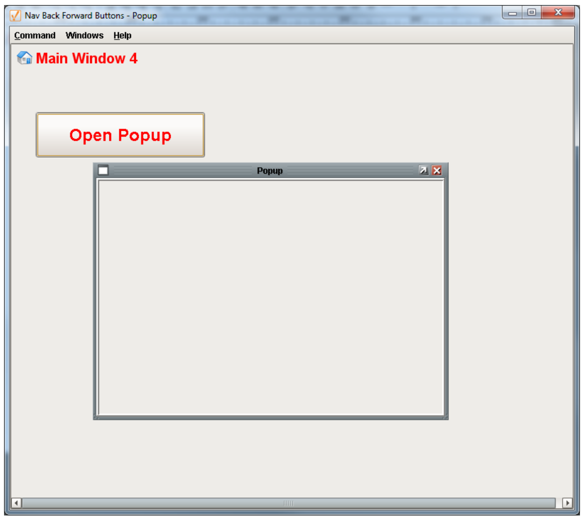
Event Handlers > action > actionPerformed

Navigation tab: Open / Swap [✓ Open], [✓ and Center]; Window: Popups/Popup

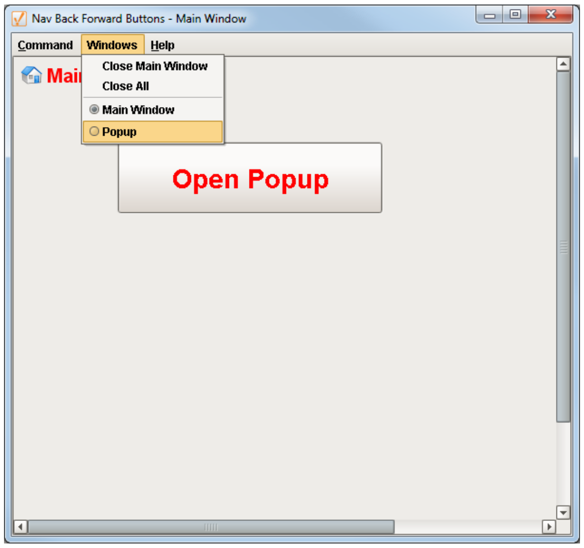
Event Description: This event is fired when the 'action' of the component occurs. What this action is depends on the type of the component. Most commonly, this is used with buttons, where

Event Object Properties: source — The component that fired this event.

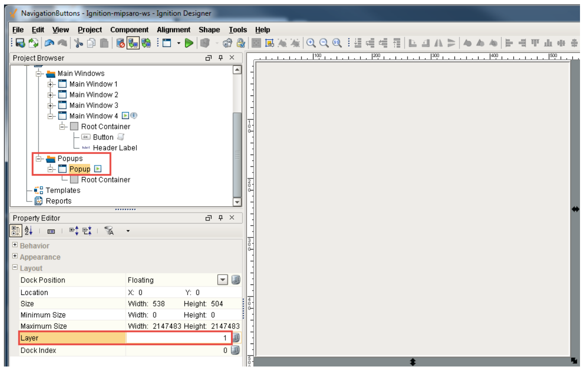
Open a **Client** and click the **Open Popup** button on your Main Window. It opens the Popup window and you can move it around, make it bigger and close it.



Another behavior of the Popup window occurs when you open up the Popup window and click on Main Window 4. It looks like the Popup window disappears, but it's actually behind Main Window 4. Go to **Windows** on the menu bar and click **Popup** to bring the Popup window back to focus.

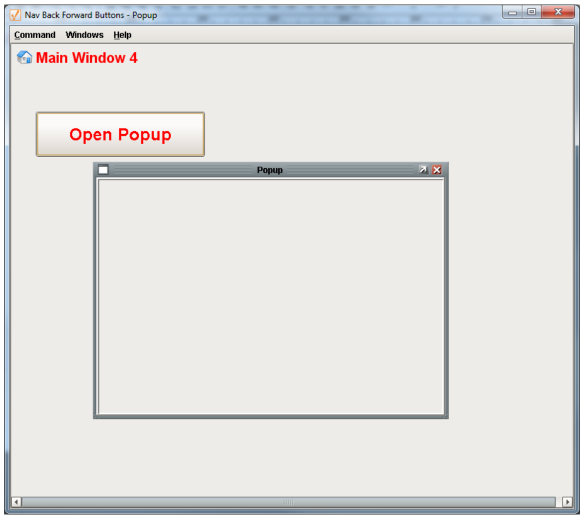


3. There is a property on the Popup window that will force the popup to stay on top of the Main Window. In the **Designer**, click on your Popup window in the Project Browser (not the Root Container), go to the **Property Editor** and click the **Layer** property. The higher the Layer number, the more on top it will be. If your 'windows' Layer properties are set to 0, they are all going to share the same layer. They can go back and forth over each other. If you set the Layer number to **1** for this example, the Popup window will always be on top. **Save and Publish** your project.

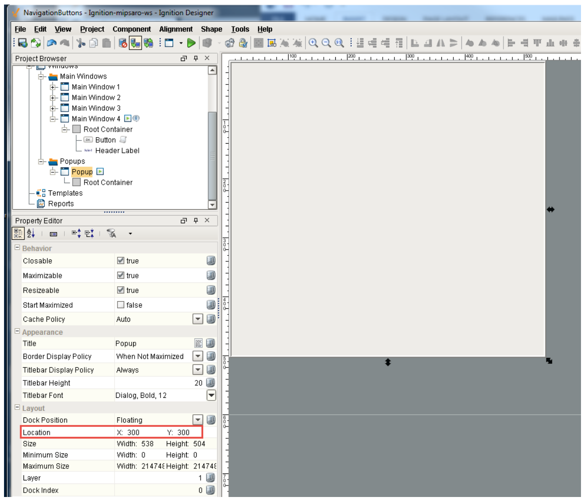


Dock Position: Floating
Location: X: 0 Y: 0
Size: Width: 538 Height: 504
Minimum Size: Width: 0 Height: 0
Maximum Size: Width: 2147483 Height: 2147483
Layer: 1
Dock Index: 0

After you update the **Client**, you will see your Popup window. Click your **Main Window** and you'll see that your Popup window will stay on top.



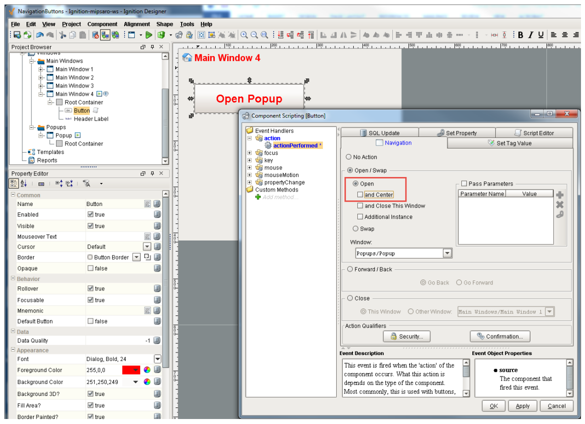
4. It is possible for you to open and set a specific location where you want the Popup window to be located. In **Designer**, click on your **Popup** window and go to the **Property Editor**. Click on the **Location** property and set **X to 300** and **Y to 300**.


NavigationButtons - Ignition-mipsaro-ws - Ignition Designer

Property Editor

Behavior	
Closable	✓ true
Maximizable	✓ true
Resizeable	✓ true
Start Maximized	false
Cache Policy	Auto
Appearance	
Title	Popup
Border Display Policy	When Not Maximized
Titlebar Display Policy	Always
Titlebar Height	20
Titlebar Font	Dialog, Bold, 12
Layout	
Dock Position	Floating
Location	X: 300 Y: 300
Size	Width: 538 Height: 504
Minimum Size	Width: 0 Height: 0
Maximum Size	Width: 214748 Height: 214748
Layer	1
Dock Index	0

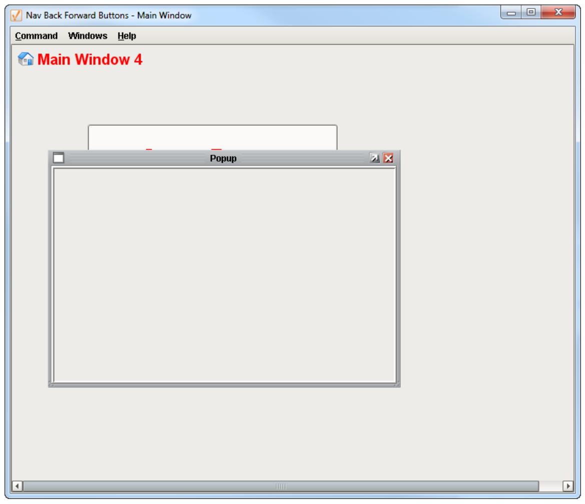
5. Go to your Main Window, double click **Open Popup** to open the Component Scripting dialog box and uncheck ‘**and Center’** so your Popup window will automatically open up in that specific X, Y location.


Component Scripting [Button]

Open / Swap: [✓ Open] and Center (unchecked)

Window: Popups/Popup

Save and Publish and update your **Client**, click on **Open Popup**.   The Popup window will open at the X 300 and Y 300 coordinates instead of opening it in the center of the screen.



In any Main Window, drag a **Button** component from the **Component Palette** to your window.   Do a slow double click on the word ‘**Button**’ and change the text to ‘**Open Popup**.‘

Next ...

- Parameterized Popup Window

Parameterized Popup Window

A parameterized popup window lets you pass information from one window to another window, where the receiving window would display the relevant information. Also, this allows you to maintain a single window that can be used to display similar information. For example, suppose you have 3 compressors organized in the following folder structure:

```
Comppressors

   C1
      HOA
      AMPS

    C2
      HOA
      AMPS

   C3
      HOA
      AMPS
```

The only difference is the compressor number. Imagine clicking on a compressor on your main window and a popup displays the diagnostic information about the compressor. What is passed is simply the number, therefore a single diagnostic window can represent many different compressors.

Parameterized Popup Window

Watch the Video

To pass parameters to the popup window

To pass parameters from one window to another, the receiving window must have Custom properties that receive the passed parameters. These parameters are passed to the receiving window's **Custom** properties on its Root Container. Once the event on the parent window is called, the parameters are passed to the receiving window's Custom properties on its Root Container. The component's properties on the receiving window can use the Root Container's Custom properties to address their bindings.

Let's suppose you have a window called `CompressorPopup` that you want to use to pass the compressor number to this window.

1. Right-click on the window and select **Customizers > Custom Properties**.
   The Custom Properties window is displayed.
2. Specify a **Name** for the **Custom** Property, for example name it `compNum`, and click **OK**.
   The **Custom** property is now created and displayed at the bottom of the **Property Editor**.

   Do not bind these **Custom** properties to anything, leave them unbound so you can pass values into them without any other values to override them.

3. Use `compNum` in your tag bindings for the controls on your screen using indirect tag bindings.
   For example, you might bind the control and indicator properties of a **Multi-State Button** to an indirect tag binding like:
   `Compressors/C{1}/HOA`

   where the {1} paremeter is bound to the property path: `Root Container.compNum`

   You could use a similar indirect binding to display the amperage in an analog Meter component.

4. Now, when opening the window, use a script like the following to open it to control compressor #6. Of course, you probably wouldn't write this script by hand, you'd use the navigation script builder. But it is useful to know what the script would look like.

```
system.nav.openWindow("CompressorPopup", {"compNum":6})
```

Next ...

- Popup Window - Multiple Instances

Popup Window - Multiple Instances

By default Ignition only opens a single copy of each window in the client. However, you can open multiple copies (or instances) of the window in the Client.

Opening multiple copies

By default, opening a window will only ever open one copy of any given window. If the window is already open, it simply brings it to the front. Normally this is the desired behavior. For example, if you opened the compressor popup window for compressor #6, and then opened it for compressor #4, the window that had been controlling #6 will switch to controlling #4.

Sometimes you may want to open a separate popup, one for #6, and one for #4, both at the same time. If this is the case, use the system.nav.openWindowInstance function call to open your window.

Popup Window - Multiple Instances

Watch the Video

Next ...

- Parameterized Popup Window and UDTs

Parameterized Popup Window and UDTs

Before we dive into Parameterized Popup Window and UDTs, a small Glossary of terms is provided to help you understand the terminology on this page.

Glossary

Name	Description
Custom Property	A property you add to a component or container. A custom property is a variable that you customize for your particular use.
User Defined Type (UDT)	Is a User-Defined data Type defined by its structure, tags, attributes and settings. UDTs are used to design re-usable, parameterized, and extendable data types.
Element / Parameter of a UDT	A dynamic value inside of a Custom Property, assuming the Custom Property is a UDT type.

INDUCTIVE UNIVERSITY

Parameterized Popup Window and UDTs

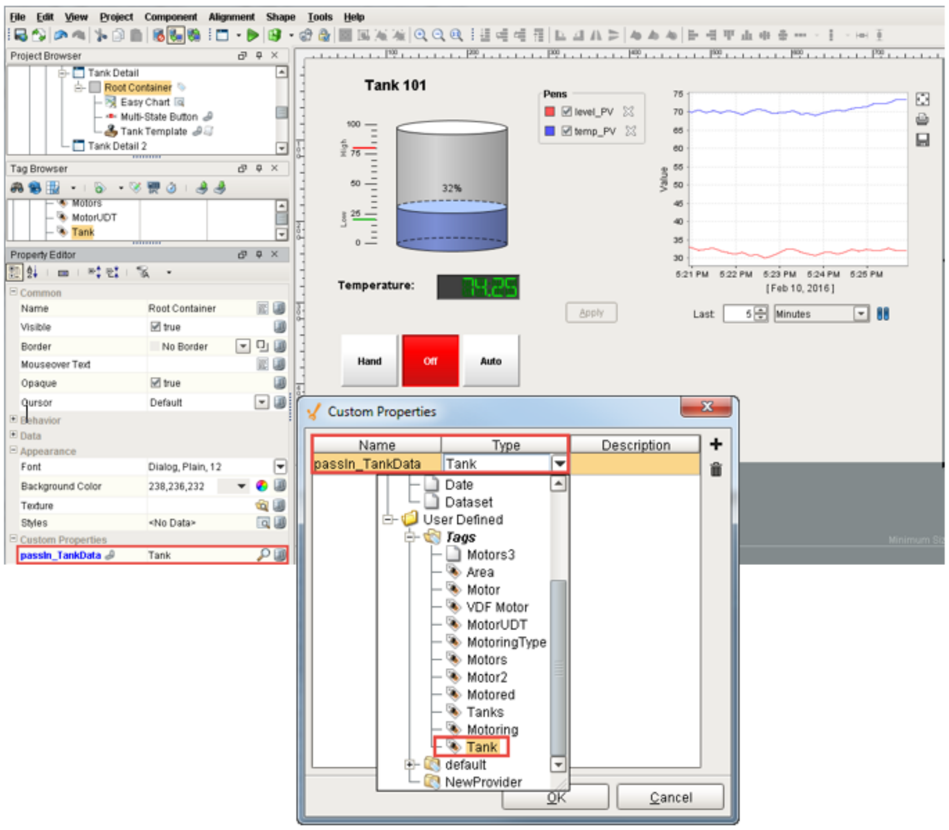
Watch the Video

UDT and matching Tank Template must be created to do this exercise

The examples in this section assume you have a UDT and matching Tank template (that uses the UDT) created. To learn more about creating UDTs and Templates go to UDT Parameter and Template Basics sections of this user manual.

When you pass an entire UDT into a popup window, you can take advantage of passing all the parameters inside of that UDT at once. By passing the entire UDT, you have every single value available to you that is inside the UDT.

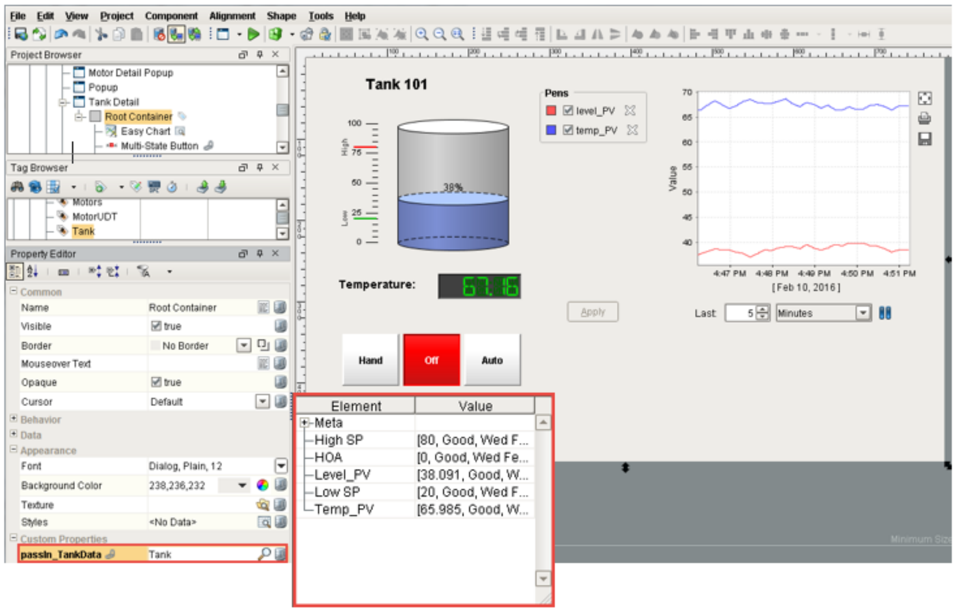
1. In this example, we have a **'Tank Detail'** popup window with a data type called '**Tank**.'
   On the root container, right click and select **Customizers > Custom Properties.**
   Click the **plus icon** and add a property called **'passIn_TankData'** and select the **'Tank'** data type from the dropdown list.
   Click **OK**. Note: parameters that are passed into a window can only be passed into properties that are on the **Root Container**.



Name	Value
Name	Root Container
Visible	true
Border	No Border
Mouseover Text	
Opaque	true
Cursor	Default
Font	Dialog, Plain, 12
Background Color	238,236,232
Texture	
Styles	<No Data>
passIn_TankData	Tank

Name	Type	Description
passIn_TankData	Tank	

2. In the Property Editor you will see the Custom Property called **'passIn_TankData.'** It was assigned a unique name so it is clear exactly what the Tank property is doing. It will hold the UDT that is passed in from another window.
   Click on the **magnifying glass** and see all the elements that are inside of the Tank UDT; (i.e., HOA, level_PV, temp_PV, etc.). All of the values will be NULL if you just added the property, they will get filled in later when tested in preview mode.



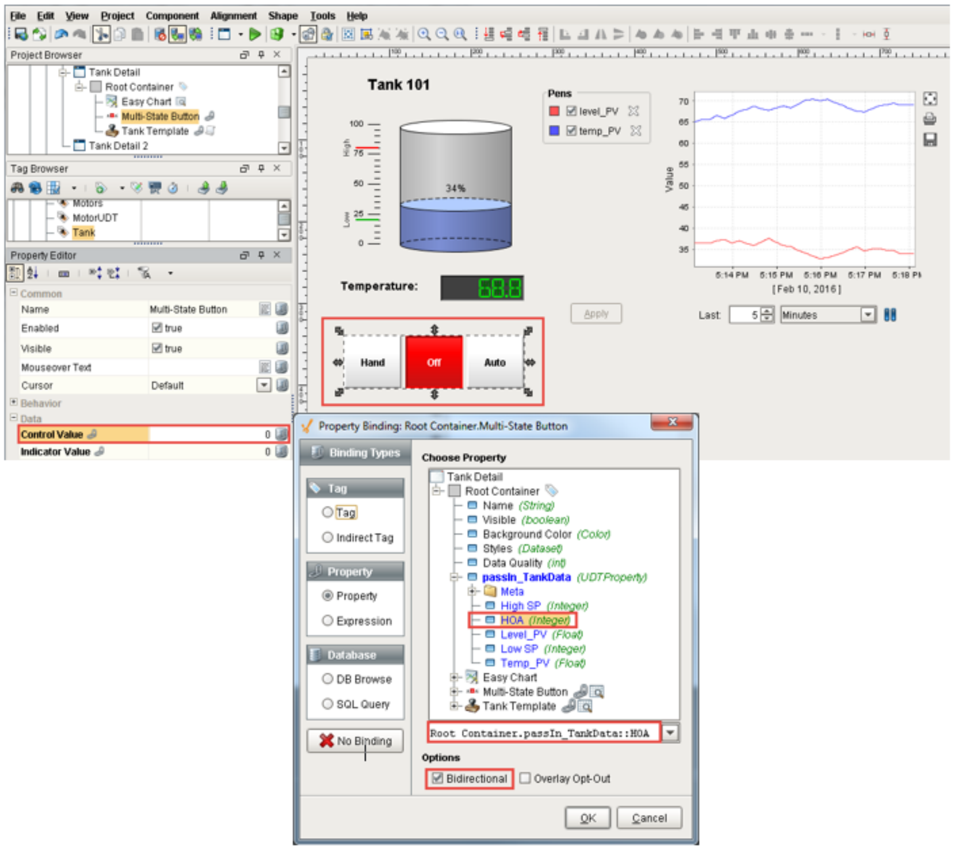
Common	
Name	Root Container
Visible	true
Border	No Border
Mouseover Text	
Opaque	true
Cursor	Default
Behavior	
Data	
Appearance	
Font	Dialog, Plain, 12
Background Color	238,236,232
Texture	
Styles	<No Data>
Custom Properties	
passIn_TankData	Tank

Element	Value
Meta	
High SP	[80, Good, Wed F...
HOA	[0, Good, Wed Fe...
Level_PV	[38.091, Good, W...
Low SP	[20, Good, Wed F...
Temp_PV	[65.985, Good, W...

3. Now, you can bind components inside the Tank Detail window to elements of the Tank UDT.

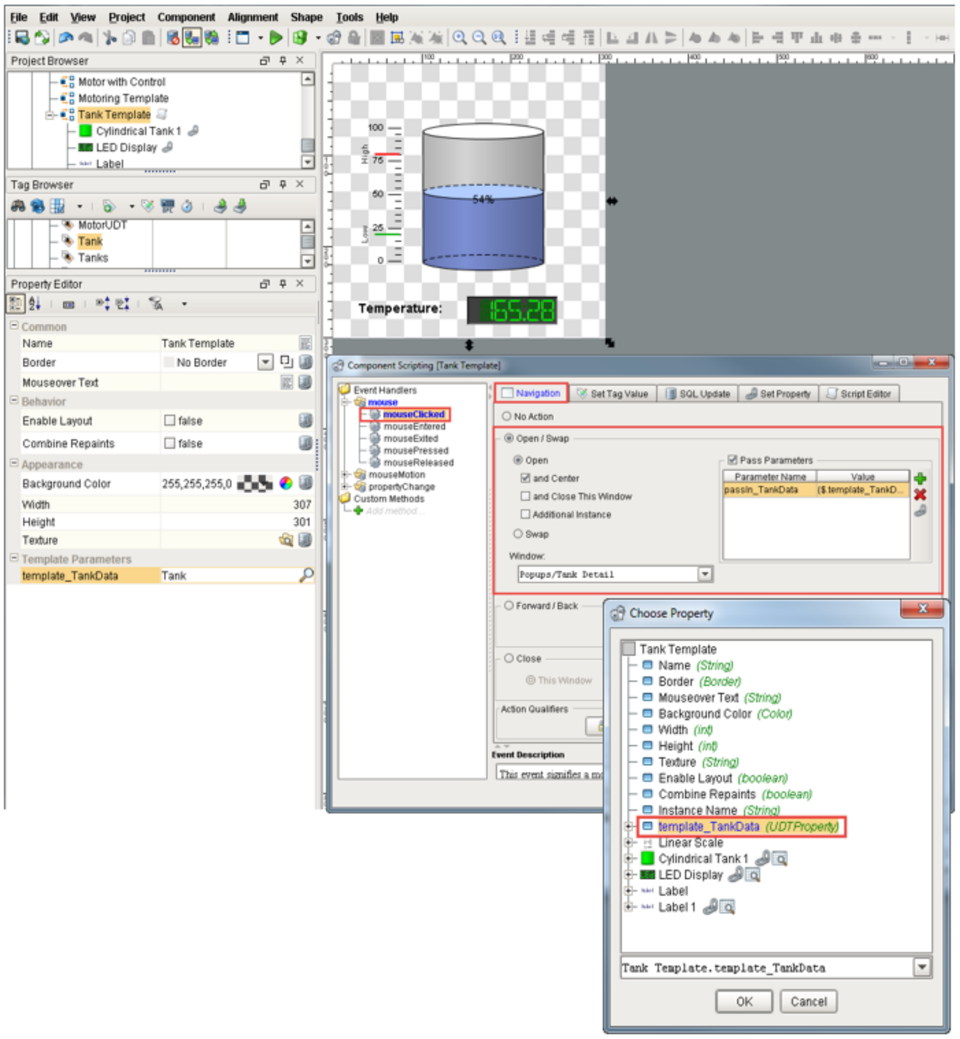
   For example, let's bind the HOA Control. Click on the **Multi-state Button** in your Tank Detail window.
   In the **Property Editor**, click the **binding icon** next to the Control Value to open the Property Binding window.
   Select the **HOA property** of the Tank UDT and **check the Bidirectional box** to make it bidirectional.
   Click **OK**.

   By using the entire Tank UDT and all of its parameters, you have all the properties available to you making it very easy to configure the components in this popup window. You can now bind properties on the window without having to use indirect tag bindings. Bind the rest of your components to the appropriate tags in the Root Container UDT.

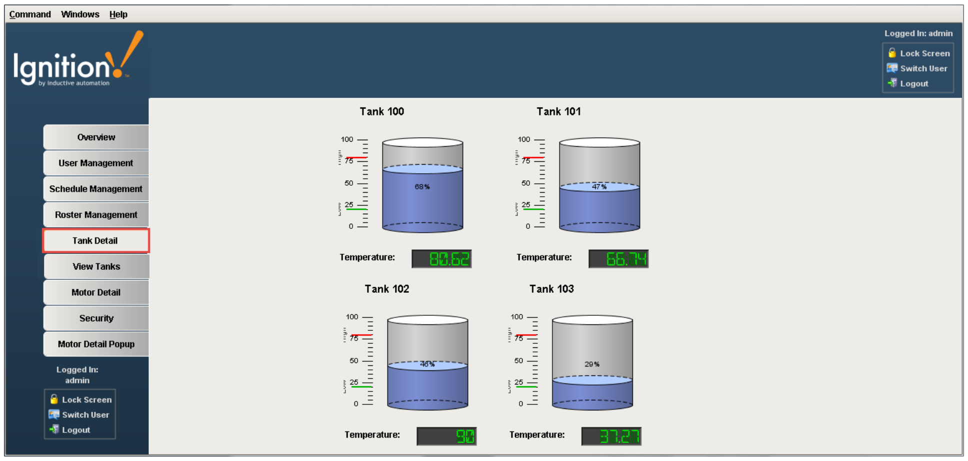


4. Now that there is a property to pass into, we need to set up the script to pass a Tank UDT into the Tank Detail window. A great place to do this is from the Tank template definition. The Tank template is using the same parameter as the Tank Detail window, but for clarity, it was assigned a unique name called **'template_TankData.'**

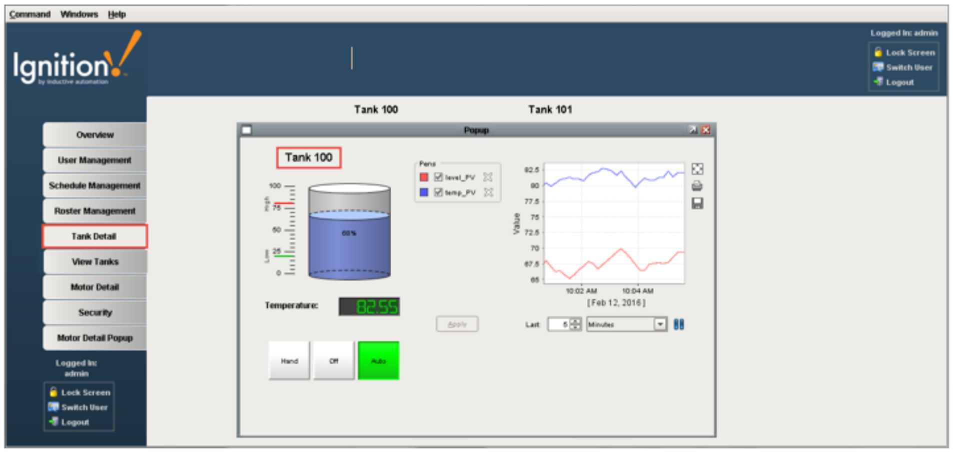
   Navigate to your **Tank Template** and right click on the **gray area of the canvas** and scroll down to **Scripting.**
   Under **Event Handler**, select **'mouseClicked.'**
   In the **Navigation tab**, click **Open / Swap, Open and Center** and under the **Window** dropdown, select your **'Tank Detail'** window.
   Check the **Pass Parameters** box. Press the green plus icon to add a row and click in the **'Parameter Name'** cell. This is a dropdown list that shows all of the custom properties on the Root Container of the selected window. Select the **'passIn_TankData'** parameter to pass in to.
   Select the whole row and click the choose property **link icon** and select the **'template_TankData'** property. The is the value we are passing to the window.
   Click **OK**.



5. Open the **Runtime** and go to any window that uses your tank template. We used the exact same UDT that is in the Tank template and passed it to the Tank Detail Popup window.



Click on one of the Tanks (ie: Tank 100) and it will open your Tank Details window and display all the details about Tank 100. You can do this with any Tank in your runtime.



Next ...

- Property Binding
- Template Repeater Component
- Component and Window Security

Components and Property Binding

Components are visual elements that range in complexity from a single Button to the powerful Easy Chart component. A component gets connected to the real world through its property binding.

In this section ...

Understanding Components

Components are what fill up your windows with useful content. Anyone familiar with computers should already understand the basic concept of a component - they are the widgets that you deal with every day, such as: buttons, text areas, dropdowns, charts, and so on. The Vision module comes with a host of useful components out of the box, many of which are specialized for industrial controls use. Other modules, like the Reporting module, add more components for specialty purposes.

Configuring components will likely be the bulk of a designer's work when designing a Vision project. The basic workflow is to take a component from the palette and drop it into a container on a window. From there, you can use the mouse to drag and resize the component into the correct position. While the component is selected, you can use the **Property Editor** panel to alter the component's properties, which changes the component's appearance and behavior.

Shapes are components too. Each shape may be individually selected, named, and has its own properties. Shapes have some additional capabilities that other components don't have, such as the ability to be rotated. Shapes are created using the shape tools, not dragged from the component palette.

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.

In this section ...

Component Overview

Components are what fill up your windows with useful content. They are the widgets that you deal with every day – buttons, text areas, dropdowns, charts, and so on. The Vision module comes with a host of useful components out of the box, many of which are specialized for industrial controls use. All components have position, layout and other customizers, some of which are highly specialized for the individual component. Other modules, like the Reporting module, add more components for specialty purposes.

Component Overview

Watch the Video

Next ...

- Creating Components

Creating Components

Use the different tools in Ignition such as components, shapes, images, Symbol Factory graphics, and Scalable Vector Graphics (SVGs) to create the components.

There are two primary methods for creating components:

1. Select the component in the palette, and then use the mouse to draw a rectangle in a container. While a component is selected in a palette, the mouse curser will be a crosshair (+) when hovering over a container that the component can be dropped in. Draw a rectangle in the container to specify where the component should be placed and what size it should be.
2. Drag a component's icon from a palette onto a container. The component will be placed where you dropped it at its default size. It can then be resized.

Creating Components

Watch the Video

Choosing the Component Palette

There are two styles of component palette in Ignition Vision: the tabbed palette and the collapsible palette. These palettes work in the same way, but the tabbed palette is meant to dock to the north or south edge of the workspace, and the collapsible palette is meant to dock to the east or west edge. By default, the tabbed palette is visible in the window workspace. To switch palettes, navigate to the **View > Panels** menu, and select the **Tabbed Palette** or the **Collapsible Palette**.

Creating a Custom Palettes

Custom palettes are like expanded copy/paste clipboards. You can put customized components or groups of components into a palette for quick access.

To create a custom palette, right-click on a tab in the tabbed palette or a header in the collapsible palette, and choose New Custom Palette. Your custom palette will appear as the last palette. Your custom palette has one special button in it, the capture button (). To add components to your palette, select them and press the capture button. This effectively does a copy, and stores the captured components as a new item in the clipboard. You can then use that item much like a normal component, and add multiple copies of it to your windows.

Note that these are simple copies, and are not linked back to the custom palette. Re-capturing that palette item will not update all uses of that item across your windows.

Creating Components using Tags

Components can also be created by simply dragging a tag onto a container. Depending on the datatype of the tag, you will get a popup menu prompting you to select an appropriate type of component to display or control that tag.

For example, suppose you have an Int4 type tag, if you drag the tag from the Tag Browser panel onto a component, you will be prompted either to display or control the tag with a variety of labels, level indicators, numeric entry fields, and control buttons.

This technique is great for beginners and for rapid application design. By dropping a tag into a container and choosing a component type, a few steps are happening:

1. The component that you chose is created at the position you dropped it.
2. A variety of property bindings are created automatically.

The bindings depend on what kind of tag was dropped and what kind of component was created. For example, lets suppose you have a Float8 point that represents a setpoint, and you want to set it. Drop the tag onto a container and choose to control it with a Numeric Text Field. The following bindings will be set up automatically:

- The text field's **doubleValue** property gets a bidirectional tag binding to the tag's **Value** property.
- The text field's **minimum** and **maximum** properties get tag bindings to the tag's **EngLow** and **EngHigh** properties, respectively.
- The text field's **decimalFormat** property gets a tag binding to the tag's **FormatString** property.
- The text field's **toolTipText** property gets a tag binding to the tag's **Tooltip** property.

It is important to realize that multiple property bindings are created when creating components this way. These bindings not only using the tag's value, but much of the tag's metadata as well. Using the tags metadata in this way can greatly improve a project's maintainability. For example, if you decide that the setpoint needs 3 decimal places of precision, you can simply alter the tag's **FormatString** to be `#,##0.000`, and anywhere you used that tag will start displaying the correct precision because of the metadata bindings.

See also: Property Binding, Tag Metadata Properties

Creating Components using Shapes

All of the shapes that you can draw using the shape tools are themselves components. As such, they have properties, event handlers, names, layout constraints, and all of the other things that you'll find on other components. They also have some things that normal components don't.

Binding Shape Position

One such thing that shapes have that normal components don't is a set of properties that control their location. These properties are called relX, relY, relHeight, and relWidth. The "rel" prefix stands for "relative". This comes from the fact that the values of these properties are always treated as relative to the shape's parent container's width and height, even in a running Client where that container may be a wildly different size due to the layout mechanism.

For example, let's say that you have a shape that is located at x=100, y=100, and was 125 by 125 inside a container that is 500 by 500. If you want to animate that shape so that it moves back and forth across the screen, you'd set up a binding so that relX changed from 0 to 375. (You want X to max out at 375 so that the right-edge of the 125px wide shape aligns with the right edge of the 500px container).

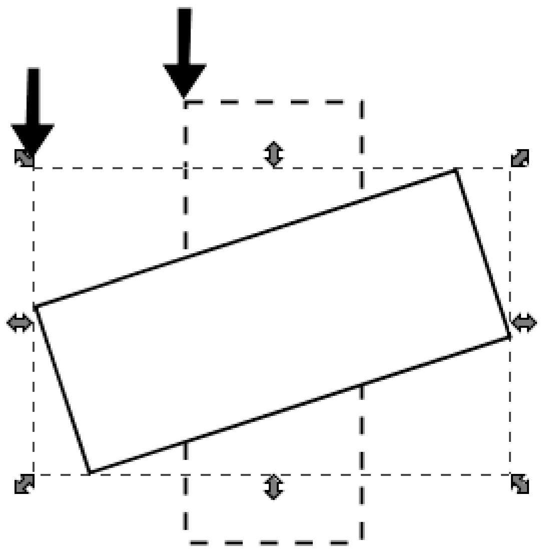
Now, at runtime, that container might be 1000 by 1000 on a user's large monitor. By binding relX to go between 0 and 375, the true X value of your shape (whose width will now be 250px due to the relative layout system), will correctly move between 0 and 1750, giving you the same effect that you planned for in the Designer.

Long story short, using the rel* properties let you animate the shape using bindings and not worry about the details of the layout system and how they'll resize the coordinates at runtime.

Binding Rotation

Another ability unique to shapes is the ability to be rotated. Simply click on a selected shape and the resize controls become rotate controls. There's even a rotation property that can be edited directly or bound to something dynamic like a tag.

Binding the rotation comes with one big warning: observe that when you change a shape's rotation, its position also changes. The position of any shape is the top-leftmost corner of the rectangle that completely encloses the shape.



Because of this effect, if you wish to both dynamically rotate and move a component, special care must be taken since rotation alters the position. You don't want your position binding and the rotation binding both fighting over the position of the component. The way to both rotate and move a shape is as follows:

1. Bind the rotation on your shape as you wish.
2. Create a shape (for example, a rectangle) that completely encloses (in other words, it's bigger than) your shape at any rotation angle.
3. Set that rectangle's visible property to false.
4. Select your shape and the rectangle and group them.
5. Bind the position on the resulting group.

If you follow these steps you can animate both the rotation and position of a shape.

Next ...

- Scalable Vector Graphics (SVGs)

Scalable Vector Graphics (SVGs)

Ignition can import SVGs (Scalable Vector Graphic) into a window. Once imported, SVGs can be modified and styled. To use an SVG in your project, simply drag the SVG file directly onto the window you want the SVG to appear on. The SVG becomes a new polygon component on the window.

Sometime the way the SVG imports may result in the SVG appearing very small, in which case you can manually expand the SVG to your desired size.

> Some SVGs make use of additional effects , such as Gaussian blurs. These SVGs are not supported, and can not be imported.

Scalable Vector Graphics (SVGs)

Watch the Video

Using the Symbol Factory

If you have the Symbol Factory module installed, you've got nearly 4,000 industrial symbols at your fingertips to use in your projects.

To open the Symbol Factory Browser

1. Launch the Designer and open your project.
2. Choose **Symbol Factory** under the **Tools** menu or the project navigation tree.
   If you can't find these, the Symbol Factory module probably isn't installed.
   The Symbol Factory browser opens as a pop-up window that stays on top of the Designer.
3. Browse the different categories to explore what symbols are available, or search to find a specific symbol.
4. Find a symbol that you'd like to use and drag it onto an open Vision window.
   The symbol will become a group of shapes. Just like any group, you can double-click it to get to the shapes inside, or simply un-group it. This way you can edit the symbol if you need to.

   Note: If the Symbol Factory module is in trial mode, you'll only be able to use the first symbol from each category.

Next ...

- Images (png, jpg, gif)

Images (png, jpg, gif)

Images such as PNGs, JPGs, or GIFs can be imported and used inside of windows in Ignition.

To use an image on a window

1. Drag an **Image** component onto the window.
2. In the image component's **Image Path** property, click on the **Browse** icon.
3. From the **Image Browser**, find an image file you want and double-click on it to display it in your window.
   The path to the image is now displayed in the components **Image Path** property.

Image Manager

In addition to the method described above, you can also import images using the Image Manager available from the **Tools > Image Management** menu.

Exporting and Importing Projects

When the project is exported, the images are not included only the path associated with the Image Path property of the image component is. When you import the project into a different Ignition server, the file paths associated with the image components will not have any images to display. It is possible to export the images to a local disk that can accompany the project. The export function is located on any Browse icon associated with Image Path properties.

Ignition does not support bitmap files.

INDUCTIVE UNIVERSITY

Images (png, jpg, gif)

Watch the Video

Next ...

- Selecting Components

Selecting Components

There are a number of different ways to select components within a window, each of which have their own advantages.

Mouse Selection

Using the mouse is the most common way to select components. Make sure that the selection tool (⇱) is the active tool. Simply click on a component to select it. If the component you want to select is obscured by other components, hold-down **Alt** and keep clicking, the selection will step down through the z-order.

You can also select components using window-selection. Click-and-drag in a container to draw a selection rectangle. If you drag the window left-to-right, it will select all components that are contained within the rectangle. If you drag the window right-to-left, it uses window-crossing selection. This will select all components that are contained within the rectangle or intersect the edge of the rectangle. Lastly, you can start dragging a window selection and then hold-down the **Alt** key to use touch selection. This will draw a line as you drag, and any components that the line touches will become selected. As you're using these techniques, components that are about to become selected will be given a yellow highlight border.

Tree Selection

By selecting nodes in the **Project Browser** tree you can manipulate the current selection. This is a handy way to select the current window itself, which is hard to click on since it is behind the Root container. However, you can click to it, using **Alt-click** to step down through the z-order. It is also the only way to select components that are invisible.

INDUCTIVE UNIVERSITY

Selecting Components

Watch the Video

Next ...

- Manipulating Components

Manipulating Components

Manipulating components can be done with both the mouse and the keyboard. You can move components around, resize them, and rotate them.

Resizing

When you select the component you want to resize, they'll get 8 resize-handles displayed around the edge of the selection. These handles look like double-sided arrows around the perimeter. Use the mouse to drag them to change the size of the components in the selection. To maintain the selection's aspect ratio, hold down **Ctrl** as you resize. To resize around the center of the current selection, hold down **Shift**.

You can also resize the current selection using the keyboard. To nudge the right or bottom edge of the selection in or out, use **Shift** combined with the arrow keys. To nudge the top or left edge of the selection, use **Ctrl-Shift** combined with arrow keys. These nudge actions will resize one pixel at a time. To resize faster, add the **Alt** key.

Moving

To move the component, simply drag it anywhere within the component's bounds. You can also move whatever is currently selected by holding down **Alt** while dragging, regardless of whether or not the mouse is over the current selection. This is important because it is the primary way to move a Container component. (Normally, dragging in a container draws a selection rectangle inside that container).

While a component is selected, you may also use the keyboard's arrow keys to move a component around. The arrow keys will move the selection one pixel at a time. Just like resizing with the arrow keys, to move faster, add the **Alt** key. Components can be easily duplicated by dragging them as if you were going to move them and holding down the **Ctrl** key. This will drop a copy of the component at the desired drop location. It is often useful to also hold down Shift as you do this to ensure exact alignment. You may also use the **Ctrl-D** shortcut to quickly duplicate a component in place.

Rotating

Shapes can be rotated directly using the selection tool. Other components cannot be rotated in this manner. To rotate a shape, first select it using the selection tool so that you see the resize handles around it. Then simply click on it once again and you'll see the rotation handles appear. Clicking (but not double-clicking) on selected shapes toggles back and forth between the resize handles and the rotation handles.

Once you see the rotation handles, simply start dragging one to rotate the shape or shapes. Holding down the **Ctrl** key will snap your rotation movements to 15° increments. When the rotation handles are present, there is also a small crosshair handle that starts in the middle of the selection. This is the rotation anchor: the point that the selection will rotate around. You can drag it anywhere you'd like to rotate around a point other than the center of the shape.

INDUCTIVE UNIVERSIT

Manipulating Components

Watch the Video

Next ...

- Component Properties

Component Properties

Each component has a unique set of properties. A property is simply a named variable with a distinct type that affects something about the component's behavior or appearance. Hover your mouse over the property in the **Property Editor** panel to see its data type and scripting name.

INDUCTIVE UNIVERSITY

Component Properties

Watch the Video

The Property Editor

The **Property Editor** is a dockable panel that appears in the Designer's central workspace, usually under the **Tag Browser**. It displays the properties of the selected component. If more than one component is selected, it will show all properties that the current selection set have in common.

Filters

It is common for components to have many properties, so by default, the Property Editor only shows the basic properties. These are the properties that you'll most commonly want to set or bind for a given component. There are also the standard properties. This is a larger set of properties that includes the basic properties and many other useful properties. Some properties are expert properties. These are properties that are either uncommon to set or whose purpose might require an in-depth understanding of the inner-workings of the component. You can change the filter using the **filter** butt on ( ) in the **Property Editor**'s toolbar.

Status Indication

The name of a property in the **Property Editor** conveys important information about that property:

- A blue name indicates that the property is a custom property.
- A **bold name** with a link icon indicates that the property is bound using a property binding.
- A **bold name** with a color palette icon indicates that the property is being affected by the component styles settings.
- A **red bold** name with a warning icon indicates that the property is double-bound. This means that two things, a property binding and the styles settings are both trying to drive the property value. This is almost assuredly a mistake.

The Binding Button

To the right of most properties is the binding button ( ). Use this button to modify the property binding that is driving that property. You can only use this button when the window workspace is not in preview mode. Some properties cannot be bound because their datatype is not supported by the binding system. You can still use scripting to affect these properties.

Data Types

There is a wide variety of datatypes across all of the Vision Module's components. Each property has a distinct type. Here are the most common types that you'll find on the **Property Editor**.

Numeric Types	
`Boolean`	A true/false value. Modeled as 0/1 in Python. Technically, 0 is false and anything else is true.
`Short`	A 16-bit signed integer. Can hold values between -215 and 215-1. Thats -32,768 to 32,767, inclusive.
`Integer/int`	A 32-bit signed integer. Can hold values between -231 and 231-1. Thats -2,147,483,648 to 2,147,483,647 inclusive.
`Long`	A 64-bit signed integer. Can hold values between -263 and 263-1. Thats -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 inclusive.
`Float`	A 32-bit signed floating point number in IEEE 754 format.
`Double`	A 64-bit signed floating point number in IEEE 754 format.
Non-Numeric Types	
`String`	A string of characters. Uses UTF-16 format internally to represent the characters.

`Color`	A color, in the RGBA color space. Colors can easily be made dynamic or animated using Property Bindings or Styles.
`Date`	Represents a point it time with millisecond precision. Internally stored as the number of milliseconds that have passed since the "epoch", Jan 1st 1970, 00:00:00 UTC.
`Dataset`	A complex data structure that closely mimics the structure of a database table. A Dataset is a two-dimensional matrix (also known as, a table) of data organized in columns and rows. Each column has a name and a datatype.
`Font`	A typeface. Each typeface has a name, size, and style.
`Border`	A component border is a visual decoration around the component's edges. You can make a border dynamic by using **Styles**, the **toBorder()** expression function, or Java scripting.

What is the difference between Integer vs. Int?

The difference is that an Integer property will accept the special null (or None in Python-speak) value, while an int property will not. This distinction holds true for all of the numeric types: the type name that starts with a capital letter accepts null, while the all-lowercase version does not.

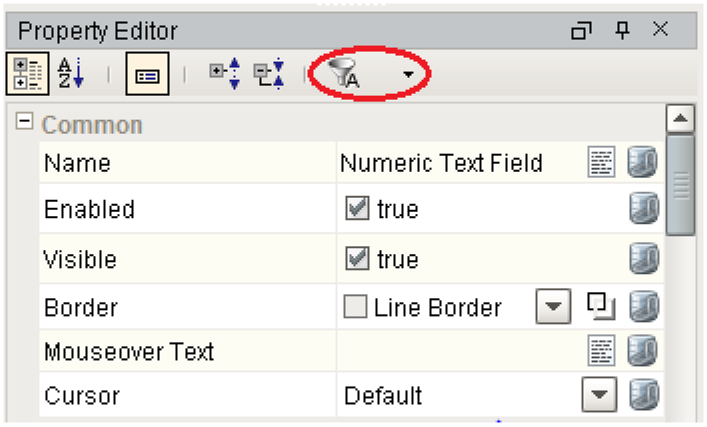
Expert Tip!

Most of these datatypes are actually defined by Java. For example, the **Date** datatype is really an instance of a **java.util.Date**. This means that you can use the **java.util.Calendar** class to manipulate them, and the **java.text.SimpleDateFormat** class to format and parse them. Learn more about these classes in the Java 2 Platform online documentation at http://java.sun.com/j2se/1.5.0/docs/api/index.html

See also: Working with Different Datatypes

Common Properties

Every component has properties arranged into groups based on what his has available. IE: Common, Data, Appearance, etc. Each component has a different list of properties to effect how it behaves, but every component has the **Common** group or properties at the top. Note: When you first open the designer, the property editor is set to only show basic properties. To see all of them, change the property filter to **All**.



Name	Numeric Text Field
Enabled	true
Visible	true
Border	Line Border
Mouseover Text	
Cursor	Default

These **Common** properties will behave the same for all components. Here's a list for each property and when it might be used.

Name

The name of the component. This string is used to identify your components in the Project Browser. This is especially important for Bindings and Scripting. Binding is allowed on this property but it is recommended to **never bind this property.** Binding it can break your scripts, bindings, and cause errors.

Enabled

This Boolean controls whether a component can be interacted with. Most commonly used with data entry components to allow the user to see the value but not change it.

Visible

This Boolean controls whether the component is shown on the window. You can bind this property to show/hide the component based on any logic you want. IE: security, process step, etc.

Border

The border that surrounds the component. There is a dropdown to select from a list of common borders, and a button to the right to manually edit a border from several different options with a second tab that shows Titled Borders. When binding this property, note that this is a complex data type. It is a Java **Border** data type, not a string or an enumeration. The common ways to make this property dynamic are to bind it with an Expression binding type or to set it through a script, but using the Expression binding is preferred. If you are using an Expression binding, you must use the **toBorder()** expression function to return the correct data type. If you are using a script, you need to make sure you use the Java **Border** data type. See the Java documentation for more information on setting a border through scripting.

Mouseover Text

The text that is displayed when a user moves the mouse over the component. This string is commonly used to provide your operators more information about an object. IE: showing the PLC address of an on-screen value, or telling the operator exactly what will happen when a button is pressed.

Cursor

The mouse pointer image to use when the operator moves the mouse over the component. This int property is an enumeration of a list. That is; a value of 0 gives the first option ('default'), 1 is the second, and so on. Selecting 'default' means the operating system decides what pointer to use.

Next ...

- Custom Properties

Custom Properties

You can add your own custom properties to the basic component properties in Ignition. Think of these custom properties as your own variables in the window. The custom properties are extremely useful as they let you turn a "plain" component into one customized for your particular use.

Ignition provides a number of **Customizers** to configure components in ways that are too complex or cumbersome for basic properties. In this page, the **Customizers > Custom Properties** and the **Customizers > Styles Customizer** are described. These two Customizers are used repeatedly for many different components. But there are other Customizers that are unique for a specific component, such as the Easy Chart Customizer.

Custom Properties

Watch the Video

Expert Tip

Often, a Customizer is really just a user-friendly user interface to one or more expert properties. For example, all the **Easy Chart** Customizer really does, is to modify the contents of the pens, tagPens, calcPens, axes, and subplots Dataset properties. Knowing this is very powerful, because this means you can also use Property Bindings and scripting to modify the values of these expert properties at runtime, giving you the ability to dynamically perform complex manipulations of components.

To add a custom property, right-click on the component and choose the **Customizer** menu, or use the Customizer split-button ( ) icon in the Vision main toolbar.

Root Container Custom Properties

Properties on the window's Root Container are special in that they double as a window's parameters. For more information, see Parameterized Popup Windows.

Configuring Custom Properties

You can use the custom properties like any other properties - with data binding, scripting, styles, and so on.

To configure the custom property

1. Right-click on a component and go to **Customizer > Custom Properties**.
   The Custom Properties window is displayed.
2. Click the **green +** icon to add text boxes to the table.
3. Add a **Name**, **Type**, and **Description** for each custom property you want to add.
4. Click **OK**.
   Notice the custom properties you added are displayed in the **Property Editor** in blue.

Configuring the Style Customizer

Many components support the **Style Customizer** which lets you define a set of visual styles that change based on a single **Driving Property**. Typically, you'll have a property (often a custom property) on your component that you want to use as a driving property, usually a discrete state, and you have multiple visual properties, like the font, border, foreground color, visibility, and so on that you want to change based on that one driving property. Style Customizer lets you define these relationships all at once, and lets you preview them too!

To configure the style using the Style Customizer

1. Right-click on a component and go to **Customizer > Style Customizer**.
   The Style of Component window is displayed.
2. Choose a **Driving Property**.
   This is the property whose value will determine the current style.
3. Choose one or more **Styled Properties**.
   These are the properties that will change as the style changes.
4. Add the values of the driving property that define the styles.
   For example, 0=off, 1=hand, 2=auto.
5. Customize the values of the styled properties for each style.

Example 1

Lets say that you have a Level Indicator component that is displaying the level in a tank. Lets say that you want to have its appearance change based on the alarm state of the tank's temperature. You can add an integer custom property, call it **Severity**, to the level indicator that you'll bind to the tank temperature tag's **AlertCurrentSeverity** property.

Now go to the **Style customizer**. Choose your **Severity** property as the driving property, and the **Border** and **Filled Color** properties as the styled properties. Add states for -1 (not in alarm), 2 (Medium alarm) and 4 (High alarm). Leave the -1 state alone. Use a red border for state 2 and an orange fill color. For state 4, you can animate it to get a flashing effect. Add two animation frames and set their delay to 500ms each. Configure the frames differently from each other so that you can get a flashing effect. Click **OK** and notice that the styled properties you chose are now bold and have the styles indicator (●) next to them. This is to help remind you that those properties are being driven, so if you change their values directly, you changes will be overwritten.

Example 2

Let's look at another example that will use the Custom Properties and the Styles feature together. Take the lowly Label component. It seems pretty plain at first - it just displays a String. Of course, you can use its foreground color, background color, and border to make it look interesting. Add an integer custom property on it and name it **state**, and bind that property to a discrete state tag coming out of a PLC. Now use the **state** property to drive its Styles configuration to make the component look different and display different things based on the state being 0, 1, or 2 (maybe for a Hand/Off /Auto indicator).

We could have used the Multi-State Indicator component from the very beginning, but understanding this example will let you create your own types of components by combining the existing components in creative ways.

Next ...

- Component Grouping

Component Grouping

Shapes and components can be grouped together so that they act like a single component in the Designer. Grouping components is very similar to putting them in a Container. In fact, it is the same thing as cutting and pasting them into a perfectly-sized container and then putting that container into group mode, with one exception. If the group contains only shapes and no other kinds of components, it will be a special shape-group that has the ability to be rotated and has some other shape-like properties.

When components or shapes are in a group, clicking on them in the Designer will select the group instead of the shape. If you double-click on a group, it will become "super-selected", which will allow you to interact with its contents until you select something outside of that group.

Layout also works differently for groups. The layout setting for components and shapes inside a group is ignored. All members of a group act as if they are in relative layout with no aspect ratio restrictions. This special group-layout mode is also active when resizing a group inside of the Designer, whereas traditional layout doesn't take effect in the Designer.

Groups can contain other groups, creating a nested structure. Groups themselves are also components, meaning that you can add custom properties to groups, bind them, and so on.

Component Grouping

Watch the Video

Next ...

- Component Layout

Component Layout

Layout is the concept that a component's size and position, relative to its parent container's size and position, can be dynamic. This allows the creation of windows that resize gracefully using either **Anchored** or **Relative** layouts and can optionally keep the original aspect ratio.

This is a very important concept because of the web-launched deployment of Vision clients - they often end up being launched on many different monitors with many different resolutions.

This is also important for components that have user-adjustable windows like popup windows. Imagine a popup window that is mostly displaying a large table or chart. If you're running on a large monitor, you may want to make the window bigger to see the table or chart easier. Of course, this is only useful if the table or chart actually gets larger with the window.

Changing a component's layout is as simple as right-clicking on the component and opening the Layout dialog box. You can also alter the default layout mode that gets assigned to new components. See Designer/Window Editing Properties.

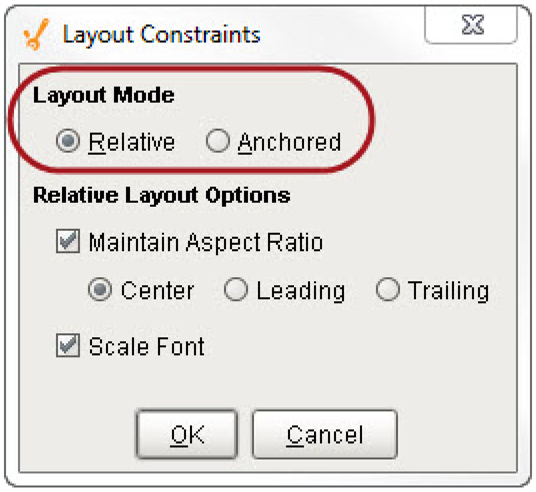
INDUCTIVE UNIVERSITY

Component Layout

Watch the Video

Relative vs Anchored Layout

There are two layout modes, and they are set on a per-component basis. Both affect the component's size and position relative to its parent container. The root container's size is dictated by the window size. To edit the layout of a component, right-click on the component and select **Layout** from the menu. the Layout Constraints window displays showing all the default settings. These default settings can be altered in the Project Properties.



Layout Constraints

Layout Mode

[✓ Relative] Anchored

Relative Layout Options

[✓ Maintain Aspect Ratio]

[✓ Center] Leading Trailing

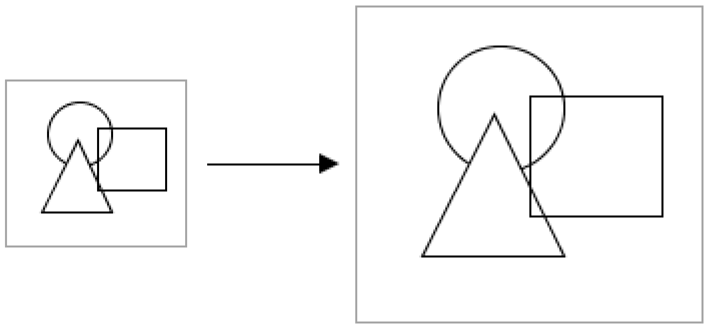
[✓ Scale Font]

Layout Modes

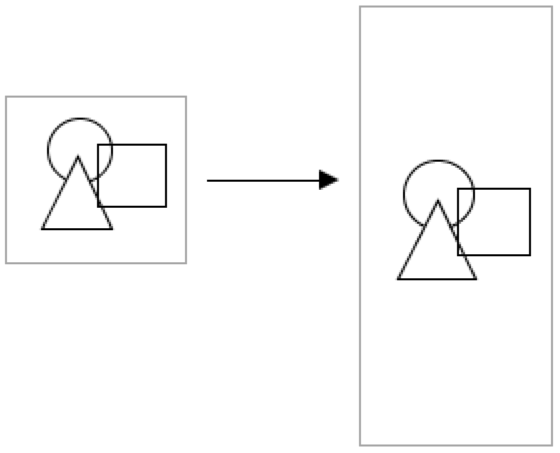
- Relative
  This mode makes a component's size and location relative to its parent's size and location. When the parent changes size, the component changes accordingly. This creates components that auto-scale.
- Anchored
  This mode makes the edge of a component's two axes (horizontal and vertical) anchored to the edge or edges of its parent.

Relative Layout

Relative layout is the default mode. This is a simple and effective layout mode that simply keeps a component's size and position constant relative to its parent container, even when the parent container grows or shrinks. More precisely, it remembers the component's position and size as a percentage of its parent's bounds at the last time the window was saved. Relative layout also has the option of scaling a component's font appropriately.



Note that relative layout mode respects aspect ratio. So if the parent component is distorted, the contents will not be. The extra space is distributed evenly on both sides of the contents.

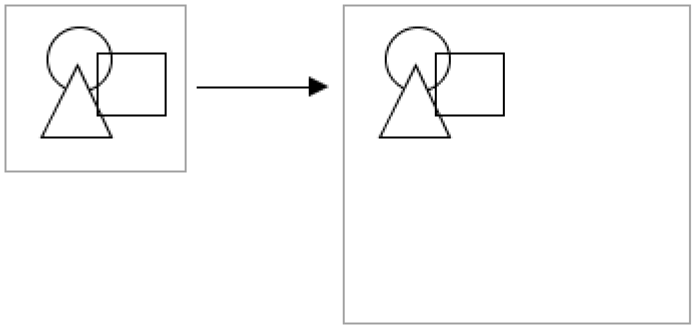


Relative Layout Options

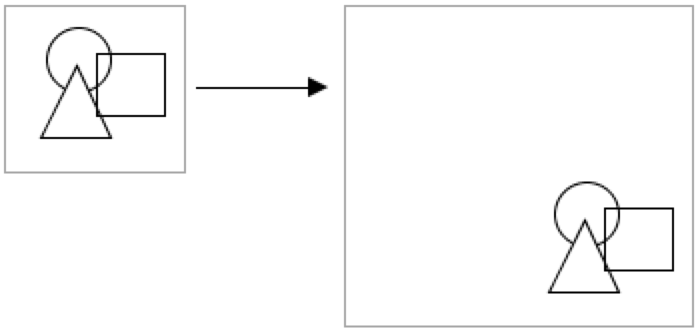
- Maintain Aspect Ratio
  If selected, the component's original aspect ratio is preserved. Otherwise, it can stretch tall or wide.
- Center
  When maintaining the aspect ratio, centers the component with respect to its parent.
- Leading
  When maintaining the aspect ratio, aligns the component with the paper leading edge.
- Trailing
  When maintaining the aspect ratio, aligns the component with the parent's trailing edge.
- Scale Font
  If selected, the component's font will scale along with its size as the relative layout adjusts the component. This will override other font size settings. If this setting is applied to a Group, then all components in the group will use this setting.

Anchored Layout

Anchored layout lets you specify various "anchors" for the component. The anchors dictate how far each of the 4 edges of the component stay from their corresponding edges in the parent container. For example, if you anchor top and left, then your component will stay a constant distance from top and left edges of its parent. Since you didn't specify an anchor for the right or bottom sides, they won't be affected by the layout.



If you anchor bottom and right instead, the components will again stay the same size (since you didn't specify an anchor for their other edges, but they will stay a constant distance from their parent's right and bottom edges.



Anchored Layout Options

- North/South
  If one of these is selected, the distance between that edge of the component and that edge of the container is preserved. If both are selected, the component will stretch its **height** to maintain both distances.

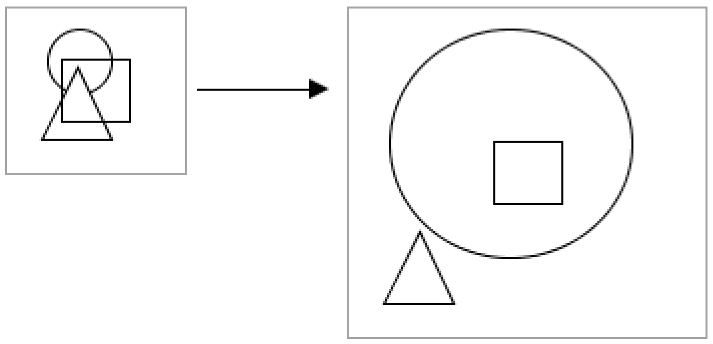
- West/East
  If one of these is selected, the distance between that edge of the component and that edge of the container is preserved. If both are selected, the component will stretch its **width** to maintain both distances.

- Center Vertically
  When selected, both top and bottom buttons will be deselected. This option maintains the height of the component and centers it **vertically** in the container.

- Center Horizontally
  When selected, both left and right buttons will be deselected. This option maintains the width of the component and centers it **horizontally** in the container.

Of course, you can mix and match the various modes for the different components in a given container. The following image shows the following:

- The **square** uses a horizontal and vertical centering anchor. It is centered, and stays the same size.
- The **triangle** is anchored south and west.
- The **circle** is anchored north, west, south, and east. This means that its edges are all anchored and stay a fixed distance to each of its parent's edges, so it grows.



Client Minimum Size

Clients can define a minimum size, because even with dynamic layout, you usually don't want the client to get too small. This is because it would become unusable and unreadable. This is what the Minimum Size property is for. By default it is set to 800x600, but you can adjust it. For more information, see Client/User Interface Properties.

Next ...

- Multi-Line Labels and Buttons
- Template - Resizing and Enable Layout

Multi-Line Labels and Buttons

Labels and Button's have single lines of text by default. If the text of the label or button exceeds the space the component has made available, the component cuts the text off and trails it with three dots.

You can use HTML to format the text to have line wrapping for a long line of text, and to insert a line break any place inside a text to have multiple lines of text.

Multi-Line Labels and Buttons

Watch the Video

To auto line wrap the text

1. Select the component that needs the automatic line wrapping.
2. Select the component's Text property.
3. Insert `<html>` before the start of the text and press enter.

To insert a line break inside the text

1. Select the component that needs the line break.
2. Select the component's Text property.
3. Insert `<html>` before the start of the text.
4. Select the part of the text that you want to see the break occur and type `<br>` in this location and press enter.

Next ...

- Adding Icons to Labels and Buttons

Adding Icons to Labels and Buttons

Most components have an **Image Path** property, which augments the text property of a component by placing an image next to it. The image can be an icon that is included with Ignition or an image such as a .jpeg or .png file.

The location of the image in relationship to the component's text is configured in the component's layout section of its Property Editor.

Adding an Icon to a Label

1. Create an Label on your window.
2. Go to the component's Property Editor and select the folder icon from the Text property.
3. From the resulting popup, navigate to the icon of your choice and double-click it using your mouse.

The component's Horizontal Text Position and Vertical Text Position properties will change the location of the icon in relationship to the text.

Adding Icons to Labels and Buttons

Watch the Video

Next ...

- HTML in Ignition

HTML in Ignition

Many Ignition components display a text string. By default, a component's text is displayed in a single font and color and will not wrap when its content exceed the space the component has made available to the text.

HTML stands for HyperText Markup Language. It is commonly used to style text within webpages. The features that HTML brings to style web pages can be applied to many components within Ignition to style the text within components.

You can use HTML if you want to mix fonts or colors within the text or if you want formatting such as multiple lines. HTML formatting can be used in Ignition components such as buttons, labels, and tables. It can be used in common properties such as the mouse over text property.

To specify that a component's text has HTML formatting, just put the `<html>` element at the beginning of the text, then use any valid HTML element in the remainder.

> Closing the HTML element is optional. In other words there is no need to place a </html> at the end of your stylized text.
>
> Also, the HTML elements are not case sensitive.

INDUCTIVE UNIVERSITY

HTML in Ignition

Watch the Video

Common HTML Elements

HTML tags are the special characters that instruct text to become stylized differently than other text within the same text. The following table describes the most common HTML elements that you can use within Ignition.

HTML Element	Name	Description
<html>...</html>		Initiates an html formatting. In most cases closing the html with </html> is optional.
<b>...</b>	Bold	Applies a bold style to the contents of these elements.
<br>	Break	Applies a line break at this specific location.
<li>...</li>	List Item	Used to represent a list item. Should be contained in an order list (<ol>) or unordered list (<ul>).
<ol>...</ol>	Ordered List	Places the text into an ordered list. Text inside list items are ordered by number.
<ul>...</ul>	Unordered List	Places the text into an unordered list. Text inside list items are ordered by bullets.
<u>...</u>	Underline	Underlines the text contained within the elements.
<center>	Center	Centers the contents of the text. Used directly after the HTML tag (that is, <html><center>...)
<font color="red">...</font>	Font	Colors the contents red. Works with standard color names, hex numbers, or RGB numbers.

Applying HTML to Components

In Ignition, you can add HTML to the text property of any component such as, a label, button, or table.

Adding HTML into a Label

Suppose you want to make a label appear like this: This is a **bold** word

1. Create a label on your window.
2. Select the label's text property and enter the following text: `<html><b>This is a <b>bold</b> word`

Adding HTML into a Button

Suppose you want to make a button appear with the following words: Stop the Process**.**

The word "Process" starts on a new line. Also, the words must be centered on the button.

1. Create a button on your window.
2. Select the buttons text property and enter the following text: `<html><center>Stop the <br><font color="red">Process</font>`

Adding HTML into a Component's Mouseover Text Property

The mouse over text property creates a popup when the user places their mouse over the component for an extended period of time. The mouse over text is an excellent way of giving the user additional information. The mouse over text property also handles HTML formatting.

Suppose you want a list of instructions to appear in the mouse over text of a component as the following:

> These are the instructions:
>
> 1. Stop this process.
> 2. Check on this.
> 3. Remove that.

To create this type of mouse over text, do the following:

1. Create or choose a component on your window.
2. Select the components mouse over text property and select the property's text editor icon ( ) to edit the text.
3. Enter the following text:

```
<html>
These are the instructions:
<ol>
<li>Stop the process.</li>
<li>Check on this.</li>
<li>Remove that.</li>
</ol>
```

> Note that this technique allows you to format the text across multiple lines. Line breaks do not occur unless they are explicitly declared with the <br> element or the component wraps the contents because of the lack of space for its contents.

Adding HTML into a table

The contents of a table will render HTML formatting. The HTML formatting has to be part of the specific cell. If you want to have HTML formatting affect all rows of the data then it will be necessary to concatenate the HTML markup into the source of the data that is populating the table.

Suppose that you wanted to edit a cell of a table in order to render its contents with HTML formatting. Simply access the table's underlying data and enter the HTML formatting into the cell that you would like to format. The same techniques used for HTML formatting throughout Ignition apply to the cells of a table as well.

Next ...

- Component Data Types

Component Data Types

There is a wide variety of datatypes across all of the Vision Module's components. Here are the most common types that you'll find.

Numeric Types

Boolean	A true/false value. Modeled as 0/1 in Python. Technically, 0 is falseand anything else is true.
Short	A 16-bit signed integer. Can hold values between -215 and 215-1. Thats -32,768 to 32,767, inclusive
Integer / int	A 32-bit signed integer. Can hold values between -231 and 231-1. Thats -2,147,483,648 to 2,147,483,647 inclusive
Long	A 64-bit signed integer. Can hold values between -263 and 263-1. Thats -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 inclusive
Float	A 32-bit signed floating point number in IEEE 754 format.
Double	A 64-bit signed floating point number in IEEE 754 format.

Non-Numeric Types

String	A string of characters. Uses UTF-16 format internally to represent the characters.
Color	A color, in the RGBA color space. Colors can easily be made dynamic or animated using Property Bindings or Styles
Date	Represents a point it time with millisecond precision. Internally stored as the number of milliseconds that have passed since the "epoch", Jan 1st 1970, 00:00:00 UTC.
Dataset	A complex datastructure that closely mimics the structure of a database table. A Dataset is a two-dimensional matrix (a.k.a. a table) of data organized in columns and rows. Each column has a name and a datatype.
Font	A typeface. Each typeface has a name, size, and style.
Border	A component border is a visual decoration around the component's edges. You can make a border dynamic by using Styles or the toBorder expression.

Whats the difference: Integer vs int? The difference is that an Integer property will accept the special null (or None in Python-speak) value, while an int property will not. This distinction holds true for all of the numeric types: the type name that starts with a capital letter accepts null, while the all-lowercase version does not.

Expert Tip: Most of these datatypes are actually defined by Java. For example, the Date datatype is really an instance of a java.util.Date. This means that you can use the java.util.Calendar class to manipulate them, and the java.text.SimpleDateFormat class to format and parse them. Learn more about these classes in the Java 2 Platform online documentation at http://java.sun.com/j2se/1.5.0/docs/api/index.html

See also:

Working with Different Datatypes

Next ...

- add link here

Drawing Tools

The drawing tools are provided so you can create your own shapes such as lines, rectangles, and circles. These shapes are components with their own set of properties.

In this section ...

Drawing Tools Overview

Shapes or graphics such as lines, rectangles, and circles can be created using the 2D drawing tools in Ignition. All SVG (Scalable Vector Graphics) images are importable in ignition.

INDUCTIVE UNIVERSITY

Drawing Tools Overview

Watch the Video

Shape Tools

Shapes such as lines, rectangles and circles are created using the shape tools. By default the shape toolbar appears alongside the right-hand edge of the Designer window, but you can drag it to wherever you prefer. There are a number of tools here for your use, and each one makes the window editing workspace act differently when active.

Shape tools allow you to create various shapes as well as edit them after they are created. Click on the tool's icon to make it the active tool. You can also double-click on a shape to change to that shape's native tool. When a shape tool is active, a toolbar will appear that has specific actions and settings for that tool.

After a shape is created, you can change its fill color, stroke color, and stroke style. See Fill and Stroke for more information. All shapes can be treated as paths and be used with composite geometry functions to alter or create other shapes. For more information, see Bezier Curves and Editing Shape Paths.

Selection Tool

Normally the selection tool ( ) is active. When this tool is active, you can select shapes and components. Selected components can be moved, resized, and rotated. For more on using the selection tool to manipulate components and shapes, see Manipulating Components.

Rectangle Tool

The rectangle tool ( ) creates and edits rectangle shapes. To create a rectangle, select the tool and drag inside a window to create a new rectangle. Hold down **Ctrl** to make it a perfect square. Once a rectangle is created, you can use the square handles to change the rectangle's height and width. This is important because it is the only way to resize a rotated rectangle and let it remain a rectangle. If you resize a non-orthogonally rotated rectangle using the selection tool, it will skew and become a parallelogram, but if you double-click on it so that the rectangle tool is active, you can change the rectangle's width and height using the tool-specific handles.

There are also small circle handles that allow you to alter the rectangle's corner rounding radius. Simply drag the circle down the side of the selected rectangle to make it a rounded rectangle. Hold down control to drag each rounding handle independently if you want non-symmetric corner rounding. You can use the **Make Straight** button in the rectangle tool's toolbar ( ) to return a rounded rectangle to be a standard, straight-corner rectangle.

Ellipse Tool

The ellipse tool ( ) creates and edits circles and ellipses. It is used in much the same way as the rectangle tool. While it is the active tool, you can drag inside a window to create a new ellipse. Hold down **Ctrl** to make it a perfect circle. When an ellipse is selected, use the width and height handles to alter the shape.

Polygon / Star Tool

The polygon tool ( ) is used to create polygons and stars. Use the toolbar that becomes visible when this tool is active to alter the settings of the shape that is created when you drag to create a polygon.
This tool can be used to make any polygon with 3 corners (a triangle) or more. Once created, you can use the center square handle to move the polygon around, and the diamond handles to alter the size and angle of the polygon. Hold down **Ctrl** to keep the polygon's rotation an even multiple of 15°.

Arrow Tool

The arrow tool ( ) is used to create single or double-sided arrow shapes. When it is active, simply drag to create a new arrow. Use the checkbox on the toolbar to choose a single or double-sided arrow. To alter the arrow, use the diamond handles to change the two ends of the arrow, and the circle handles to change the size of the shaft and the head. When changing the arrow's direction, you may hold down **Ctrl** to snap the arrow to 15° increments.

Pencil Tool

The pencil tool ( ) is used to draw freehand lines and shapes. When this tool is selected, you can draw directly on a window by holding down the mouse button. Release the mouse button to end the path. If you stop drawing inside the small square that is placed at the shape's origin, then you will create a closed path, otherwise you'll create an open path (line).

On the pencil tool's toolbar, there are options for simplification and smoothing, as well as a toggle between creating straight line segments or curved line segments. The simplification parameter is a size in pixels that will be used to decrease the number of points used when creating the line. Points will be in general as far apart as this setting. If you find the line isn't accurate enough, decrease this setting. If you choose to create curved segments, then the segments between points will be Bézier curves instead of straight lines. The smoothing function controls how curvy these segments are allowed to get.

Line Tool

The line tool (⚡) can be used to draw lines, arbitrary polygons, or curved paths. Unlike all of the other tools, you don't drag to create new paths with the line tool. Instead, you click for each vertex you'd like to add to your path.

To draw a straight line, simply click once where you want the line to start, and double-click where you want the line to end. To make a multi-vertex path, click for each vertex and then double-click, press enter, or make a vertex inside the origin box to end the path.

As you draw the line, "locked-in" sections are drawn in green and the next segment is drawn in red. Hold down **Ctrl** at any time to snap the next segment to 15° increments.

On the line tool's toolbar, you can choose between three modes:

- normal line mode
- perpendicular mode
- curve mode

Perpendicular mode is just like line mode except that each segment is restricted to either horizontal or vertical. Curve mode will create a Bézier curve path by attempting to draw a smooth curve between the previous two vertices and the new vertex.

Path Tool

All shapes and paths can be edited directly by using the path tool. This tool lets you directly modify the nodes in the path, adding new nodes, removing nodes, and toggling segments between straight or curved. Learn more about paths in the Geometry and Paths section.

Gradient Tool

The gradient tool is used to affect the orientation and extent of any gradient paints. Learn more about gradients in the Fill and Stroke section.

Eyedropper Tool

The eyedropper tool is used to set the selected shape(s) and/or component(s) foreground/background or stroke/fill colors by pulling the colors from somewhere else in the window. When this tool is active, left-click to set the selection's fill or background, and right-click to set the selection's stroke or foreground. Note that this tool works on most components as well as shapes. For example, right clicking will set the font color on a Button, or left-clicking will set the background color.

Next ...

- Drawing a Line

Drawing a Line

The line tool ( ) lets you create shapes and graphics by drawing lines, arbitrary polygons, or curved paths. Unlike all of the other tools, you don't drag to create new paths with the line tool. Instead, you click for each vertex you'd like to add to your path.

To draw a straight line, simply click once where you want the line to start, and double-click where you want the line to end. To make a multi-vertex path, click for each vertex and then double-click, press enter, or make a vertex inside the origin box to end the path.

As you draw the line, "locked-in" sections are drawn in green and the next segment is drawn in red. Hold down **Ctrl** at any time to snap the next segment to 15° increments.

On the line tool's toolbar, you can choose between three modes:

- normal line mode
- perpendicular mode
- curve mode

Perpendicular mode is just like line mode except that each segment is restricted to either horizontal or vertical. Curve mode will create a Bézier curve path by attempting to draw a smooth curve between the previous two vertices and the new vertex.

Drawing a Line

Watch the Video

Next ...

- Fill and Stroke

Fill and Stroke

All shapes have three properties that affect how they look: Fill Paint, Stroke Paint, and Stroke Style.

- **Fill Paint** is a property of type **Paint** that represents the interior color of the shape.
- **Stroke Paint** property (also a **Paint**) represents the color of the shape's outline.
- **Stoke Style** property is a property of type **Stroke** that affects the thickness, corners, and dash properties of the shape's outline.

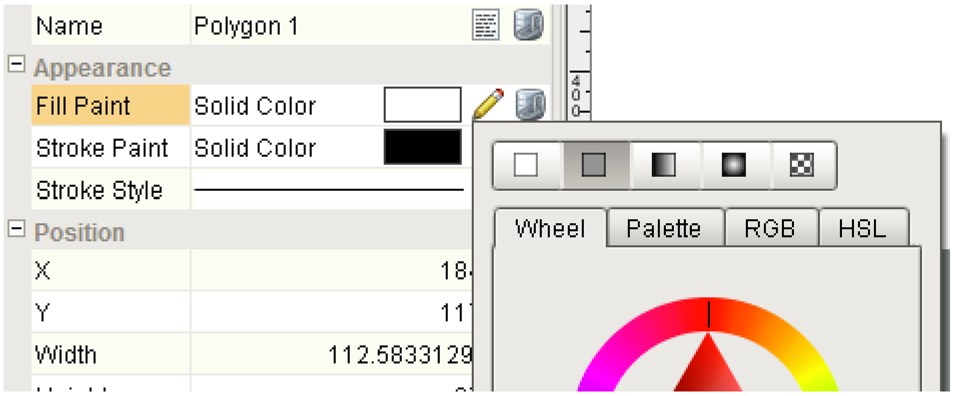
INDUCTIVE UNIVERSITY

Fill and Stroke

Watch the Video

Editing Paints

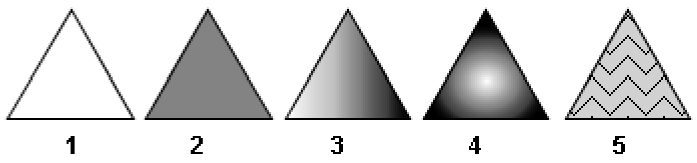
Both the fill and stroke paints can be a variety of different kinds of paints. To edit a shape's fill or stroke paint, you can either use the paint dropdown in the property editor table by clicking on the **pencil** icon ( ) or open up the dedicated **Fill and Stroke** panel from the **View** menu.



Name	Polygon 1
Appearance	
Fill Paint	Solid Color
Stroke Paint	Solid Color
Stroke Style	
Position	
X	18
Y	11
Width	112.5833129

Paint Types

The top of the paint editor is a selection area that allows you to choose between the five different kinds of paints.

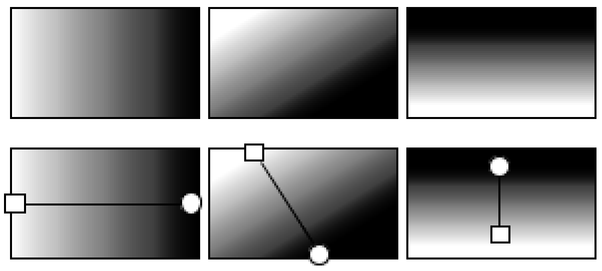


1. The first paint type is no paint ( ). If used as a fill paint, then the interior of the shape will be transparent. If used as the stroke paint, then the paint's outline will not be drawn at.
2. The second paint type is a solid color ( ). This paint type is equivalent to the Color type used elsewhere throughout the component library. A solid color is any color, including an alpha (transparency) level.
3. The third paint type is a linear gradient ( ). Linear gradients smoothly blend any number of colors along a straight line across the shape. Each color is called a Stop. Each stop is represented as a drag-able control on a horizontal preview of the gradient in the gradient editor. You can click on a stop to select it and change its color or drag it to reposition it. You can right-click on it to remove it. You can right-click on the preview strip to add new stops and change the gradient's cycle mode.
4. The fourth paint type is the radial gradient ( ). Radial paints are very much like linear paints except that the colors emanate from a point creating an ellipse of each hue. Radial paints are configured in the same way as linear paints.

5. The fifth paint type is the pattern paint ( ). This paint uses a repeating pixel-pattern with two different colors. You can pick a pattern from the dropdown or create your own using the built-in pattern editor.

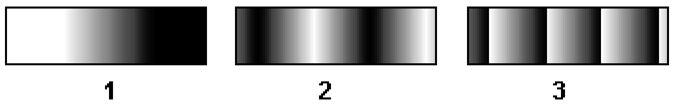
Gradient Paint Bounds

The two gradient paints are more than a list of colored stops, they also need to be placed relative to the shape. The same gradient may look wildly different depending on how it is placed against the shape. By default, a linear gradient will run horizontally across the width of the entire shape, but this is readily changed. By switching to the Gradient Tool ( ), you can drag around handles that control how the gradient is applied.



Gradient Cycles

The two gradient paints (linear and radial) both have a cycle mode that you can change by right-clicking within the preview strip. The cycle modes are illustrated below:



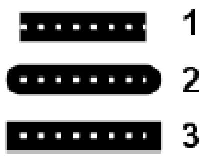
1. **No Cycle**. The first and last stops are repeated forever after the edge of the paint bounds.
2. **Reflect**. Beyond the bounds of the paint, it will be reflected and drawn in reverse, and then reflected again, creating a smooth repetition.
3. **Repeat**. Beyond the bounds of the paint, it will be repeated forever.

Stroke Style

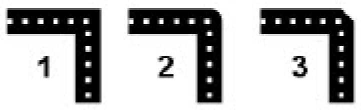
A shape's stroke paint is only half the story. The stroke style is also an important component of how an outline is drawn. Primarily the style controls the thickness of the line drawn, but it also can be used to create a dashed line. The setting for thickness is specified in pixels, and creating a dashed line is as easy as picking the style from the list.

The effect of the thickness and dash pattern settings is fairly self-explanatory, but the other stroke settings are a bit more subtle. You can notice their effect more readily on thick lines.

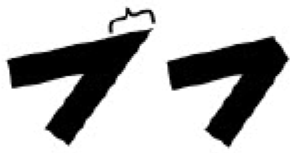
Cap style is a setting that controls what happens at the end of a line segment. You can either have the line simply be terminated with no decoration (#1), round-off the end with a semi-circle (#2), or cap the end with a square (#3).



Join style is a setting that affects how a line is drawn where two segments meet (a corner). The default setting is called a miter join (#1), where the stroke is extended into a point to make a sharp corner. The other options are rounded corners (#2) or beveled edge corners (#3).



Miter style joins can become a problem for very sharp angles. With a sufficiently sharp angle, the miter decoration can become extremely long. To control this, there is a miter length setting to limit the length of a miter decoration. The illustration on the left shows the same miter join with two different miter length settings. The first drawing illustrates the length of the miter join.



Next ...

- Gradients

Gradients

Gradients smoothly blend any number of colors and can be positioned along a straight line or form an ellipse across the shape.

Gradients

Watch the Video

Next ...

- Shape Geometry

Shape Geometry

Shape geometry allows you to apply functions such as a union (combine), difference (exclusion), and intersection (division) on multiple shapes to get the resulting shape that you want.

INDUCTIVE UNIVERSITY

Shape Geometry

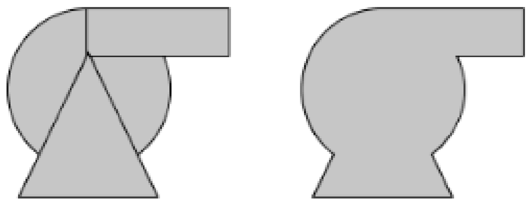
Watch the Video

Creating and Editing Shapes Using Constructive Area Geometry

Editing paths directly can be a bit awkward. Using Constrictive Area Geometry is usually a much easier and more intuitive way to get the shape that you want. These functions are accessed from the **Shape** me nu and operate when two (or more) shapes are selected. Note that the order that you select the shapes is important for many of these functions. Typically, the first shape you select is the shape you want to retain, and the second shape is the shape that you want to use as an "operator" on that first shape.

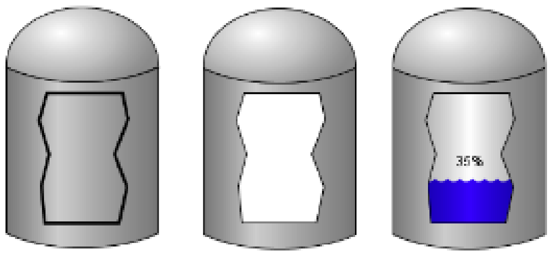
Union

The **union** function combines two or more paths into one. The resulting shape will cover the area that any of the shapes covered initially. The example shows how the union of a circle, rectangle, and triangle can be unioned together to create a basic pump symbol. Creating the symbol using this method took a few seconds, whereas attempting to draw this shape by hand using paths would be quite frustrating.



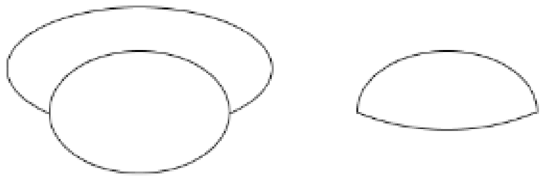
Difference

The **difference** function can be thought of as using one shape as a "hole-punch" to remove a section of another shape. The example shows how a zigzag shape drawn with the line tool can be used to punch a cutaway out of a basic tank shape. The level indicator is added behind the resulting shape to show how the area where the zigzag shape was is no longer part of the tank shape.



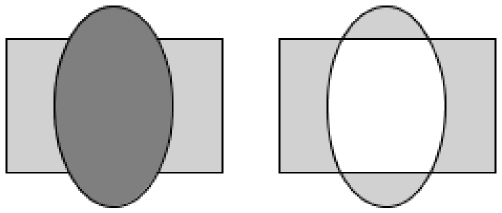
Intersection

The result of an **intersection** function will be the area only where where two shapes overlap. The example shows how the "top" of the tank in the difference example was easily made using two ellipses.



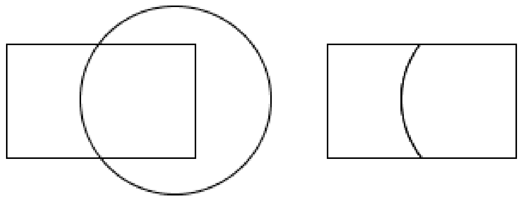
Exclusion

The **exclusion** function, sometimes called X-OR, creates a shape that occupies the area covered by exactly one of the source shapes, but not both.



Division

The **division** function divides or cuts one shape up along the outline of another shape.

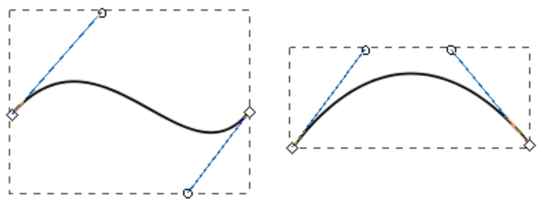


Next ...

- Bezier Curves

Bezier Curves

A Bézier curve, also sometimes called a quadratic curve, is a type of curved line used in vector graphics that connects two points, allowing you to create smooth vector graphic shapes. A Bézier curve is configured using four points: the two end-points and two control points. The curve starts along the line between the an endpoint and the first control point, and then curves to smoothly meet the line between the second control point and the other endpoint.



INDUCTIVE UNIVERSITY

Bezier Curves

Watch the Video

Next ...

- Editing Shape Paths

Editing Shape Paths

Editing the paths of your vector graphic shapes is done by using the path tool ( ). Simply select any shape or line while the path tool is active to start editing it. If the shape is already a path, you can switch to the path tool by double-clicking on the shape. You can convert any shape into a general path by selecting the **To Path** ( ) function under the **Shape** menu. Shapes will also implicitly be turned into paths if they are altered in a way not supported by the underlying shape. For example, if you stretch a rotated rectangle, thereby skewing it into a parallelogram, it will become a path automatically.

Each point on the path is represented by a diamond-shaped handle when the path editor is active. These handles can be dragged to move them around. They can also be selected by clicking on them or dragging a selection rectangle to select multiple points. This allows groups of points to be altered simultaneously. Bézier segments also have handles on their control points, represented by small circles with a line drawn back to an endpoint that can be dragged to change the curvature of the segment.

To change a segment between open, straight, and curved, simply use the toolbar functions that become visible when the path editor tool is active. Points can also be added and removed using the functions on the path editor toolbar. Filled shapes have two fill settings that control whether or not holes in the shape should be filled. To remove the fill entirely, simply set the fill paint property to **No Paint**.

When editing paths directly, it is often useful to be zoomed in on the path. Don't forget that you can zoom in on a location by holding down **ctrl** and using your mouse wheel to zoom in on a particular area without having to zoom in and then scroll. Also, if you press your mouse wheel in, you can pan around your window.

INDUCTIVE UNIVERSITY

Editing Shape Paths

Watch the Video

Next ...

- Eyedropper Tool

Eyedropper Tool

The **Eyedropper** tool is used to set the selected vector graphic shape(s) and/or component(s) foreground /background or stroke/fill colors by pulling the colors from somewhere else in the window.

To change the foreground/background color of a component

1. Select the object that you want to change.
2. Select the **Eyedropper** tool from the group drawing tools on the far right of the Designer.
3. Select a color from any object on the Root Container of the window and left-click to set the background color and the right-click to set the foreground color.
   The technique works the same on other components by affecting the component's **Fill Paint** colo r property and the **Stroke Paint** color property.
4. To exist the **Eyedropper**, select the Selection Tool (the mouse pointer icon) from the drawing tools menu.

INDUCTIVE UNIVERSITY

Eyedropper Tool

Watch the Video

Next ...

- add link here

Binding Properties

Property Binding is perhaps the most important concept to understand when designing a project using the Vision module. It is primarily through property binding 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.

When you initially place a component on a screen, it doesn't really do anything. Changing its properties in the Designer will make it look or act different, but it has no connection to the real world and this is what property binding adds.

Property binding, as its name suggests, lets you bind a property to something else, and that something else could be:

- An OPC Tag
- The results of a SQL query executed against a remote database
- Some other component's property
- An expression involving any of these things
- The results of a Python script
- and so on...

For example, bind the Value property of an LED Display to an OPC Tag, and voilà - the value property will always be the value of that tag - creating a dynamic display. Bindings can also work the other way, using a bidirectional binding. Bind the value of a numeric text box to a tag, and that tag will be written to when someone edits the value in the text box.

The power of property bindings comes from the variety of different binding types that exist, and the fact that you can bind nearly any property of a component to anything else. Want its foreground to turn red when an alarm is above a certain severity? Bind its LED Lit (`glyphForeground`) color to a tag's `AlertCurrentSeverity` property. Want it to only appear if a supervisor is on shift? Bind its visible property to the result of a SQL query that joins a personnel table with a shift table. The possibilities are, quite literally, endless.

Property Binding Types

A property can have many different types of binding, for example it can have a Tag or an Expression binding. The following is a list of the different binding types for a property:

- Tag
- Indirect Tag
- Tags Historian
- Property
- Expression
- Database Browse
- SQL Query
- Cell Update
- Function

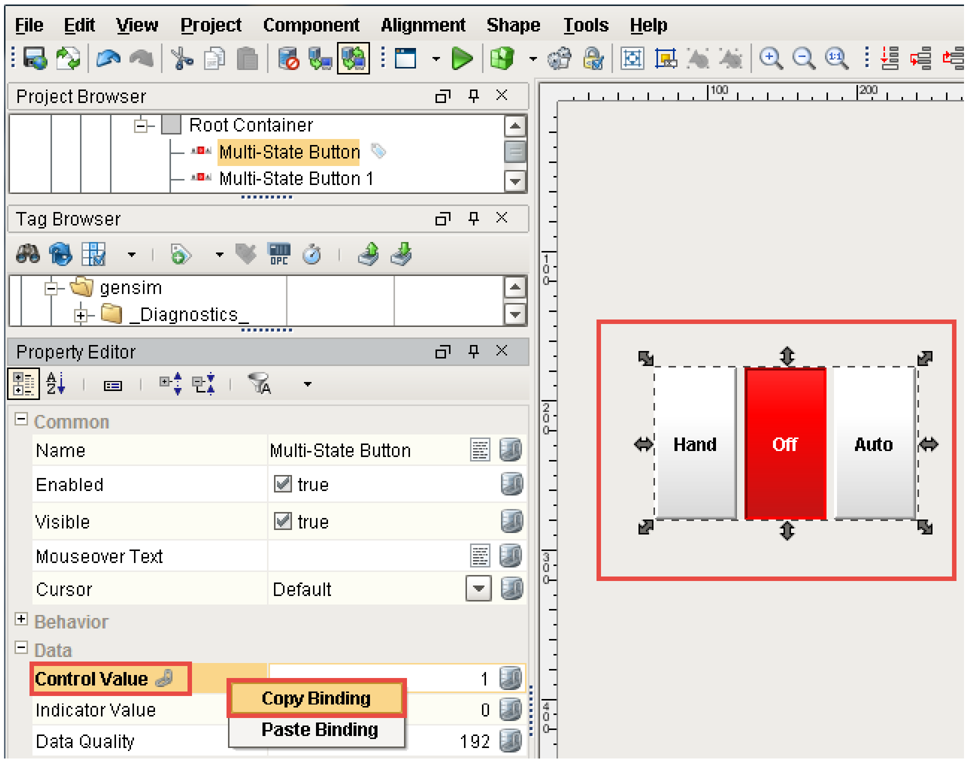
Copying Bindings

When you copy a component, all bindings, scripts, etc. are copied along with it, but you can also copy a property binding from one property to another. This can be on the same component or a completely different component. The only prerequisite is that both property bindings must use a compatible property type. I.E. A binding that resolves to a string will not work on an integer property, etc.

Here's how to copy a binding property:

- In the Designer, click on a **component**.
- Select the **bound property** and right click.

- Choose **Copy Binding**.



Name	Multi-State Button
Enabled	true
Visible	true
Mouseover Text	
Cursor	Default
Control Value	1
Indicator Value	0
Data Quality	192

In this section ...

Property Binding

A property binding is a very simple type of binding. It simply binds one component's property to another. When that property changes, the new value is pushed into the property that the binding is set up on.

Why aren't all properties listed?

You may notice that the list of properties available to bind to is smaller than the list of all properties. While nearly all properties can be bound, only some properties can be bound to. Only properties for which a **propertyChangeEvent** is fired may be bound to.

INDUCTIVE UNIVERSITY

Property Binding

Watch the Video

How Bindings Work: Event-based vs. Polling

While there are quite a few different binding types, they all boil down into two broad categories: event-based and polling. Some complex bindings can span both categories.

Event-based bindings are evaluated when the object they are bound to changes. For example, when you bind a property to a tag, that binding listens to the tag, and every time the tag changes, it assigns the tag's new value into the property that it is on. If you bind the value of a Cylindrical Tank to the value of a Slider, every time the slider changes, it fires a propertyChangeEvent. The binding is listening for this event, and when it is fired, updates the tank's value. The following bindings are event-based:

- Tag bindings
- Property bindings

Polling bindings are evaluated when a window first opens, on a timer, or when they change. For example, if you bind the data property of a Table to the results of a SQL query, that query will run on a timer, updating the Table every time it executes. The following bindings are based on polling:

- SQL query bindings
- Some expression functions, like runScript() or now()

Many bindings can combine elements of a polling binding and event-based binding. An expression binding may combine lots of other bindings to calculate a final result. A query binding will often itself be dynamic, altering the query based on other bindings.

For example, you might have a dropdown on a window that lets the operator choose a type of product that is produced. Then you can use a query binding like the following to calculate the defect rate for the given product:

```
SELECT
   SUM(defective) / COUNT(*) AS DefectRate
FROM
   production_table
WHERE
   productCode = '{Root Container.ProductPicker.selectedValue}'
```

The blue code is a property binding inside of the query binding. Every time this (event-based) binding fires, the query will run again. Using bindings like this, you can create highly dynamic and interactive screens with no scripting whatsoever.

Polling Options

The following are the options you can choose from for bindings that poll:

- **Polling Off**
  A polling-off binding will execute once when the window is opened, and then it will only execute again if it changes. The typical example of a binding that can change is a SQL query binding where it uses the brace-notation ({}) to include dynamic information inside the query. When this dynamic information changes the query, it will run again.

- **Relative Rate**
  The binding will execute at a regular rate, based on a delta off of the project's base polling rate, see Setting Project Polling Base Rate. This is usually a good idea so that you can speed up or slow down an entire client's polling system in one place.

- **Absolute Rate**
  Using this option, you can specify an absolute rate for the binding to execute at, instead of one that is based off the relative rate.

Next ...

- Property Binding - Bidirectional

Property Binding - Bidirectional

Tag bindings and Query bindings can be set as bidirectional bindings. This means that not only is the binding assigning the tag value or query value into the property, but it is also listening for changes to that property, which will then be written back to the tag or the database.

INDUCTIVE UNIVERSITY

Property Binding - Bidirectional

Watch the Video

Tag Bindings

Tag bindings can be made bidirectional simply by checking the **Bidirectional** checkbox at the bottom of the **Property Binding** window. A tag can be set as a bidirectional binding, if it has a read/write permission and if the user has the security permission to write to the tag.The Fallback Delay is the amount of time that the value will remain at the written value, waiting for a tag change to come in. If no tag change comes in within the allotted time (specified in seconds), the property will fall-back to the value as it was before the write. This is needed, because sometimes even if a write succeeds, another write or ladder logic in a PLC might have written something different, even the old value, in which case no tag change event will be generated. As a rule of thumb, the fallback delay should be twice the tag's scan class rate.

Query Bindings

When a query binding is made bidirectional, it needs an UPDATE query to execute when the property changes. You can use the special marker `{this}` as a placeholder for the new value. Bidirectional query bindings are only available on scalar-typed properties, that is, not Datasets.

Next ...

- Tag Binding

Tag Binding

A tag binding is a very straight-forward binding type. It simply binds a property directly to a tag. This sets up a tag subscription for that tag, and every time the chosen property changes, the binding is evaluated, pushing the new value into the bound property.

If the tag is a leased scan class, this binding will activate the lease while the window is open. If you choose a tag in the tree, and not a property, the Value property is assumed.

Tag Binding

Watch the Video

Next ...

- Tag Binding - Drag and Drop

Tag Binding - Drag and Drop

You can quickly bind a tag by dragging it from the Tag Browser into a component or into the component property. In the Property Editor, the properties that have the new bindings will display in bold font along with a binding icon.

INDUCTIVE UNIVERSITY

Tag Binding – Drag and Drop

Watch the Video

Binding to a Component

Ignition automatically creates the tag bindings to several of the component properties when you choose to bind a tag to a component by dragging and dropping . In addition, some of the bindings will be bidirectional or they may be expression bindings. How Ignition handles the binding depends on the disparity between the datatypes of the tag and the target component. For example, binding a numerical tag to a label component will result in an expression binding that formats the number to more conducive to a label component.

Binding to a Component Property

Ignition automatically creates a tag binding to the property that you dropped the tag on to resulting in the property binding to the value of the tag. In addition, it is possible to bind the tag attributes to the component's property. For example, the tag's engineering high limit attribute could be bound to the capacity property of a cylindrical tank component.

Next ...

- Tag Binding - Bidirectional

Tag Binding - Bidirectional

Choosing bidirectional will make the tag bindings also write to the chosen tag when the property changes. The fallback delay is the amount of time to keep the property at the written value waiting for a new tag value update to come in. If no update arrives within the given timeout, the property falls back to the original value.

See also: Property Binding - Bidirectional

INDUCTIVE UNIVERSITY

Tag Binding - Bidirectional

Watch the Video

Next ...

- Tag and Component Overlays

Tag and Component Overlays

It is especially important in HMIs to be able to gauge the health and accuracy, also called quality, of what is displayed at a glance. For these reasons, components show visual overlays for various reasons to indicate that the data they are displaying is of poor quality or is pending a reply from the device.

Choosing 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 Op-out option is located in the tag bindings for components. If this option is chosen the user would have no indication that the underlying tag quality is something other than of good quality therefore it is possible the user.

INDUCTIVE UNIVERSITY

Tag and Component Overlays

Watch the Video

Next ...

- Indirect Tag Binding

Indirect Tag Binding

An indirect tag binding is very much like a standard tag binding, except that you may introduce any number of indirection parameters to build a tag path dynamically in the runtime. These parameters are numbered starting at one, and denoted by braces, for example, {1}.

The binding will be bound to the tag represented by the tag path after the indirection parameters have been replaced by the literal values they are bound to. An indirection parameter may represent a property on any component in the same window, or the value of any tag.

Indirect tag bindings can use bidirectional mode just like standard tag bindings.

Making bindings indirect is an important part of the binding system. Indirect Tag, Expression, and SQL Query bindings can all be made indirect. All this means is that what the binding is bound to can be changed based on the value of something else.

For example, instead of binding straight to a tag's path, like

```
[TagProvider]MyPlant/EastArea/Valves/Valve4/FlowRate
```

you can use other properties to make that path indirect. Suppose the "area" and valve number that we were looking at was passed into our window via parameter passing. Then we might use those parameters in the tag path, like this:

```
[TagProvider]MyPlant/{1}Area/Valves/Valve{2}/FlowRate
{1}=Root Container.AreaName
{2}=Root Container.ValveNumber
```

Now our binding will change what tag it is pointing to based on the values of those root container properties.

Making query bindings indirect, or dynamic, is so common that there are probably more indirect query bindings than direct ones. All this means is that the query is calculated dynamically. A common example of this would be to use a dynamic start date and end date in a query. Suppose we had a Classic Chart that we're binding to a range of history, and a Date Range that we wanted to have the operator use to select a time period. Then we could use an indirect query binding like this:

```
SELECT
   t_stamp, flow_rate, amps
FROM
   valve_history
WHERE
   t_stamp >= '{Root Container.DateRange.startDate}' AND
   t_stamp <= '{Root Container.DateRange.endDate}' AND
   valve = {Root Container.ValveNumber} AND
   area = '{Root Container.AreaName}Area'
```

Indirect Tag Binding

Watch the Video

Creating Indirect Tag Bindings

Indirect tag bindings are an important part of Ignition and the binding system. This means that what the binding is bound to, can be changed based on the value of something else. For example, you can have a selection from a drop-down list to choose which tags to look at on the screen.

We have four motor UDT tags and a motor template. Let's create a single window with one drop-down list and one motor template. The drop-down list gives the option of which motor to look at. The motor tags look like this:

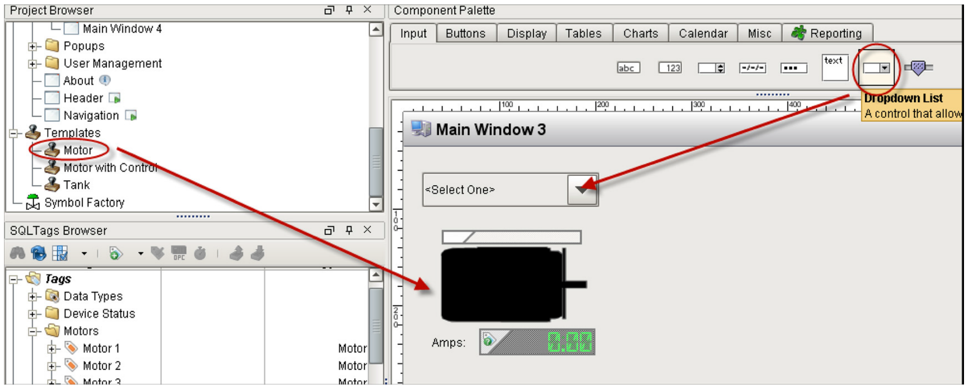
```
Motors/Motor 1
Motors/Motor 2
Motors/Motor 3
Motors/Motor 4
```

The only difference between each motor is the number.

To create a window with a drop-down list and a motor template

1. From Project Browser, under **Templates**, drag in a template, in this example it is a **Motor** template.

2. From the **Input** tab of **Component Palette**, drag a **Dropdown List** component.



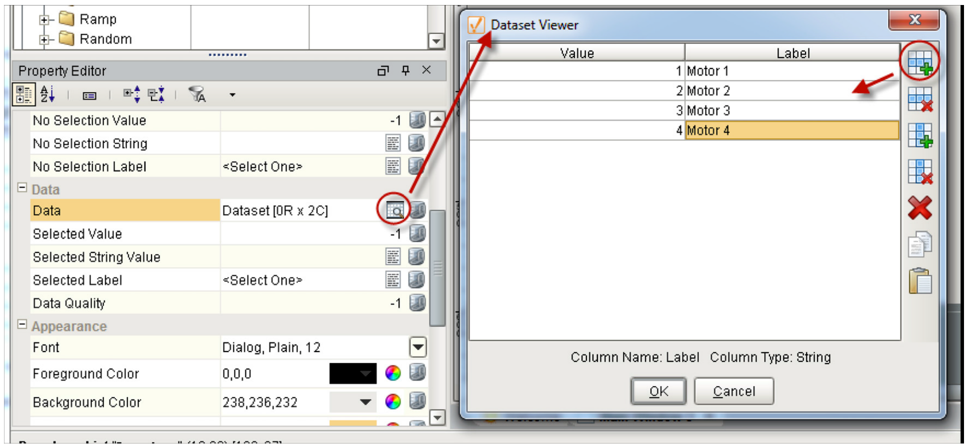
Now we need to setup the options for the drop-down list in the **Property Editor**. The drop-down component has the **Data** property which is a dataset. There are three ways to set up the dataset and the corresponding behavior is as follows:

One Column [String]	Apples Oranges Bananas	• Drop-down displays values from the first column. • Selected Value is undefined. • Selected String Value represents value from first column. • Selected Label represents value from first column.
Two Column [Integer, String] (Default)	201 Apples 202 Oranges 203 Bananas	• Drop-down displays values from second column. • Selected Value represents value from first column. • Selected String Value represents value from second column. • Selected Label represents value from second column.
Two Column [String, String]	APL Apples ORN Oranges BAN Bananas	• Drop-down displays values from second column. • Selected Value is undefined. • Selected String Value represents value from first column. • Selected Label represents value from second column.

Let's use the default mode which is the second option above, **Two Columns [integer, string]**, to setup the options in the drop-down list.

3. Select the drop-down component and click the **Dataset view** icon next to the **Data** property.
4. On the right side of **Dataset Viewer** window, click the **+ green row** icon four times to add 4 rows.
5. Set the rows to the following values:

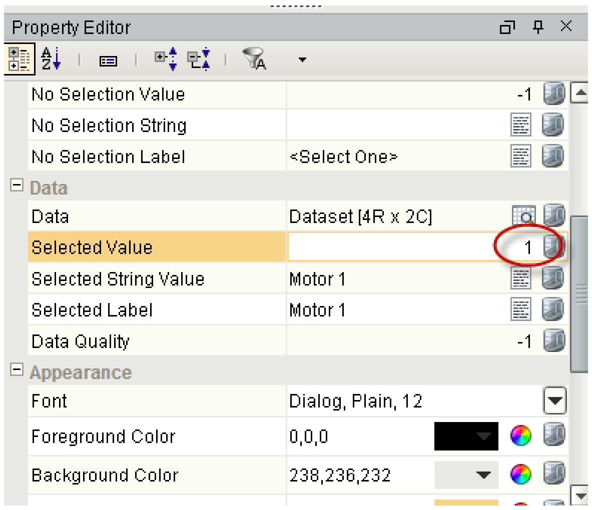
```
1     Motor 1
2     Motor 2
3     Motor 3
4     Motor 4
```



Property	Value
No Selection Value	-1
No Selection String	
No Selection Label	<Select One>
Data	
Data	Dataset [0R x 2C]
Selected Value	-1
Selected String Value	
Selected Label	<Select One>
Data Quality	-1
Appearance	
Font	Dialog, Plain, 12
Foreground Color	0,0,0
Background Color	238,236,232

Value	Label
1	Motor 1
2	Motor 2
3	Motor 3
4	Motor 4

6. Click **OK** to save.
7. Set the **Selected Value** property to **1**.
   Notice the values for **Selected String Value** and **Selected Label** fill in for you.

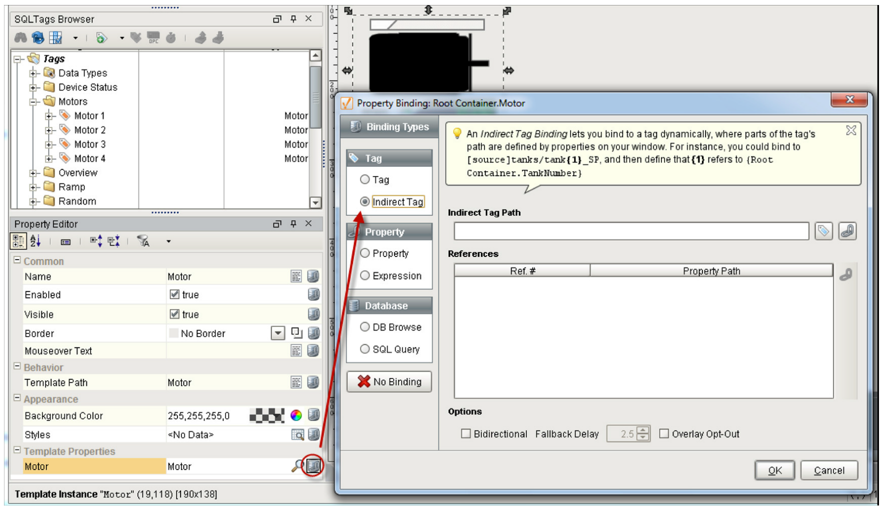


Property Editor

No Selection Value	-1
No Selection String	
No Selection Label	<Select One>
Data	
Data	Dataset [4R x 2C]
Selected Value	1
Selected String Value	Motor 1
Selected Label	Motor 1
Data Quality	-1
Appearance	
Font	Dialog, Plain, 12
Foreground Color	0,0,0
Background Color	238,236,232

We want to use the Selected Value in an indirect binding on the template to dynamically select which motor to look at.

8. Select the Motor template on the window and bind the Motor UDT property to an Indirect Tag.



SQLTags Browser

- Tags
  - Data Types
  - Device Status
  - Motors
    - Motor 1 — Motor
    - Motor 2 — Motor
    - Motor 3 — Motor
    - Motor 4 — Motor
  - Overview
  - Ramp
  - Random

Property Editor

Common	
Name	Motor
Enabled	true
Visible	true
Border	No Border
Mouseover Text	
Behavior	
Template Path	Motor
Appearance	
Background Color	255,255,255,0
Styles	<No Data>
Template Properties	
Motor	Motor

Template Instance "Motor" (19,118) [190x138]

Property Binding: Root Container.Motor

Binding Types: Tag ( ) Tag [✓ Indirect Tag]; Property ( ) Property ( ) Expression; Database ( ) DB Browse ( ) SQL Query; No Binding

An *Indirect Tag Binding* lets you bind to a tag dynamically, where parts of the tag's path are defined by properties on your window. For instance, you could bind to `[source]tanks/tank{1}_SP`, and then define that **{1}** refers to `{Root Container.TankNumber}`

Indirect Tag Path

References

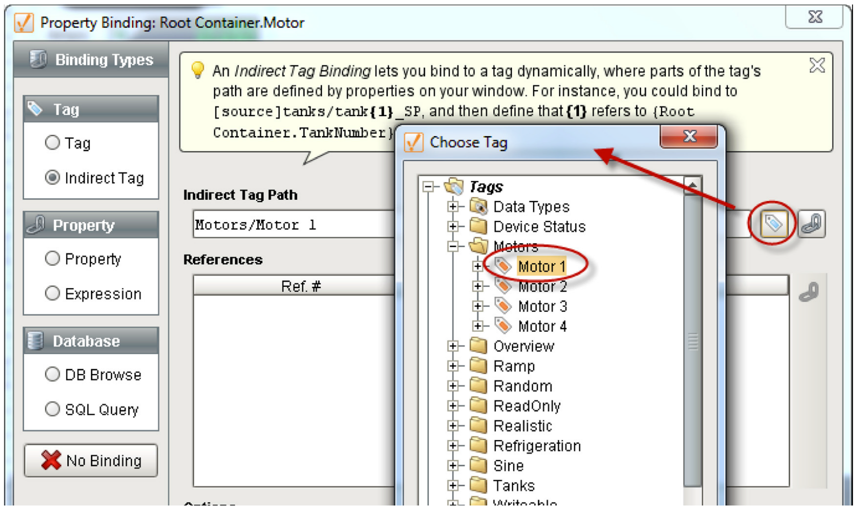
Ref. #	Property Path

Options: Bidirectional Fallback Delay 2.5 Overlay Opt-Out

OK Cancel

The easiest way to use the indirect tag binding, is to select a tag as a starting point and then make it dynamic.

9. Click the Tag button on the right of Indirect Tag Path and select the Motor 1 tag as the reference.



Property Binding: Root Container.Motor

Binding Types: Tag ( ) Tag [✓ Indirect Tag]; Property ( ) Property ( ) Expression; Database ( ) DB Browse ( ) SQL Query; No Binding

An *Indirect Tag Binding* lets you bind to a tag dynamically, where parts of the tag's path are defined by properties on your window. For instance, you could bind to `[source]tanks/tank{1}_SP`, and then define that **{1}** refers to `{Root Container.TankNumber}`

Indirect Tag Path: `Motors/Motor 1`

References

Ref. #

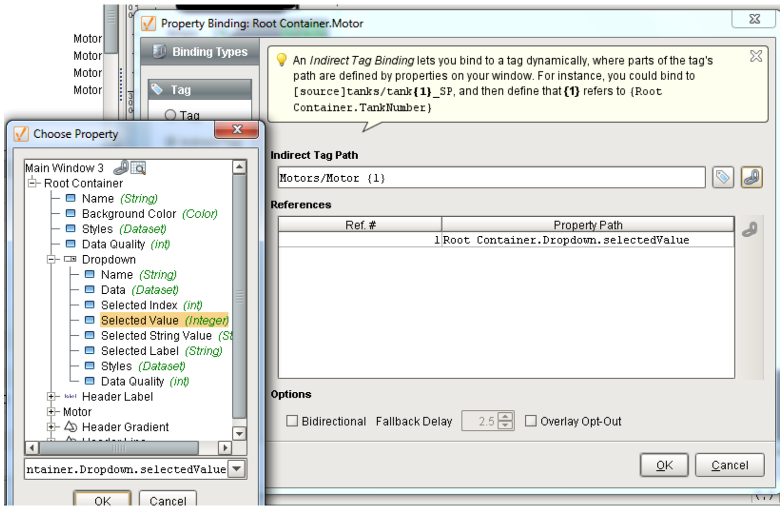
Choose Tag

- Tags
  - Data Types
  - Device Status
  - Motors
    - Motor 1
    - Motor 2
    - Motor 3
    - Motor 4
  - Overview
  - Ramp
  - Random
  - ReadOnly
  - Realistic
  - Refrigeration
  - Sine
  - Tanks

The drop-down list contains which motor number to look at. So we need to link the Motor tag number to the drop-down’s selected value.

10. Highlight the number 1 (in Motors/Motor 1) in the Indirect Tag Path, then click on the link button to the right to bring in the drop-down's Selected Value property.

    Ignition replaces the 1 with a {1} and links it to the drop-down. That means Ignition substitutes {1} with the value of the drop-down allowing it to select which motor tag to look at.



Property Binding: Root Container.Motor

Binding Types

Tag

Tag

An *Indirect Tag Binding* lets you bind to a tag dynamically, where parts of the tag's path are defined by properties on your window. For instance, you could bind to `[source]tanks/tank{1}_SP`, and then define that **{1}** refers to `{Root Container.TankNumber}`

Indirect Tag Path

`Motors/Motor {1}`

References

Ref. #	Property Path
1	Root Container.Dropdown.selectedValue

Options

Bidirectional   Fallback Delay 2.5   Overlay Opt-Out

OK   Cancel

Choose Property

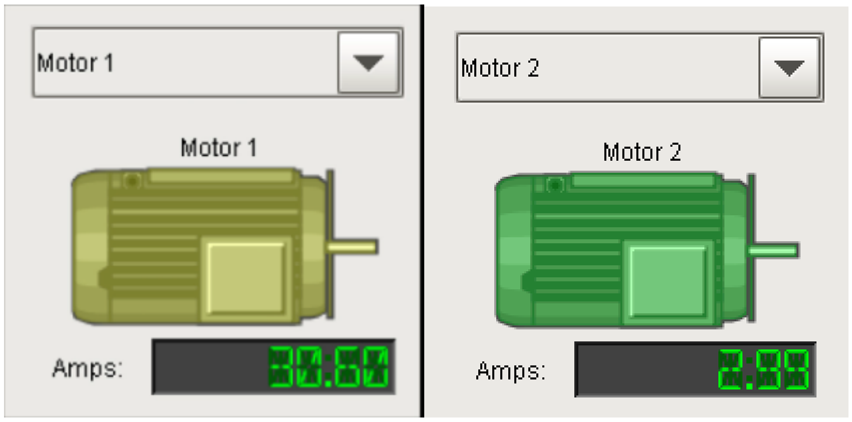
```
Main Window 3
Root Container
    Name (String)
    Background Color (Color)
    Styles (Dataset)
    Data Quality (int)
    Dropdown
        Name (String)
        Data (Dataset)
        Selected Index (int)
        Selected Value (Integer)
        Selected String Value (St
        Selected Label (String)
        Styles (Dataset)
        Data Quality (int)
    Header Label
    Motor
    Header Gradient
```

ntainer.Dropdown.selectedValue

OK   Cancel

11. Click **OK**.
    Now you can try it out.

12. Save your project and open the runtime.
    You can change the drop-down list and it will select a different motor.



It is as easy as that to setup tag indirection. You can apply this method to popups or anywhere in Ignition.

See also: Parameterized Popup Windows

Next ...

- Tag History Binding

Tag History Binding

The Tag Historian binding type, which is only available for Dataset type properties, runs a query against the Tag Historian.

INDUCTIVE UNIVERSITY

Tag Historian Binding

Watch the Video

Selected Historical Tags

For this type of query, you must select at least one tag path to query. The Dataset returned by the query will have a timestamp column, and then a column for each path that you select here.

These paths may use indirection following the same rules as the Indirect Tag Binding. Simply type the indirection parameters (for example, {1}) into a selected tag path by double-clicking in the list of selected paths. All valid parameters will appear in the lower indirection table.

Date Range

Choose either a Historical or Realtime query. Historical queries use a date range that must be bound in from other components on the screen, typically a Date Range or a pair of Popup Calendars. Realtime queries always pull up a range that ends with the current time, so all they need is a length.

Sample Size and Aggregation Mode

The sample size determines how the query results will look.

- **Natural** - A Natural query will look up the logging rate for the queried tags (when possible), and return results spaced apart at that rate. This means that the return size will vary with the date range.
- **On Change/Raw** - An On Change query will return points as they were logged, and can be thought of as a "raw" query mode. This means that the results may not be evenly spaced. Also, it is important to note that every changed value will result in a row, and therefore if you are querying multiple tags and once, you may end up with more rows than you anticipated. For example, if tag A and tag B both change, you would end up with [[A_0, B_0],[A_1, B_0], [A_1, B_1]].
  Note: If you want to essentially retrieve raw values, while coalescing them down into fewer rows, try using the Interval sample mode, with an interval set to your largest acceptable time between rows, and select "prevent interpolation" from the advanced settings.
- **Fixed** - A Fixed query will return the given number of rows. Where data was sparse, interpolated values will be added. Where data is dense, the Aggregation Mode will come into play.
- **Interval** - Similar to fixed, but with the spacing based on time, rather than the number of requested results.

The aggregation mode dictates what happens when multiple raw values are encountered for a given sample window (the size of which is determined by the number of requested rows, or the interval size).

- **Min/Max** - The minimum and maximum values will be returned for the window. In other words, two rows will be returned. If only one value is seen in the interval, only one row will be returned.
- **Time-weighted Average** - The values are averaged together, weighted for the amount of time they cover in the interval.
- **Closest Value** - The value closest to the ending time of the interval will be returned.
- **Simple Average** - The values are summed together and divided by the number of values.
- **Sum** - The values in the interval are summed together.
- **Maximum** - The maximum value in the interval.
- **Minimum** - The minimum value in the interval.
- **Duration On** - Returns the number of seconds that the value was recorded as non-zero.
- **Duration Off** - Returns the number of seconds that the value recorded as zero.
- **Count On** - Returns the number of times the tag's value went from a zero value to non-zero: i.e.,
- **Count Off** - Returns the number of times the tag's value changed from a non-zero value to zero.

  The following feature is new in Ignition version **7.8.1**
  Click here to check out the other new features

- **Count** - Returns the number of times a value was recorded

- The following feature is new in Ignition version **7.8.3**
  Click here to check out the other new features

- **Percent Good** - Time-weighted percentage of good values over the date range.
- **Percent Bad** - Time-weighted percentage of bad values over the date range.
- **Range** - Returns the range between the highest and lowest value for the period.
- **Standard Deviation** - Returns the standard deviation of values, or how much spread is present in the data; low standard deviation shows the values are close to the mean, and high standard deviation shows that the data points are spread out over a large range of values. Only good quality values are used when calculating.
- **Variance** - Returns the variance of values. Similar in concept Standard Deviation. Only good quality values are used when calculating.

Return Format

Return format dictates how the requested data will be returned. The options are "wide" (default), in which each tag has its own column, and "tall", in which the tags are returned vertically in a "path, value, quality, timestamp" schema.

Advanced Options

These option affect the query results in more subtle ways.

- Ignore Bad Quality - Only data with "good" quality will be loaded from the data source.
- Prevent Interpolation - Requests that values not be interpolated, if the row would normally require it. Also instructs the system to not right result rows that would only contain interpolated values. In other words, if the raw data does not provide any new values for a certain window, that window will not be included in the result dataset.
- Avoid Scan Class Validation - "Scan class validation" is the mechanism by which the system determines when the gateway was not running, and returns bad quality data for these periods of time. By enabling this option, the scan class records will not be consulted, which can improve performance, and will not right bad quality rows as a result of this check.

Tags Historian information is often easiest to work with in the Easy Chart component, which handles all of these options automatically.

Next ...

- Expression Binding
- **Tag Historian**
- **Data Types in Component Properties**
- **system.tag.queryTagHistory**

Expression Binding

An expression binding is one of the most powerful kinds of property bindings.

It uses a simple expression language to calculate a value. This expression can involve lots of dynamic data, such as other properties, tag values, results of Python scripts, queries, and so on.

You can use expressions for many different purposes. Anytime information needs to be massaged, manipulated, extracted, combined, split, and so on - think expressions!

INDUCTIVE UNIVERSITY

Expression Binding

Watch the Video

Example 1

You have 3 bits in a PLC, only one of which will be on at a time. You want to turn these three 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})
```

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'
```

Example 3

You have a button that starts a batch, but you only want to let it be pressed after the operator has entered a scale weight. Bind the button's enabled property to:

```
 {Root Container.EntryArea.WeightBox.doubleValue} > 0.0
```

Example 4

You want to display a process's current state, translating a code from the PLC to a human-readable string, use of these two expressions (they're equivalent)

```
if ({CurrentProcessState} = 0, "Not Running",
if ({CurrentProcessState} = 1, "Warmup phase - please wait",
if ({CurrentProcessState} = 2, "Running", "UNKNOWN STATE")))

- or -

switch ({CurrentProcessState},
   0,1,2,
   "Not Running",
   "Warmup phase - please wait",
   "Running",
   "UNKNOWN STATE")
```

See also: Expression Overview and Syntax

Next ...

- Expression Binding - Concat Strings

Expression Binding - Concat Strings

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.

The following is an example:

```
concat("The answer is: ", "42") //returns "The answer is: 42"
```

For more information see the Appendix.

Expression Binding – Concat Strings

Watch the Video

Next ...

- Expression Binding - Celsius to Fahrenheit

Expression Binding - Celsius to Fahrenheit

Use an expression binding to convert a temperature from Celsius to Fahrenheit. This is an example of how 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

Next ...

- Expression Binding - Format Date

Expression Binding - Format Date

You can format a date in an expression binding by using the `dateFormat` and `now` functions.

INDUCTIVE UNIVERSIT

Expression Binding – Format Date

Watch the Video

To make a label that updates to show the current time

Let's create a label that updates every second to reflect 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. The second argument refers to the date format that you want returned. The `now` functio n returns the current time and in this case it will update every second.
   For more information see the Appendix.

Next ...

- Expression Binding - Date Manipulations

Expression Binding - Date Manipulations

You can manipulate dates in expression bindings such as a date addition/subtraction with the `dateArithmetic` 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")
```

See the dateArithmetic expression function, in the Appendix.

Expression Binding – Date Manipulations

Watch the Video

Next ...

- Expression Binding - Bit Functions

Expression Binding - Bit Functions

You can use various bit functions in expression bindings like `getBit` to return individual bits of a word.

Expression Binding - Bit Functions

Watch the Video

Example

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.

Next ...

- Expression Binding - Switch
- getBit

Expression Binding - 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))
```

You can see more examples in the Appendix.

Expression Binding – Switch

Watch the Video

Next ...

- Expression Binding - Checking Conditions

Expression Binding - Checking Conditions

You can use expression bindings to return true or false based on different conditions.

INDUCTIVE UNIVERSIT

Expression Binding - Checking Conditions

Watch the Video

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
```

Next ...

- DB Browse Binding
- Expression Overview and Syntax

DB Browse Binding

The DB Browse binding is technically equivalent to the SQL Query binding, except that it helps write the queries for you. Using the database browser, you can pick the table that you want to pull content from. If you have a fixed range of data to choose, simply select it in the table, and watch the query get generated.

In the browse tree, you can choose which columns should act as your keys (these columns get put in the WHERE clause based on your selection) and which columns should be used to sort the data (these columns are put in the ORDER BY clause).

This binding type also serves as a convenient jumping-off point for the more flexible SQL Query binding. Construct the basic outline of your query in the DB Browse section, and then flip over to the SQL Query binding. Your query will be retained and can then be improved by hand.

INDUCTIVE UNIVERSITY

DB Browse Binding

Watch the Video

Next ...

- DB Browse Binding - Dynamic Filters

DB Browse Binding - Dynamic Filters

You can use the DB Browse binding to filter results from values on the screen.

DB Browse Binding - Dynamic Filters

Watch the Video

Next ...

- SQL Query Binding

SQL Query Binding

The SQL Query binding is a polling binding type that will run a SQL Query against any of the database connections configured in the Gateway.

SQL Query Binding

Watch the Video

Dynamic Queries

Using the brace notation, you can include the values of component properties (within the same window) and tag values inside your query. This is a very common technique to make your query dynamic. The values of the property or tag represented are simply substituted into the query where the braces are.

Note that because the substitution is direct, you'll often need to quote literal strings and dates to make your query valid. If you're getting errors running your query complaining about syntax, it is important to realize that these errors are coming from the database, not from Ignition. Try copying and pasting your query into the Query Browser and replacing the braces with literal values.

Example

A common requirement is to have a query filter its results for a date range. You can use the Date Range component or a pair of Popup Calendar components to let the user choose a range of dates. Then you can use these dates in your query like this:

```
SELECT
   t_stamp, flow_rate, amps
FROM
   valve_history
WHERE
   t_stamp >= '{Root Container.DateRange.startDate}' AND
   t_stamp <= '{Root Container.DateRange.endDate}'
```

Notice the single quotes around the braces. This is because when the query is run, the dates will be replaced with their literal evaluations. For example, the actual query sent to the database might look like this:

```
SELECT
   t_stamp, flow_rate, amps
FROM
   valve_history
WHERE
   t_stamp >= '2010-03-20 08:00:00' AND
   t_stamp <= '2010-03-20 13:00:00'
```

Fallback Value

If the property that is being bound is a scalar datatype (that is, not a Dataset), the value in the first column in the first row of the query results is used. If no rows were returned, the binding will cause an error unless the Use Fallback Value option is selected. The value entered in the fallback value text box will be used when the query returns no rows.

When binding a Dataset to a SQL Query, no fallback value is needed, because a Dataset will happily contain zero rows.

See also: Setting Project Polling Base, Connecting to Databases

Next ...

- SQL Query Binding - Polling

SQL Query Binding - Polling

There are several polling modes (polling off, relative, and absolute) that work with the SQL Query binding.

SQL Query Binding - Polling

Watch the Video

Next ...

- SQL Query Binding - Dynamic Filters

SQL Query Binding - Dynamic Filters

Make your SQL queries dynamic by including property values and tag values in the query.

SQL Query Binding - Dynamic Filters

Watch the Video

Next ...

- SQL Query Binding
- SQL Query Binding - Scalar Query and Fallback

SQL Query Binding - Scalar Query and Fallback

You can bind a scalar property to a SQL query and return the fallback value when no rows are returned.

SQL Query Binding - Scalar Query and Fallback

Watch the Video

Next ...

- SQL Query Binding - Scalar Query and Update

SQL Query Binding - Scalar Query and Update

You can bind an input component bidirectionally to a SQL query to allow data be retrieved and updated in the database.

SQL Query Binding - Scalar Query and Update

Watch the Video

Next ...

- Cell Update Binding

Cell Update Binding

The Cell Update binding enables you to easily make one or more cells inside a dataset dynamic. This is particularly useful for components such as the Linear Scale or the Easy Chart, that store configuration information inside datasets.

For example, when you configure indicators on a **Linear Scale** component using that component's customizer, the indicators that you set up are stored in the **Indicators** property on the scale. Suppose you wanted high-setpoint and low-setpoint indicators on the scale that weren't simply static values, but actually bound to a tag indicating the realtime high and low setpoints. To do this, you'd set up a **Cell Update** binding on the Linear Scale's Indicators property. You would configure two cell bindings - one for the low setpoint indicator's Value column, and one for the high setpoint. You would then bind these to the appropriate tags.

As another example, let's say you had an **Easy Chart** on a window that displayed 5 pens representing the history of a Compressor: running status, amperage, rpm, output pressure, and so on. Using Tag Historian, you had simply dragged the 5 applicable tags onto the Easy Chart. But now you want to use that same Easy Chart to dynamically display the same 5 pens of any of the many compressors in your system. To do this, you could pass the compressor number into the window as a parameter, and use it to calculate the tag path of the folder containing the pens. Then set up a Cell Update binding on the Easy Chart's **Tag Pens** property, dynamically altering the pens' tag paths. Now you have a generic chart window that you can use for any compressor.

Note that this binding type is only applicable for Dataset-typed properties.

INDUCTIVE UNIVERSITY

Cell Update Binding

Watch the Video

Next ...

- Function Binding

Function Binding

The function binding is a generic binding type that lets you bind a dataset property to the results of a function. It allows any of the function's parameters to be calculated dynamically via tag and property bindings. The function that you choose determines the parameters that are available. The following are the three different functions you can use for function binding:

- Alarm History
- Alarm Status
- Audit Log

Function Binding

Watch the Video

Next ...

- Component Styles

Component Styles

The **Style Customizer** feature in Ignition allows you to define a set of visual styles for a component that will change based upon a single driving property. To use the Style Customizer, you need a numeric or Boolean property on your component to use as the driving property. Custom properties on your on your component can be used as a driving property.

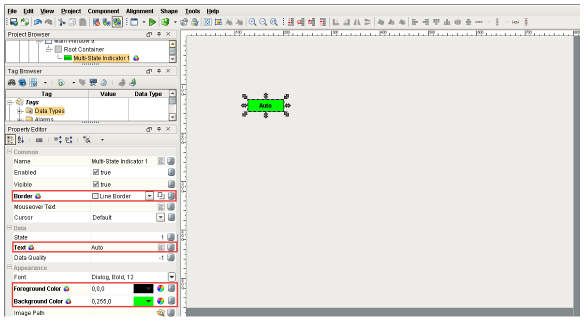
With the Style Customizer, you can change multiple property values on a component in a single location. Without styles, you have to go to every property and bind them all individually.

Some components have styles already setup and others do not. Let’s do an example that already has styles setup on the component.

Component Styles

Watch the Video

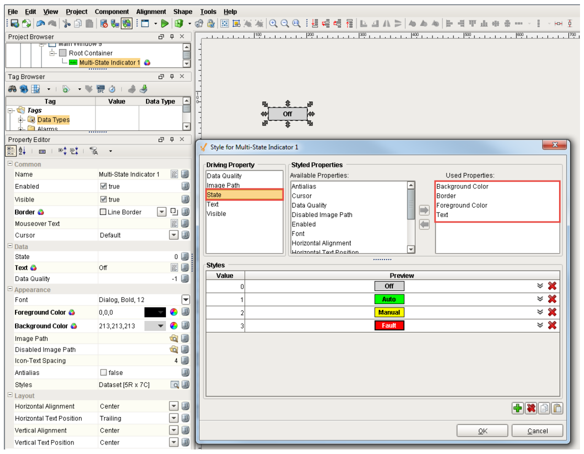
1. Drag in a **Multi-State Indicator** from your component palette on to your window. Right click on the component and notice in the Property Editor that several properties appear in bold with a paint canvas icon next to them letting you know that they are used in styles.



Property Editor	
Common	
Name	Multi-State Indicator 1
Enabled	☑ true
Visible	☑ true
Border	☐ Line Border
Mouseover Text	
Cursor	Default
Data	
State	1
Text	Auto
Data Quality	-1
Appearance	
Font	Dialog, Bold, 12
Foreground Color	0,0,0
Background Color	0,255,0
Image Path	

2. Right click on the **Multi-State Indicator component** and scroll down to **Customizers > Style Customizer**.

   Since styles is already setup on this component, you’ll notice that it already has a driving property selected. The driving property is **State** and it uses multiple visual properties like **Background Color**, **Border**, **Foreground Color** and **Text** which are already configured.



Property Editor	
Common	
Name	Multi-State Indicator 1
Enabled	☑ true
Visible	☑ true
Border	☐ Line Border
Mouseover Text	
Cursor	Default
Data	
State	0
Text	Off
Data Quality	-1
Appearance	
Font	Dialog, Bold, 12
Foreground Color	0,0,0
Background Color	213,213,213
Image Path	
Disabled Image Path	
Icon-Text Spacing	4
Antialias	☐ false
Styles	Dataset [5R x 7C]
Layout	
Horizontal Alignment	Center
Horizontal Text Position	Trailing
Vertical Alignment	Center
Vertical Text Position	Center

Style for Multi-State Indicator 1

Driving Property: Data Quality, Image Path, **State** (selected), Text, Visible

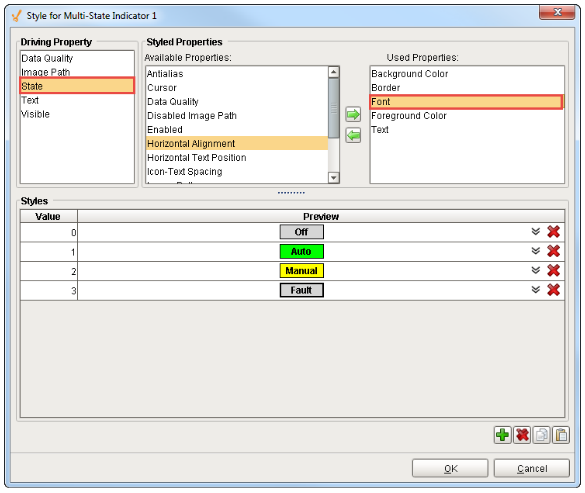
Styled Properties – Available Properties: Antialias, Cursor, Data Quality, Disabled Image Path, Enabled, Font, Horizontal Alignment, Horizontal Text Position

Used Properties: Background Color, Border, Foreground Color, Text

Styles

Value	Preview
0	Off
1	Auto
2	Manual
3	Fault

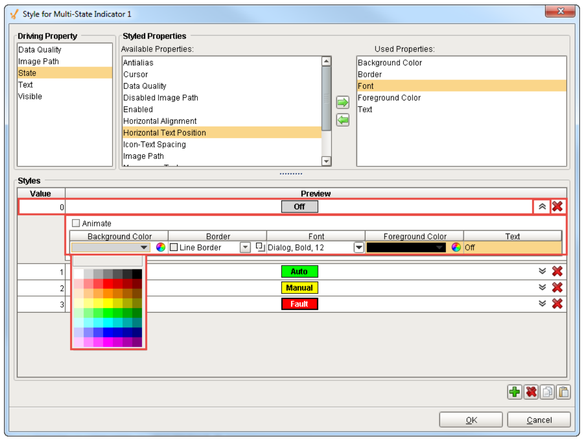
3. Styles are totally customizable. If you don’t like the pre-defined style for the preset values for the Multi-State Indicator, you can change them. You can add or remove styled properties. Adding and removing a styled property affects all the **Values** under the component. In this example, the **Font** property was added.



4. For each **Value** property, you can change its styled property to fit your requirements.

   Click the **Expand icon** and see all the styles for the **Value '0'**. Starting with Background Color, click on the **dropdown list** to change the color. You can configure all the styles you want to change: Border, Font, Foreground Color and Text.

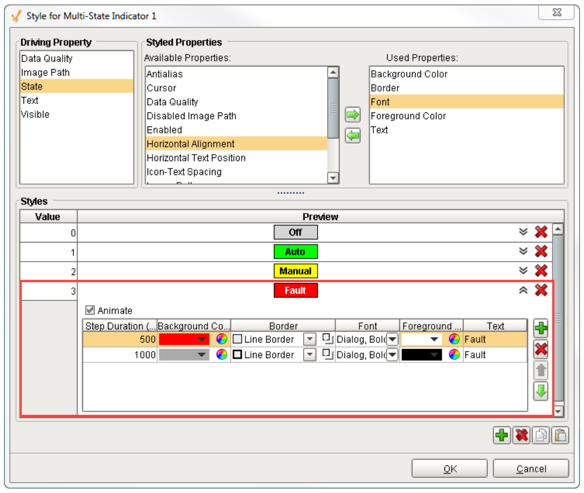
   You can change the styles for any Value in this Multi-State Indicator component. You can **Preview** each change after you make it.



5. With the styles feature you can **'Animate'** between different styles on the same **State**. To **Anim ate**, you can add one or more rows and set the style for what you want those new rows to look like.

   In this example, expand **Value '3'** and you will see that the **Animate** box is checked. Two rows were added and each animation step has a different background and foreground color set along with a **Step Duration** set in milliseconds to control the blinking between the 2 steps. You can add many steps and each step can have a different duration. See how it works in **Preview**.

   Press **OK**.



Style for Multi-State Indicator 1

Driving Property

- Data Quality
- Image Path
- State
- Text
- Visible

Styled Properties

Available Properties: Antialias, Cursor, Data Quality, Disabled Image Path, Enabled, Horizontal Alignment, Horizontal Text Position, Icon-Text Spacing

Used Properties: Background Color, Border, Font, Foreground Color, Text

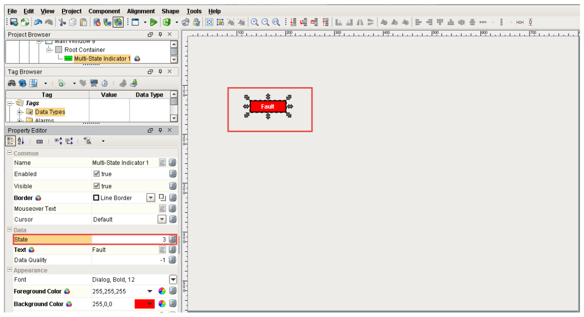
Styles

Value	Preview
0	Off
1	Auto
2	Manual
3	Fault

☑ Animate

Step Duration (...	Background Co...	Border	Font	Foreground ...	Text
500	🟥	Line Border	Dialog, Bold	⬜	Fault
1000	▨	Line Border	Dialog, Bold	⬛	Fault

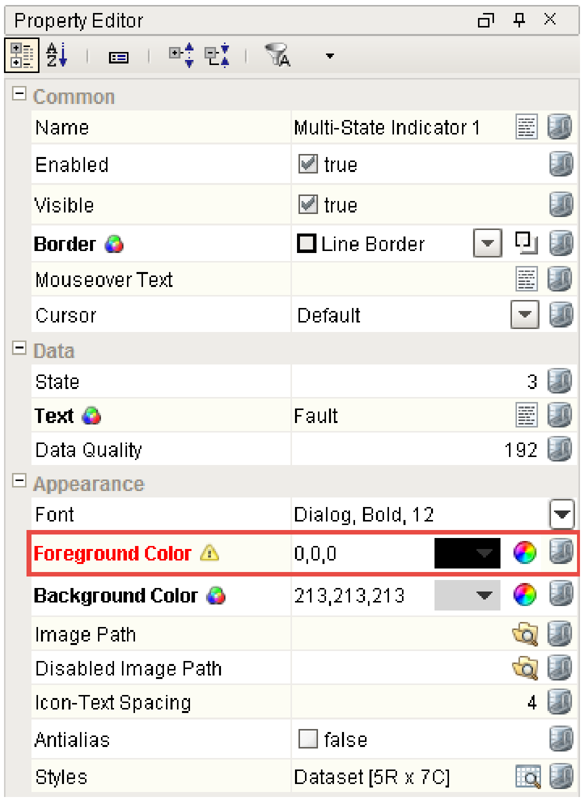
6. Change the **State** to **'3'** to see the component blink **'Faulted'** and switch between the background and foreground colors you setup.



Common	
Name	Multi-State Indicator 1
Enabled	☑ true
Visible	☑ true
Border	Line Border
Mouseover Text	
Cursor	Default
Data	
State	3
Text	Fault
Data Quality	-1
Appearance	
Font	Dialog, Bold, 12
Foreground Color	255,255,255
Background Color	255,0,0

Once your styles are setup, you can see styles coming through on the Multi-State Indicator component as it moves through the different States. In the Property Editor, you will notice that the Border, Foreground Color and Background Color properties are bold letting you know that they are used in styles

You cannot bind a property that is already being used by styles, otherwise, you get an error. The property name turns red in the In the Property Editor.


Property Editor

Common	
Name	Multi-State Indicator 1
Enabled	☑ true
Visible	☑ true
Border	☐ Line Border
Mouseover Text	
Cursor	Default
Data	
State	3
Text	Fault
Data Quality	192
Appearance	
Font	Dialog, Bold, 12
Foreground Color	0,0,0
Background Color	213,213,213
Image Path	
Disabled Image Path	
Icon-Text Spacing	4
Antialias	☐ false
Styles	Dataset [5R x 7C]

So as you can see, it's easy to change all the properties from one location using the Styles Customizer.

Next ...

- add link here

Some Useful HMI Examples

The HMI examples in this section show you how to tint vector graphics components, to make tank cutaways, and to animate components.

The following HMI examples use the symbols from the Symbol Factory module:

- Animating the Tint on a Grayscale Symbol
- Using Cutouts on Tanks

Using the Symbol Factory

If you have the Symbol Factory module installed, you've got nearly 4,000 industrial symbols at your fingertips to use in your projects.

To open the Symbol Factory Browser

1. Launch the Designer and open your project.
2. Choose **Symbol Factory** under the **Tools** menu or the project navigation tree.
   If you can't find these, the Symbol Factory module probably isn't installed.
   The Symbol Factory browser opens as a pop-up window that stays on top of the Designer.
3. Browse the different categories to explore what symbols are available, or search to find a specific symbol.
4. Find a symbol that you'd like to use and drag it onto an open Vision window.
   The symbol will become a group of shapes. Just like any group, you can double-click it to get to the shapes inside, or simply un-group it. This way you can edit the symbol if you need to.

   Note: If the Symbol Factory module is in trial mode, you'll only be able to use the first symbol from each category.

In this section ...

Color SVG Dynamically

Introduction

In Ignition, one useful HMI technique is to color an individual component of the SVG (scalable vector graphic). Whether you are bringing in a vector graphic from the Symbol Factory or importing an SVG from your computer, you can easily color in individual components of your SVG.

As with almost everything in Ignition, there are multiple ways of accomplishing the same result. For each component of the SVG, there are different binding types you can use to dynamically change the color based on a value.

INDUCTIVE UNIVERSITY

Color SVG Dynamically

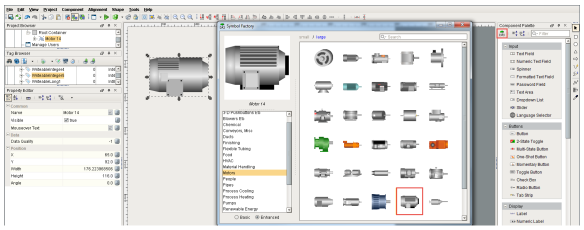
Watch the Video

Reference Information

There are a few pieces of assumed knowledge in this article that are common to all the coloring options. Click to learn more about each.

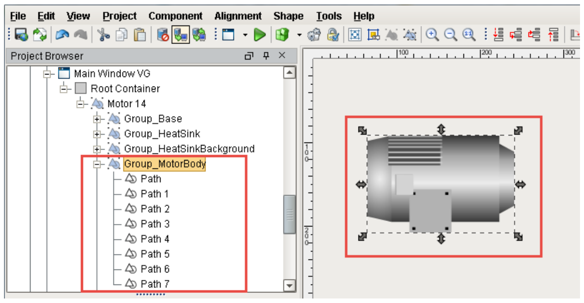
There can be many components inside an SVG. These components are often referred to by a number of different names: sub-components, members, parts, pieces, etc. All these names mean the same thing. For clarity, we will refer to an individual item inside an SVG as a **'component of the SVG.'**
The Symbol Factory has a library of symbols to choose from. Choose your symbol and drag it to your workspace.

In Designer, open the Symbol Factory in the Component Palette and drag a motor on to your workspace.

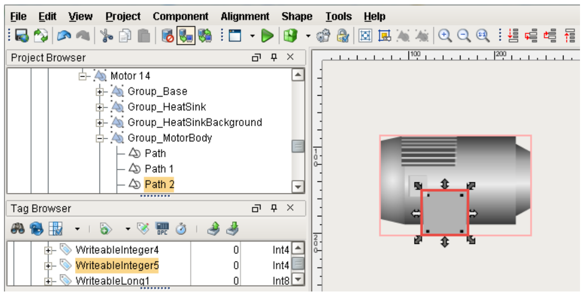


When you drag a vector graphic into a window, it's not just one flat image, it's actually comprised of many individual components.

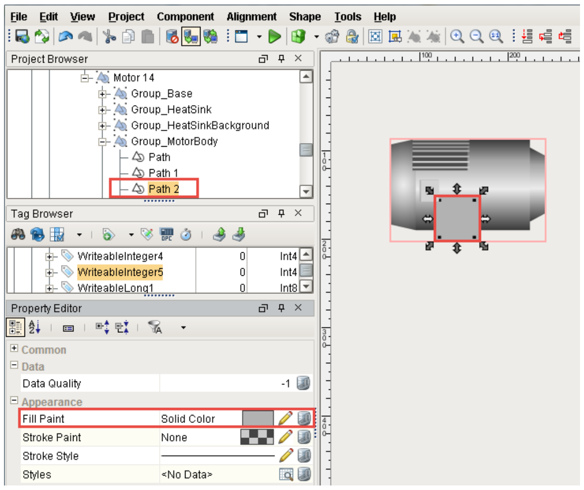
Expand the vector graphic in the **Project Browser**. You can see that the Motor is made up of many different components of the SVG. Each of these components can be colored independently.



In this example, let's select the square as the individual component of the SVG. Double click on the **M otor**, double click the **square**, and then single click on the square again.



Every individual component of the SVG has a Fill Paint property that can dynamically change based on a value. Click on the **Fill Paint** property binding icon.



Once you have the Fill Paint selected, you can combine it to a tag. Since there are a number of ways to bind the tag, let's explore our options in the following sections.
This page talks about coloring a specific part or parts of an existing SVG. There is also a way to tint the entire SVG one color. Click here to learn more about applying a Tint to the entire SVG.
While testing, if you are unable to write to a tag in the Tag Browser or through buttons in the designer, you might not have it set to allow writes. The first time you try to write, a window will popup in the Designer stating you are in Read Mode. Choose **'Enable Read/Write Mode'** so you can write to your tags for testing.

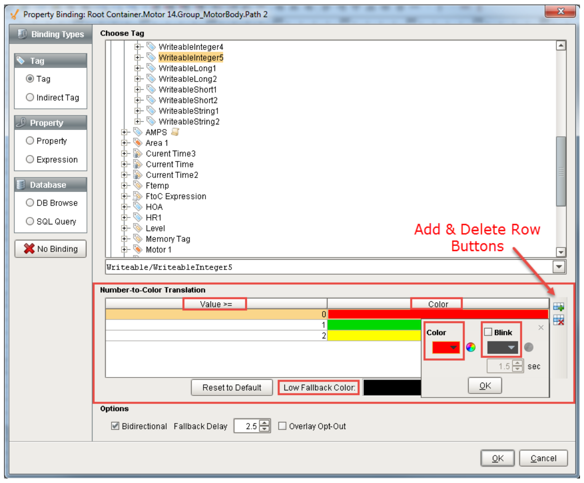
Number-to-Color Translation Table

The Fill Paint property uses the Number-to-Color Translation table in Ignition, commonly known as Color Mapping. This is where you map a value to a color. For every number range there is a set color. You can choose a different color for each value, and even make it blink between two different colors.

If you need to add or remove a value, simply use the add row and remove row icons on the right side of the Number-to-Color Translation table.

In this SVG example, value **'0'** is set to **red**, value **'1'** is set to **green,** and value **'2'** is set to blink between **yellow** and **orange**. If you are using this as a tint overlay, make sure each color is semi-transparent so you can see the actual symbol behind it.

There is a **Low Fallback Color** option so when a value falls below your lowest value, a default color can be set**.**



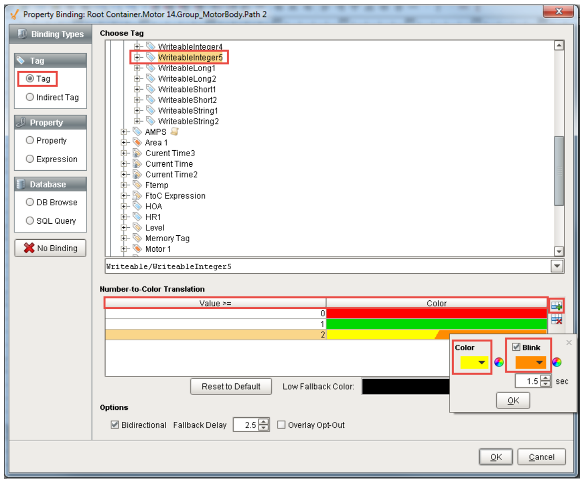
Tag Binding

A Tag Binding is a very straight-forward binding type. It simply binds a single property directly to a tag. For this example, every time the tag binding is evaluated, the Fill Paint property changes, pushing the new color to the component of the SVG. To learn more about tags, refer to the section on Understanding Tags.

Bind the **Fill Paint** property to any numeric tag**.** If you are not familiar with numeric, it's any number value. Integers are the best, but you can also use a boolean float, etc. In this example, **'WriteableIntege r5'** was selected.

Set the values in the **Number-to-Color Transition** table to drive the color of a component of an SVG.

- On Value **0**, make the component **red** by double clicking on the first row. Under **Color**, select **red** and press **OK**.
- On Value **1**, make the component **green**. Double click the second row and under **Color**, select **g reen** and press **OK**.
- Add a row by clicking the **Add Row** icon .
- On **Value 2**, make the square component blink between **yellow** and **orange**. Double click the third row and check the **Blink** box**,** and select the colors **yellow** and **orange**.
- Press **OK**.


Property Binding: Root Container.Motor 14.Group_MotorBody.Path 2

Choose Tag: Writeable/WriteableInteger5

Number-to-Color Translation

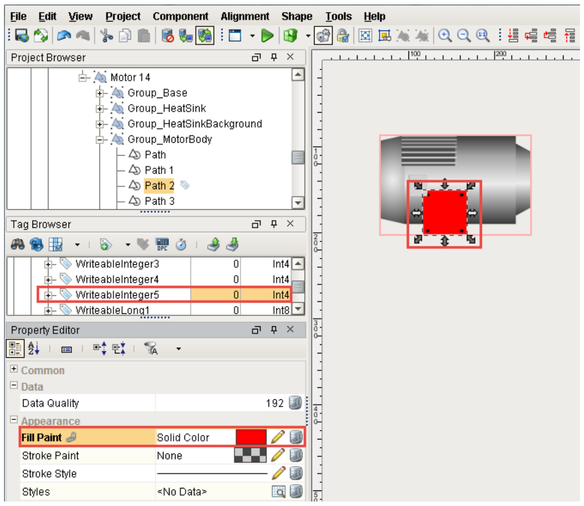
Value >=	Color
0	🟥
1	🟩
2	🟨

Color: yellow; Blink ✓: orange, 1.5 sec

Options: ✓ Bidirectional, Fallback Delay 2.5, Overlay Opt-Out

Once you press OK button, you'll notice that the component of the SVG changed color based on the current value of the tag. The Value is set to **'0'** and the color is **red**.

To change the tag value, go to the **Tag Browser**, double click on your tag value, and change the value from **'0'** to **'1'**. Press **Enter**. The component of the SVG will change from red to green. Now, change the Value from **'1'** to **'2'**, and the color will blink **yellow** and **orange**.



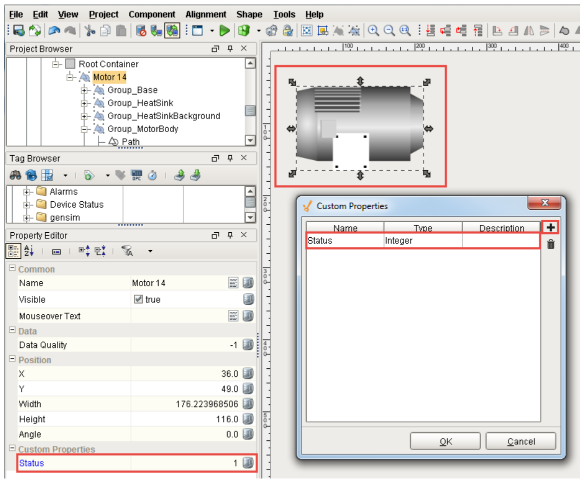
WriteableInteger3	0	Int4
WriteableInteger4	0	Int4
WriteableInteger5	0	Int4
WriteableLong1	0	Int8

Data Quality	192
Fill Paint	Solid Color
Stroke Paint	None
Stroke Style	
Styles	<No Data>

Property Binding

You can use the same number to color translation table with a property binding. Custom properties are extremely useful because they can turn your plain component into one customized for your particular use. Adding Custom properties like Status or Level can transform your SVGs into more than just an image. Refer to sections on Property Binding and Custom Properties to learn more.

Drag in a Motor SVG and create a custom property on the Motor called **'Status.'** In this example, right click on the **Motor** and select **Customizers > Custom Properties**. Enter the name **'Status'** and use the default **'Integer'** type. Press **OK**.



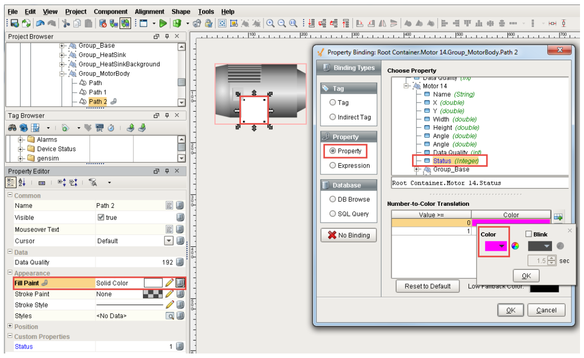
Property Editor:

Common	
Name	Motor 14
Visible	true
Mouseover Text	
Data	
Data Quality	-1
Position	
X	36.0
Y	49.0
Width	176.223968506
Height	116.0
Angle	0.0
Custom Properties	
Status	1

Custom Properties:

Name	Type	Description
Status	Integer	

Click the **Fill Property** icon, and select the **Property** binding type. Under the **Motor** object, select your new **'Status (Integer)'** property. Choose your colors and values. Press **OK**.



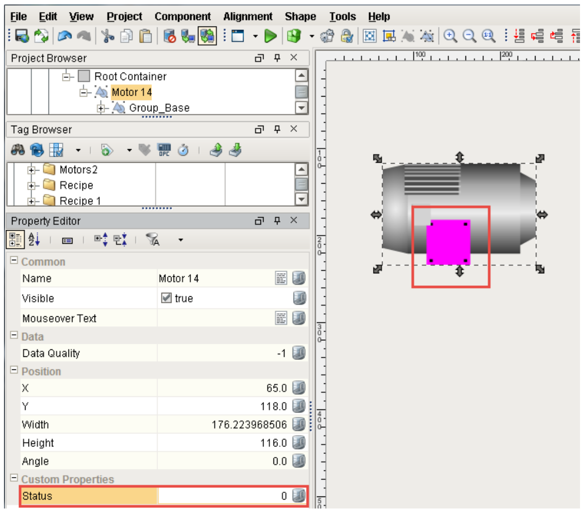
Root Container.Motor 14.Status

Property Editor:

Common	
Name	Path 2
Visible	true
Mouseover Text	
Cursor	Default
Data	
Data Quality	192
Appearance	
Fill Paint	Solid Color
Stroke Paint	None
Stroke Style	
Styles	<No Data>
Position	
Custom Properties	
Status	1

Once you press OK, you'll notice that the square component of the SVG changed color based on the value of the tag. An Integer value defaults to **'0'** and in this case the color is **purple**.

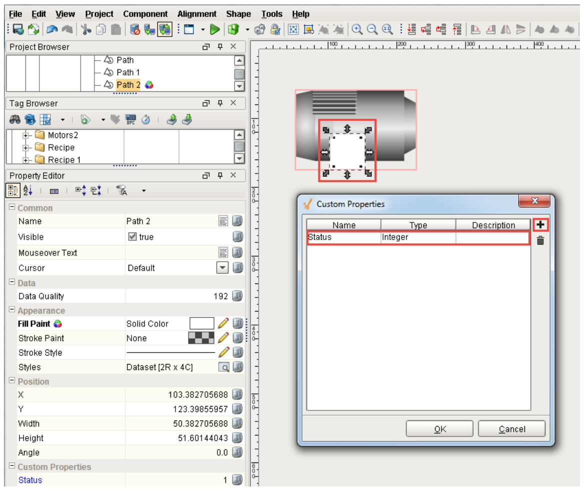
To change the tag value, go to the **Property Editor**, double click on the **Status** property value, and change the value from **'0'** to **'1'**. Press **Enter**. The component of the SVG will change from purple to whatever color you set **'1'** too.



Common	
Name	Motor 14
Visible	true
Mouseover Text	
Data	
Data Quality	-1
Position	
X	65.0
Y	118.0
Width	176.223968506
Height	116.0
Angle	0.0
Custom Properties	
Status	0

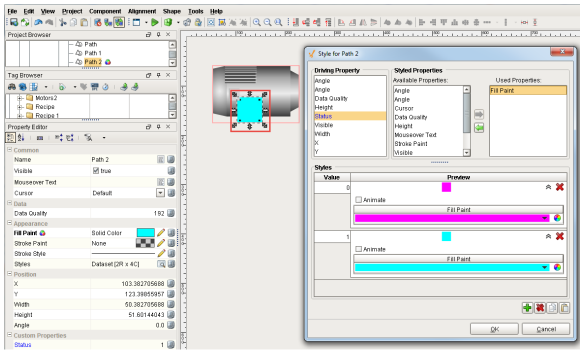
Style Customizer

You can also use the Style Customizer to achieve the same result as with the Property Binding. Like the previous example, we will need a Status property to drive the style. Let's create that same Status property on the inner component of the SVG. Select the component of the SVG, right click and scroll down to **Customizers > Custom Properties**.   Enter the name **'Status'** and use the default **'Integer.'** Pr ess **OK**.



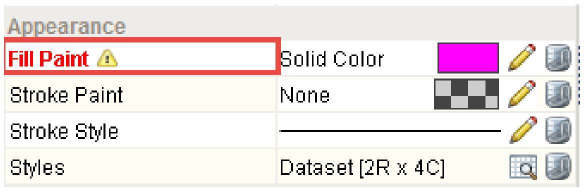
Name	Type	Description
Status	Integer	

With the inner component still selected, right-click and select **Customizers > Style Customizers**. Select the **Status** property as the driving property, select **Fill Paint** from the list of **Styled Properties**, and then choose your colors for each value, adding values if needed. Press **OK**.



To change the tag value, go to the **Property Editor**, double click on the **Status** property value, and change the value from **'0'** to **'1'**. Press **Enter**. The component of the SVG will change from purple to whatever color you set it too.

In the event you see the Fill Paint property highlighted in red, this is a warning message that you have two competing property bindings that should be resolved. In this case, this warning message popped up because the **Fill Paint** property has both a **Binding**, and a **Style** applied to it.



Appearance	
Fill Paint ⚠	Solid Color
Stroke Paint	None
Stroke Style	
Styles	Dataset [2R x 4C]

Direct Bindings

There are several types of Bindings that can be used to set colors directly, without using a mapping table.

Expression Binding

You can use the expression language to calculate a value. If you have a color that depends on multiple properties, then using an express is recommended to evaluate correctly. Here are a few examples. The first example returns a static color using the Fill Color property.

Expression

```
// binding on the Fill Color property
color(255,0,0) // static red color
```

This example takes a tag value and translates it to a color that ranges from white to blue and the tag value increases.

Expression

```
// binding on the Fill Color property
color(255,255,255-({tag value}/100*255)) // fades from white to blue when
tag value goes from 0 to 100 %
```

If you have multiple properties or tags, you can use the logic Expression Functions to select between a few colors.

Custom Properties

```
if({Tag1}>50,
        if({Tag2},
                3, // if tag1>50 and tag2 is true
                1), // if tag1>50 and tag2 is false
        if({Tag3},
                2, // if tag1<=50 and tag3 is true
                0)) // if tag1<=50 and tag3 is false
```

This example takes one integer value and selects from several options.

Expression Referencing a Tag

```
 // binding on the fill color property
switch({HOA tag},
        0,1,2, // off, on, hand
        color(255,0,0), color(0,255,0), color(255,255,0), // red, green,
yellow
        color(0,0,0)) // black (fallback color)
```

SQL Binding

SQL Query bindings only have to return an string that can be translated to a color. Any set of RGB values can be listed using an integer representation ie: "255,0,255" or HEX "FF00FF".

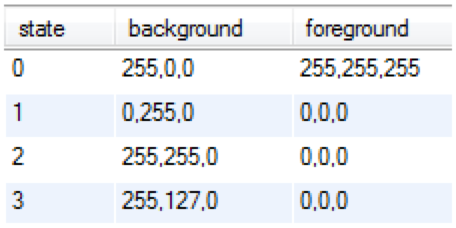
SQL Binding

```
-- A basic static query that returns the hex code for teal
SELECT "00FFFF"
```

SQL Binding

```
-- A static query that concatenates integers to return a string RGB Code
for yellow: "255,255,0"
SELECT CONCAT(255, ',', 255, ',', 0)
```

In most cases you will be using a table in the database. Assuming the following table (where background and foreground are strings):



state	background	foreground
0	255,0,0	255,255,255
1	0,255,0	0,0,0
2	255,255,0	0,0,0
3	255,127,0	0,0,0

SQL Query Binding

```
-- This query returns the background string RGB code for state #3 from the
colors_table
SELECT background
FROM colors_table
WHERE state = 3
```

Next ...

- Tint SVG

Tint SVG

Here is an example of how to apply a tint on SVGs (vector graphics) and Symbol Factory graphics.

INDUCTIVE UNIVERSIT

Tint SVG

Watch the Video

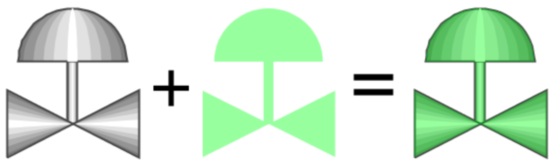
Symbol Factory Example: Animating the Tint on a Grayscale Symbol

Lets say you have chosen one of the many grayscale symbols, such as the **3-D Valve** symbol from the **V alves** category, and you want to tint the valve green when the valve is open, red when the valve has a fault, and keep it gray when the valve is closed. Suppose you have a tag called **ValveStatus**, that is 0 for closed, 1 for open, and 2 for faulted.

To tint the symbol

1. Drag the symbol onto the screen.
2. Duplicate the symbol by selecting it and choosing **Duplicate** from the **Edit** menu, or pressing **CT RL-D**.
3. Now, select the duplicate symbol, which will be above the original.
4. Press the Union button ( ) in the toolbar or find the **Union** item under the **Shape** menu. This will flatten the duplicated shape into a single shape.
5. Remove the outline by setting the **Stroke Paint** property to **No Paint**.
6. Bind the **Fill Paint** property to your **ValveStatus** tag.
7. Add three entries into the number-to-color translation table:

- fully transparent for zero
- 40% opaque green for 1
- 40% opaque red for 2



In summary, what we did to tint the symbol was to make a flat shape that had the exact same outline as the symbol, and used semi-transparent fills to achieve a tint effect for the underlying symbol.

Next ...

- Tank Cutaway

Tank Cutaway

Here is an example of how to cut out a shape on a tank (SVG) and use a level indicator to create a custom tank.

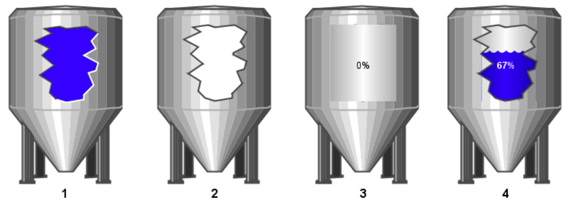
Tank Cutaway

Watch the Video

Symbol Factory Example: Using Cutouts on Tanks

The symbols in the Tank Cutaways category work well when combined with the symbols in the Tank. Use the following technique to make a dynamic cutaway tank display:

1. Drag a tank and a cutaway symbol onto the window.
2. Align the cutaway symbol on the tank where you'd like the cutaway to be placed.
3. Select the tank symbol, and then select the cutaway while holding **CTRL** to select both symbols.
4. Press the **Difference** button ( ) to use the cutaway symbol to (you guessed it!) cut away that area out of the tank.
5. Place a **Level Indicator** component on the area removed by the cutaway.
6. Push the Level Indicator below the tank, and bind it to a SQL tag to create a dynamic display.



Next ...

- Component Animation

Component Animation

Once you select a component or a group of components, you can resize, rotate, and move them using the mouse and keyboard. SVG images have **X**, **Y**, **Width**, **Height**, and **Angle** properties to make animation easy.

See also: Manipulating Components

Component Animation

Watch the Video

Next ...

- add link here

Using Special Components

The Designer offers some special components you can use when creating your projects. Control components, advanced tables, and PDF viewers are some examples of these special components.

In this section ...

Control Components

You can use various **Input** components (such as Numeric Text Field, Spinner, or Slider) and **Buttons** (such as 2 State Toggle, Multi-State, One-Shot, or Momentary) to write values back to the database or PLC.

INDUCTIVE UNIVERSITY

Control Components

Watch the Video

Input Components

The input components are the core of data entry in Ignition. These are the most commonly used components to allow data entry and along with labels and a submit button they make up almost all your forms in Ignition.

Touchscreen Mode

Ignition has a built-in Touchscreen Mode that allows access to Input components on computers that don't have a keyboard or mouse. This means that whenever Touchscreen Mode is turned on, a keyboard will be displayed when a user touches an Input component that expects a value to be entered. IE: Text field, Numeric Text Field, etc...

Defer Updates

Many of the Input components have a **Defer Updates** property that is True by default.

- If True, then any edits to the component's value will not be submitted until the enter key is presses. For example: if you type a word into a Text Field, the text property will not be updated until enter is pressed.
- If False, then the component's value will be submitted when each key is pressed. For example: if you type a word into a Text Field, the text property will be updated once for each letter in the word.

You can leverage this when you use an Input component as part of a filter for a SQL query on another component. In this case, turning off Defer Updates would re-run the query every time a key is pressed. This allows you to make your tables filter what rows are show as the operator starts typing.

Important: You should **not** turn off **Defer Updates** when using a bi-directional Tag binding for two reasons. 1) Writes are sent immediately but these bindings update at a set rate. This means your operators may not see the update and type in odd values. 2) When using Input components that have a numeric or text Format Pattern, this formatting will be applied at every keystroke. That means decimals can be truncated if the number is entered incorrectly.

Next ...

- Dropdown Component

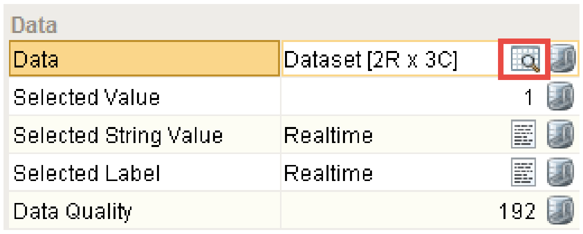
Dropdown Component

Dropdown lists are used when you want to select a single item from a list of options. The Dropdown component is under the Input section of the component palette. Simply drag it on to your window. The most important property of a Dropdown component is the Data property. It is a Dataset that contains one or more rows of data. Each of the rows are different options that you see on the component. Select the Dropdown component and click on the Dataset Viewer icon for the **Data** property to manually add some options.

INDUCTIVE UNIVERSITY

Dropdown

Watch the Video



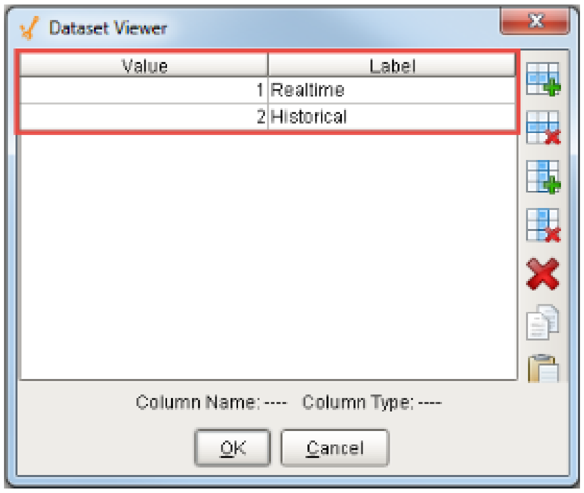
Data

Data	Dataset [2R x 3C]
Selected Value	1
Selected String Value	Realtime
Selected Label	Realtime
Data Quality	192

There are 3 ways you can use the **Data** property's dataset: a number/label pair, a single label column, and a code/label pair. Which mode is used depends on what columns are in the **Data** property's dataset, and will determine the values of the **Selected Value**, **Selected String Value** and **Selected Label** properties. Any additional columns that are added to the dataset will not affect these properties.

In the first mode, the first column is an integer (often an id) and the second column is a string (often a label). The first column is the **Value** column which is invisible to the user and is usually used in binding. The second column is the **Label** column and is visible to the user in the dropdown list.

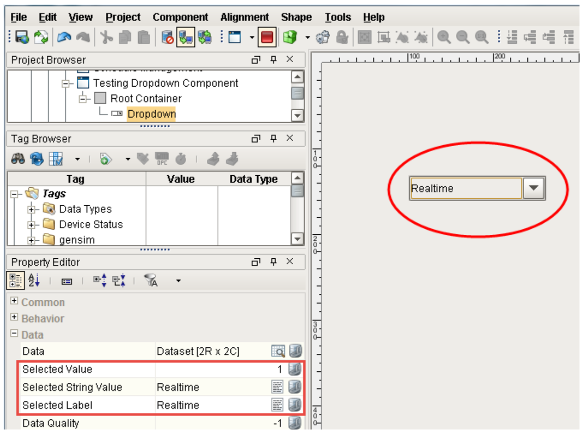
In this example, the dataset has 2 rows. Under the **Value** column, you see the integers **'1'** and **'2'** respectively. Under the **Label** column, you see the **'Realtime'** and **'Historical'** options.



Value	Label
1	Realtime
2	Historical

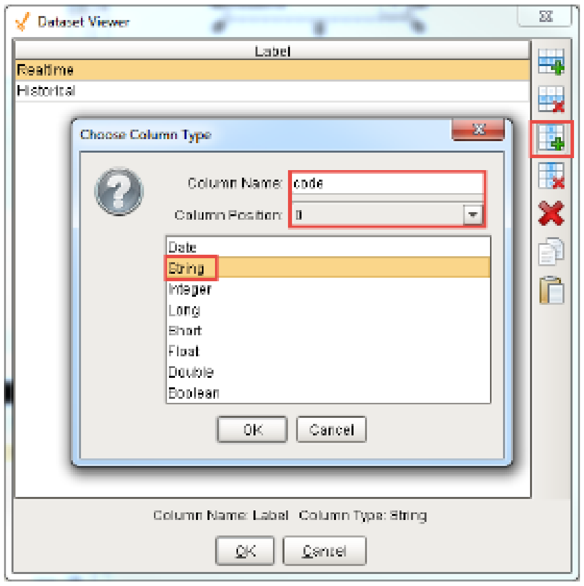
Column Name: ----   Column Type: ----

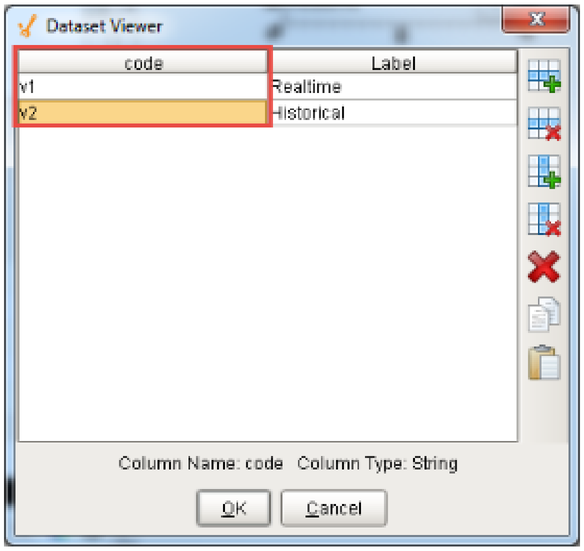
In **Preview Mode**, select from the list of dropdown options. Notice, you can only see the options in the **La bel** column. Select the **'Realtime'** option. You can see under the Data property section, the **Selected Value** is **'1'** which is the first (integer) column in the dataset and is not visible to users. The **Selected String Value** and the **Selected Label** are the same and show the **'Relatime'** option which is the second (string) column in the dataset.



Data	Dataset [2R x 2C]
Selected Value	1
Selected String Value	Realtime
Selected Label	Realtime
Data Quality	-1

The second mode you can use is a string/string combination which are 2 columns that are both strings. Remove the **Value** column from our example by selecting a cell in that column and clicking on the vertical x icon. Add another column by clicking on the verical plus icon. Call the new colum **'code,'** set the column position to **'0'** and make it a **'String.'** Both columns are now strings. Under the **'code'** column enter **'v1'** and **'v2.'** **Press OK.**

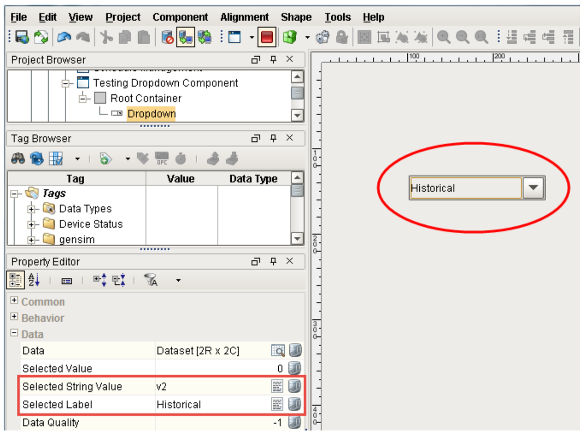




code	Label
v1	Realtime
v2	Historical

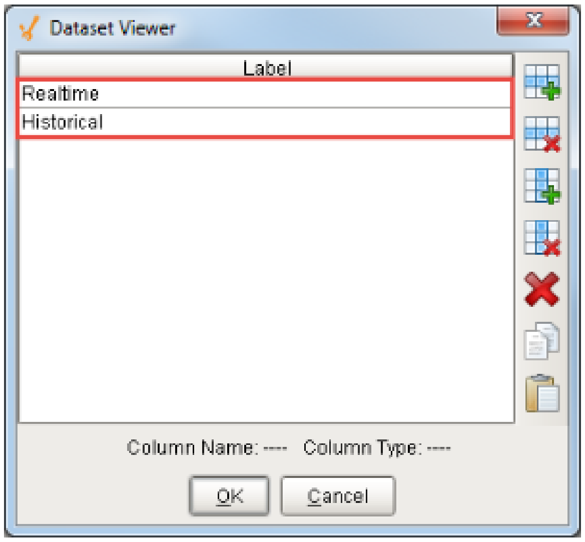
Because we added the **'code'** column in position **'0,'** the users will see the same 2 options in the dropdown list: **'Realtime'** and **'Historical**.' If you see **'v1'** and **'v2'** instead, that means your **'code'** column is not the first column.

In **Preview Mode**, click on **'Historical'** and you can see the **Selected String Value** is **'v2'** because the value is in the first column and the **Selected Label** is **'Historical.'**



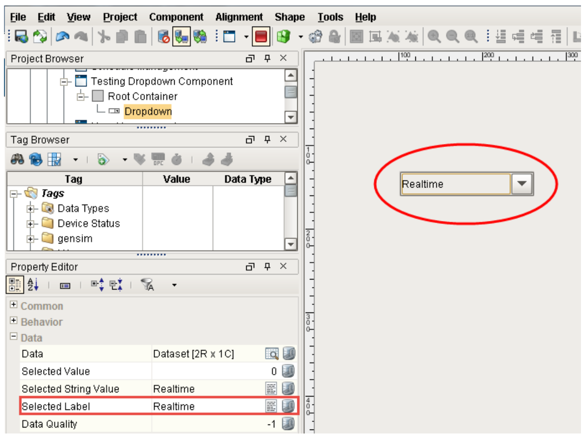
Data	Dataset [2R x 2C]
Selected Value	0
Selected String Value	v2
Selected Label	Historical
Data Quality	-1

The third mode you can use is simply one string column. Remove the **'code'** column and keep the **'Label'** column the same. Again, you are going to see the same 2 options in the dataset viewer: **'Realtime' and 'Historical.'** Note, that this only applies if there is exactly one column in this dataset.



Label
Realtime
Historical

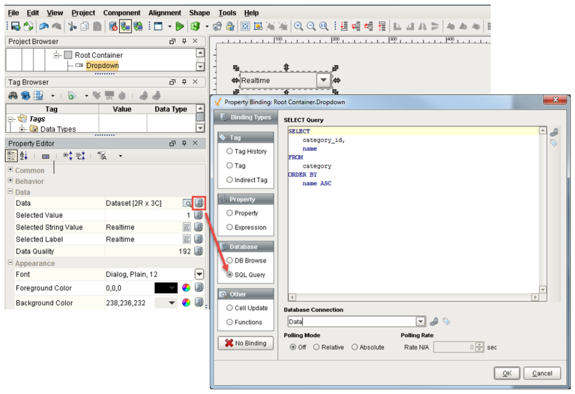
In **Preview Mode**, select the **'Realtime'** option. You will see the same value in the **Selected Label** and **Selected String Value** properties since there is only the one column.



Data	Dataset [2R x 1C]
Selected Value	0
Selected String Value	Realtime
Selected Label	Realtime
Data Quality	-1

Now, you can set these dropdown options manually or bind the Data property. In this example, you can take a Dropdown List on the window and bind the Data property using a SQL query.

Click on the binding icon for the Data property. Select the SQL Query Binding Type and type a query that brings back an ID and Name from one of your tables in the database.



Property Binding: Root Container.Dropdown

SELECT Query

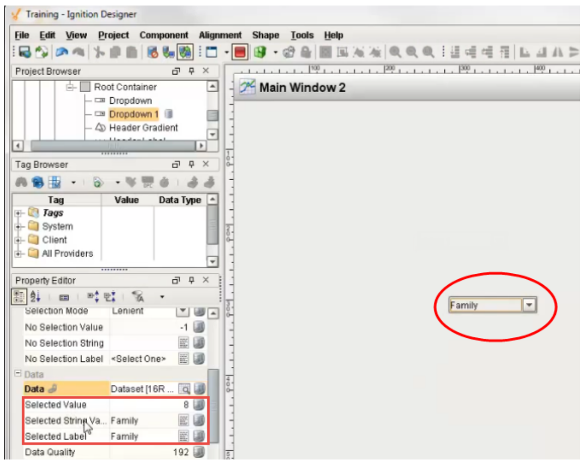
```
SELECT
    category_id,
    name
FROM
    category
ORDER BY
    name ASC
```

Database Connection: Data

Polling Mode: [✓ Off] Relative Absolute

Polling Rate: Rate N/A 0 sec

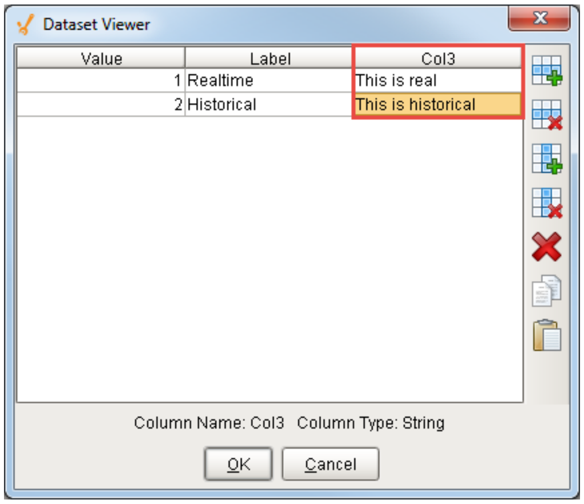
Click OK to see all the options that came back from the database. In **Preview Mode**, select any one of these options and you can see the **Selected Value**, **Selected String Value**, and **Selected Label** as shown in the following example.



Selection Mode	Lenient
No Selection Value	-1
No Selection String	
No Selection Label	<Select One>
Data	Dataset [16R ...
Selected Value	8
Selected String Va...	Family
Selected Label	Family
Data Quality	192

Another feature of the Dropdown component is you can show more than one column to a user. Drag another Dropdown component to your window. Go to the Dataset viewer and add some options manually. Instead of having only one or two columns you can add as many as you want. The first column needs to be either an integer or a string. Any additional columns will show up in the dropdown.

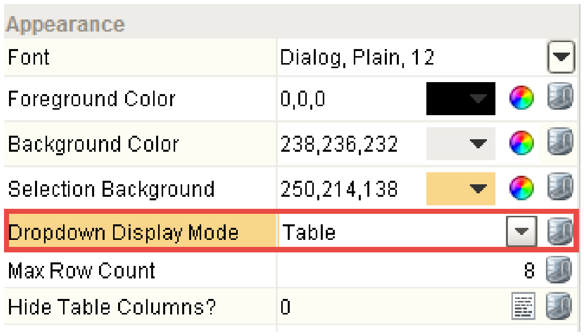
Open the Dataset View, under **Value**, enter **‘1’** and **‘2.’** Under **Label**, enter ‘**Realtime**’ and ‘**Historical**.’ Add another column and call it ‘**Col3**’ and make it a string and enter **‘This is real’** and **‘This is historical**.’


Dataset Viewer

Value	Label	Col3
1	Realtime	This is real
2	Historical	This is historical

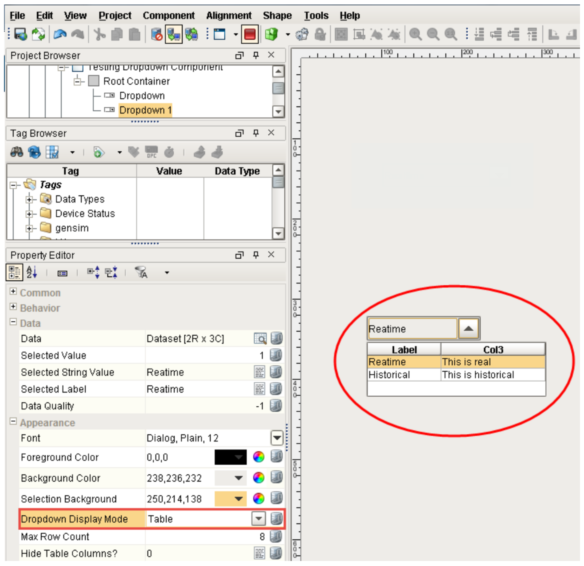
Column Name: Col3 Column Type: String

Press OK. Go to the **Property Editor > Appearance** and set the **Dropdown Display Mode** to **‘Table’** to see both columns.


Appearance

Font	Dialog, Plain, 12
Foreground Color	0,0,0
Background Color	238,236,232
Selection Background	250,214,138
Dropdown Display Mode	Table
Max Row Count	8
Hide Table Columns?	0

Now you can see **Label** and **Col3** and select between those different rows. It’s a nice way to show more information in the Dropdown list. The **Selected Value**, **Selected String Value**, and **Selected Label** properties will behave the same as previous examples ignoring any columns beyond the second.



Tag	Value	Data Type

Data

Data	Dataset [2R x 3C]
Selected Value	1
Selected String Value	Reatime
Selected Label	Reatime
Data Quality	-1

Appearance

Font	Dialog, Plain, 12
Foreground Color	0,0,0
Background Color	238,236,232
Selection Background	250,214,138
Dropdown Display Mode	Table
Max Row Count	8
Hide Table Columns?	0

Reatime

Label	Col3
Reatime	This is real
Historical	This is historical

Next ...

- Table Component

Table Component

The **Table** component is very powerful, very flexible, and easy to configure. It allows you to easily display your tabular data in a variety of ways.

Table

Watch the Video

Next...

- Power Table Component

Power Table Component

The **Power Table** is a much more customizable version of the table component, and has many more features. The Power Table contains advanced features for displaying your tabular data, such as drag-and-drop rows, multi-column sorting, column filtering, and cell-spanning. Much of the customization comes through extension functions which are available to configure how each cell of the table looks, how the headers look, and more.

Power Table

Watch the Video

Next...

- Tree View Component

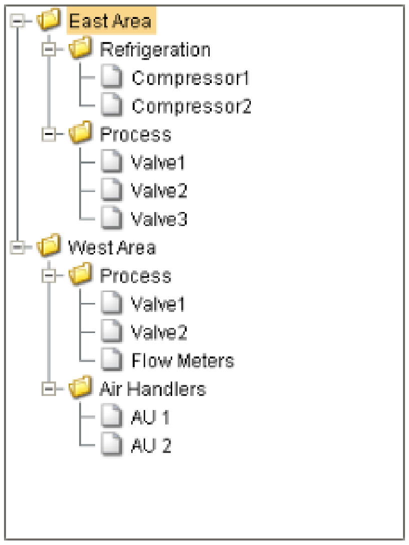
Tree View Component

The **Tree View** component displays and organizes data in a dataset in a tree hierarchy. The Tree View component is found under the Tables tab of the component palette. **Drag the Tree View component on to your window.**   In **Preview Mode**, expand the folders to see the data.   The Tree View component can be used for navigation in Ignition.

INDUCTIVE UNIVERSITY

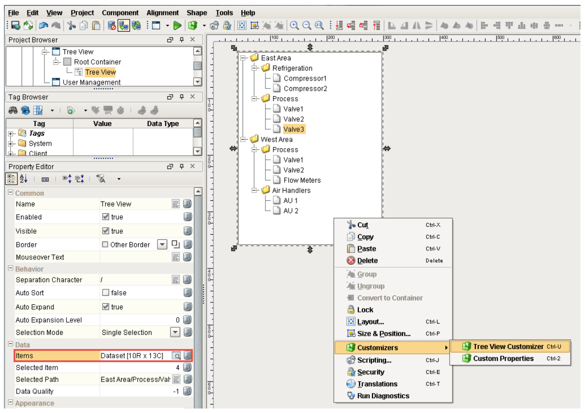
Tree View

Watch the Video



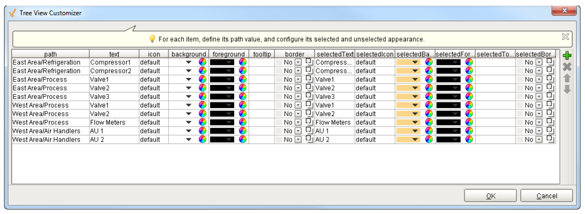
In fact, Ignition has a built in skeleton project called Multi-tier Navigation that uses the Tree View component specifically for that purpose. There is one difference between the Multi-tier Navigation and the Tree View component. The Multi-tier Navigation has one extra column in the Dataset Viewer called '**windowPath'** that does not show up in the Tree View Customizer. You will need to open the Dataset Viewer to modify it.

To configure the Tree View component, go to the **Data property** called **'Items'** next to the Dataset Viewer. This contains all the items in the tree. **Right click** on the **Tree View component** and scroll down to **Customizers > Tree View Customizers** to configure these items.



Common
Name: Tree View
Enabled: ✓ true
Visible: ✓ true
Border: Other Border
Mouseover Text:
Behavior
Separation Character: /
Auto Sort: false
Auto Expand: ✓ true
Auto Expansion Level: 0
Selection Mode: Single Selection
Data
Items: Dataset [10R x 13C]
Selected Item: 4
Selected Path: East Area/Process/Val
Data Quality: -1
Appearance

You can add one or more rows of data in the Tree View Customizer. Every row must have a **path** and **te xt** to display. In the example below, look at the first and second rows. The path is East Area /Refrigeration. This path denotes which folder to put an item inside of. The path is separated by a **'/'** which is a **Separation Character** by default, and shows that there is an East Area inside a Refrigeration folder containing 2 items called Compressor 1 and Compressor 2.   You will notice in the Tree View Customizer that you have other columns you can customize based on how you want particular items to look.

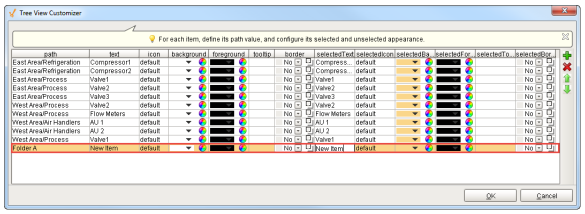


Tree View Customizer

For each item, define its path value, and configure its selected and unselected appearance.

path	text	icon	background	foreground	tooltip	border	selectedText	selectedIcon	selectedBa...	selectedFor...	selectedTo...	selectedBor...
East Area/Refrigeration	Compressor1	default		⬛		No	Compress...	default	🟧	⬛		No
East Area/Refrigeration	Compressor2	default		⬛		No	Compress...	default	🟧	⬛		No
East Area/Process	Valve1	default		⬛		No	Valve1	default	🟧	⬛		No
East Area/Process	Valve2	default		⬛		No	Valve2	default	🟧	⬛		No
East Area/Process	Valve3	default		⬛		No	Valve3	default	🟧	⬛		No
West Area/Process	Valve1	default		⬛		No	Valve1	default	🟧	⬛		No
West Area/Process	Valve2	default		⬛		No	Valve2	default	🟧	⬛		No
West Area/Process	Flow Meters	default		⬛		No	Flow Meters	default	🟧	⬛		No
West Area/Air Handlers	AU 1	default		⬛		No	AU 1	default	🟧	⬛		No
West Area/Air Handlers	AU 2	default		⬛		No	AU 2	default	🟧	⬛		No

In the following example, you can add another row after the last item in the Tree View Customizer window and set the path to **Folder A.** For this example, enter the name of your item as **'New Item'** in the **text** field. You can also set an icon, the foreground, a tooltip, border, when the item is selected and when it's not selected if you desire. Make sure the name of the item is the same for the **text** and **selecte dText** fields. You can add one or more rows and move them up and down.

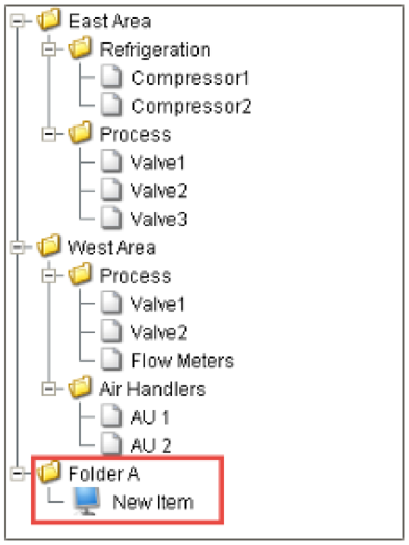


Tree View Customizer

For each item, define its path value, and configure its selected and unselected appearance.

path	text	icon	background	foreground	tooltip	border	selectedText	selectedIcon	selectedBa...	selectedFor...	selectedTo...	selectedBor...
East Area/Refrigeration	Compressor1	default		⬛		No	Compress...	default	🟧	⬛		No
East Area/Refrigeration	Compressor2	default		⬛		No	Compress...	default	🟧	⬛		No
East Area/Process	Valve1	default		⬛		No	Valve1	default	🟧	⬛		No
East Area/Process	Valve2	default		⬛		No	Valve2	default	🟧	⬛		No
East Area/Process	Valve3	default		⬛		No	Valve3	default	🟧	⬛		No
West Area/Process	Valve2	default		⬛		No	Valve2	default	🟧	⬛		No
West Area/Process	Flow Meters	default		⬛		No	Flow Meters	default	🟧	⬛		No
West Area/Air Handlers	AU 1	default		⬛		No	AU 1	default	🟧	⬛		No
West Area/Air Handlers	AU 2	default		⬛		No	AU 2	default	🟧	⬛		No
West Area/Process	Valve1	default		⬛		No	Valve1	default	🟧	⬛		No
Folder A	New Item	default		⬛		No	New Item	default		⬛		No

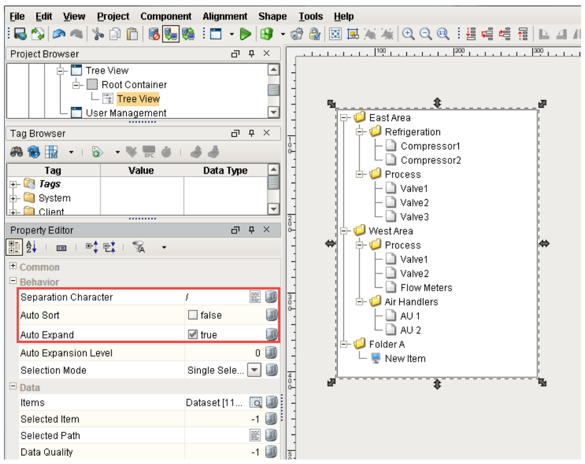
When you press **OK**, your new row will be saved in the dataset and you can see it on the Tree View. If you expand Folder A, you will see the **'New Item'** with the icon that you specified.



The Tree View component has a couple of other properties that are important like the **Separation Character** which we are using in our path name. By using the **Separation Character** or **'/'**, the path is broken up into folders.

There is a property called **Auto Sort** that automatically sorts all the items in the tree into alphabetical order. If you uncheck **Auto Sort**, it will be sorted by the order of the dataset as shown in the example below.

There is another property called **Auto Expand**. If set to True, the folders will all automatically be expanded. If your Tree View is not expanded, right click to close the window, and by opening it again you will see that all the folders are now expanded in the Tree View.



Common
Behavior

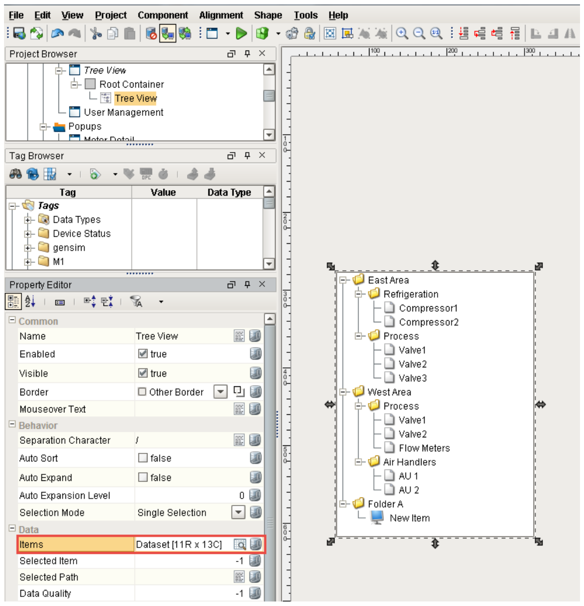
Separation Character	/
Auto Sort	false
Auto Expand	✓ true
Auto Expansion Level	0
Selection Mode	Single Sele...

Data

Items	Dataset [11...
Selected Item	-1
Selected Path	
Data Quality	-1

```
East Area
  Refrigeration
    Compressor1
    Compressor2
  Process
    Valve1
    Valve2
    Valve3
West Area
  Process
    Valve1
    Valve2
    Flow Meters
  Air Handlers
    AU 1
    AU 2
Folder A
  New Item
```

Now, the dataset has all items shown in the tree view. You can bind this to a SQL query to bring back data from the database in order to organize this data into the tree.



Common

Name	Tree View
Enabled	✓ true
Visible	✓ true
Border	Other Border
Mouseover Text	

Behavior

Separation Character	/
Auto Sort	false
Auto Expand	false
Auto Expansion Level	0
Selection Mode	Single Selection

Data

Items	Dataset [11R x 13C]
Selected Item	-1
Selected Path	
Data Quality	-1

```
East Area
  Refrigeration
    Compressor1
    Compressor2
  Process
    Valve1
    Valve2
    Valve3
West Area
  Process
    Valve1
    Valve2
    Flow Meters
  Air Handlers
    AU 1
    AU 2
Folder A
  New Item
```

Next...

- IA Labs PDF Viewer

Web Browser Component

The **Web Browser** component in Designer allows you to embed a full web browser inside of an Ignition Client. This component becomes available in Designer after you download the Web Browser module from the Inductive Automation's website. The Web Browser module installs the same way as any other modules. Once this component is added onto a window, it will behave just like any other web browser when it is inside a Client.

Web Browser

Watch the Video

Basic Usage

To add the Web Browser component to Ignition, first install the **Web Browser Module** (make sure you save your project before installing the module as the designer will need to restart). A new section titled 'Web Browser' will appear in the Designer's Component Palette. Drag the Web Browser on to a Vision window just like any other component.

Behavior

Popups AllowedPopups are allowed by default. If unchecked, popup windows will not be shown to the user.Zoom LevelZoom level is set to 0.0 by default. Positive numbers zoom in, negative numbers zoom out. The zoom level will not be applied until the page has finished loading.Show Navigation ButtonsShow Navigation Buttons is checked by default. If unchecked, the user will not see the forward, back, or reload buttons, and will not be able to type a URL into the navigation bar. They will still be able to navigate by clicking links on web pages.

Data

Mode	Mode controls whether Starting URL or Starting HTML will be used.
Starting URL	Starting URL is blank by default. When the client is started, the Web Browser will navigate to the URL entered here if mode is URL.
Starting HTML	Starting HTML is `<html><body>&nbsp;</body></html>` by default, which gives a blank page. When the client is started, the Web Browser will load the HTML entered here if the mode is HTML.
Use Proxies	Use Proxies is unchecked by default. If checked, the Web Browser will try to use the proxy settings below.
HTTP Proxy Server	HTTP Proxy Server sets the proxy server for HTTP connections. This setting is only used when Use Proxies is checked.
HTTP Proxy Port	HTTP Proxy Port sets the proxy port for HTTP connections. This setting is only used when Use Proxies is checked.
HTTPS Proxy Server	HTTPS Proxy Server sets the proxy server for HTTPS connections. This setting is only used when Use Proxies is checked.
HTTPS Proxy Port	HTTPS Proxy Port sets the proxy port for HTTPS connections. This setting is only used when Use Proxies is checked.
FTP Proxy Server	FTP Proxy Server sets the proxy server for FTP connections. This setting is only used when Use Proxies is checked.
FTP Proxy Port	FTP Proxy Port sets the proxy port for FTP connections. This setting is only used when Use Proxies is checked.
Proxy Exceptions	Domains listed here, such as <local>, will not use the proxy server. This setting is only used when Use Proxies is checked.
Use Proxy Authentication	If Use Proxy Authentication is checked, the browser will use the username and password below for proxy authentication. This setting is only used when Use Proxies is checked.

Proxy Username	The username to use for proxy authentication. This setting is only used when Use Proxies and Use Proxy Authentication are checked.
Proxy Password	The password to use for proxy authentication. This setting is only used when Use Proxies and Use Proxy Authentication are checked.

Advanced Usage

The underlying browser component is available in scripting through the getBrowser() method. Documentation on the browser component is available at the JxBrowser Programmer's Guide. The Inductive Automation support team is unable to provide detailed advice on scripting with this component. Furthermore, they are unable to provide troubleshooting beyond the basic functionality of the module.

Linux Distributions

Some 32-bit Linux distros are missing a needed library for running the Web Browser: libXss.so.1

Steps that fixed it in Ubuntu 12.04:

- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get install ia32-libs-multiarch

Next...

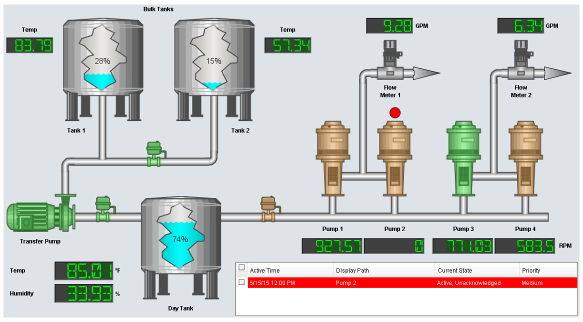
High Performance HMI Techniques

High performance HMI techniques and practices allow the designer to create designs and displays which help the viewer make the best decision in the shortest amount of time after interacting with the HMI.

High performance HMIs often look basic and even boring. Advocates of high performance HMI design would argue that boring is appropriate for a process that is running smoothly. The power of this design philosophy is when something does go wrong. A high performance HMI will quickly guide the user to the source of the problem through its innovative use of color and alarms.

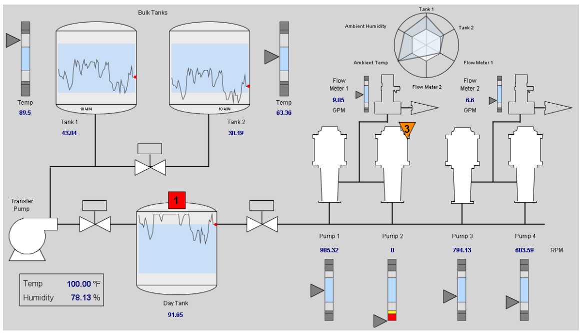
In Ignition, you create the high performance HMIs by using components such as moving analog indicators, sparkline charts, and radar charts. Here is a comparison of a traditional HMI next to a high performance HMI.

Traditional HMI



Active Time	Display Path	Current State	Priority
5/15/15 12:00 PM	Pump 2	Active, Unacknowledged	Medium

High Performance HMI



In this section ...

Use of Color

Colors are an important consideration when designing a high performance HMI where gray-scale colors are used instead of the traditional bright red, green, and blue colors.

The person that designs the HMI must understand the diverse audience that may view the HMI. For example, an operator may look at a motor on a traditional HMI. If it the HMI has colored the motor green, the operator may conclude that the motor is running. However, if a maintenance technician looks at the same motor, he may conclude that the motor is not faulted. These are logical conclusions. They reflect the respective interest of the person viewing the HMI where the operator wants the motor to run and the maintenance technician wants it to be working.

In reality, the motor is simply "scheduled" to run, meaning when it runs depends on its control mechanism. For example, the motor may run when there is product on the line or boxes on the conveyor. In other words, the motor may be periodically starting and stopping automatically.

A high performance HMI can eliminate this confusion by introducing a color that signifies a state of "scheduled." A common high performance HMI practice is to use a dark gray to signify a "scheduled" state for equipment. This color should never compete with more alerting colors that "pop" from the HMI resulting in the viewer's eye's being drawn to the portion of the HMI where the problem may be occurring.

Use of Color

Watch the Video

Colors and Alarm Indicators

Color can play an important role in how an operator responds when problems do occur. High performance HMI design refrains from coloring equipment when the equipment is in a state of fault. For example, some equipment may still run when faulted. Instead, an optimized solution is to place an alarm indicator near the equipment in such a way that when the equipment is undergoing some fault, the alarm indicator renders with the appropriate color and shape. The object and the consequential color should signify the most important alarm state occurring for the equipment at that current time.

Technical Considerations with Colors

Some HMIs in industrial settings may temporarily lose their ability to render color because of various environmental factors. High performance HMI design incorporates this possibility by encouraging the use of descriptive text with color. For example, motors of two different colors may look the same on a color deficient HMI resulting in confusion for the viewer. Even worse, the viewer may misinterpret the motor state and assume everything is fine. However, if each motor has a descriptive text such as "Motor 1 is Faulted" and "Motor 2 is Running", the problem associated with a faulty HMI failing to display color is largely reduced by the HMI's high performance design.

Accommodating Color Blind Viewers

Common color combinations such as red and green and blue and purple cannot adequately be distinguished by those with color blindness. High performance HMI design accommodates color blind uses by combining colors with descriptive text as well as incorporating alarm indicators in unique shapes.

Next...

- Alarm Indicator

Alarm Indicator

The high performance HMI design techniques make use of an object called the **Alarm Indicator** which displays a colored shape when there is a problem. This works well with the high performance HMIs as color is only used when there is a problem. The Alarm Indicator can contain descriptive text in addition to the shape and color, and is usually placed near the component that is causing the problem. You can import the Alarm Indicator, shown in the example below, from the Cloud Templates.

The Alarm Indicator represents different levels of alarm with different shapes, color, and descriptive text. For example, a motor that exists in an industrial setting is monitored by a high performance HMI. There are two alarms on the motor. The first is a critical alarm associated with the motor becoming seized, will display as a red rectangle with the letter 1. The second is a high priority alarm associated with the motor when overheating, will display as a yellow triangle with the number 2. An Alarm Indicator is placed near the motor, positioned in such a way to clearly show motor that the indicator is referring to.

Given this scenario, a high performance HMI will show the Alarm Indicator as in the following examples:

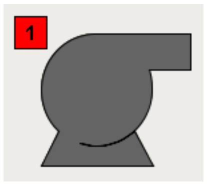
INDUCTIVE UNIVERSITY

Alarm Indicator

Watch the Video

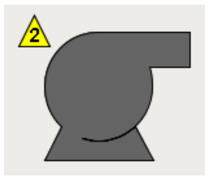
Example 1

The motor is critically faulted. This is the highest priority.



Example 2

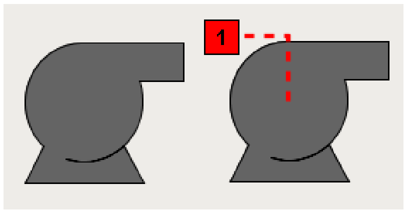
Motor is overheating resulting in a high alarm.



Reducing Ambiguity

A high performance HMI design technique to reduce ambiguity incorporates a line from the Alarm Indicator to the object experiencing the alarm.

The following example shows how this technique can ensure that the viewer associates the correct Alarm Indicator with the correct motor.



Next...

- Moving Analog Indicator

Moving Analog Indicator

The Moving Analog Indicator in Ignition, is another component, on the Display tab of the Component Palette, that fits with the high performance HMI techniques and practices. This component displays an analog value in context with other information about that value so that you can visually quickly see if the value is in the normal range or not.

When the screen shows just a number on the screen, it is very difficult to see if that number is within the normal range or outside of it.

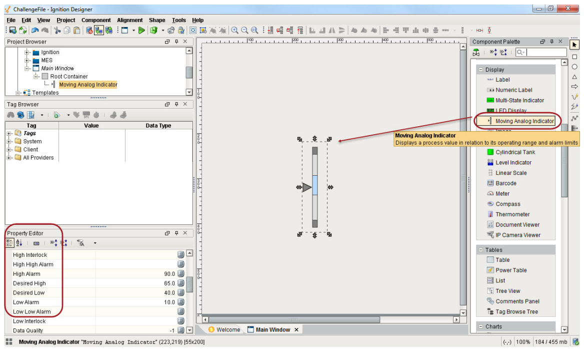
Moving Analog Indicator

[Watch the Video](#)

To use the moving analog indicator component

1. In Designer, drag the **Moving Analog Indicator** from the **Display** tab of the Component Palette to your workspace.

2. Use the different properties (for example, Range High, Range Low, and Process Value) in **Property Editor** to set the different range values.
   The Desired High and Desired Low properties set the normal ranges in a blue color.



Moving Analog Indicator
Displays a process value in relation to its operating range and alarm limits

Property Editor

High Interlock	
High High Alarm	
High Alarm	90.0
Desired High	65.0
Desired Low	40.0
Low Alarm	10.0
Low Low Alarm	
Low Interlock	
Data Quality	-1

Moving Analog Indicator "Moving Analog Indicator" (223,219) [55x200]

Next...

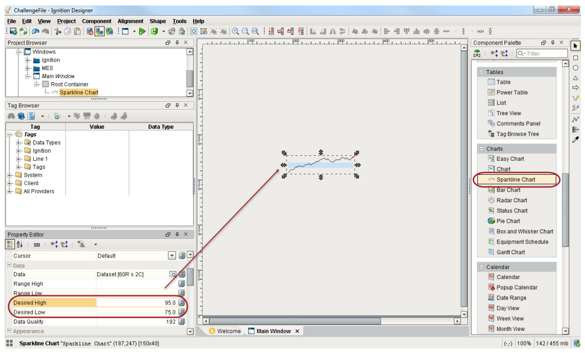
- [Sparkline Chart Component](#)

Sparkline Chart Component

The sparkline chart is a minimalistic chart component that displays a line-chart history for a single datapoint also known as trends. Sparkline or trend charts were invented as a way to show a great deal of contextual information in a very small amount of space.

To use the sparkline chart component

1. In the Designer, drag the **Sparkline Chart** from the **Charts** tab of the Component Palette to your workspace.
2. Use the properties in **Property Editor** to set the different range values.
   For example, you can use the Desired High and Desired Low properties to set the normal ranges in a blue color, as shown below.
   The red dot on the Sparkline chart represents the most latest value.



Cursor: Default
Data
Data: Dataset [60R x 2C]
Range High
Range Low
Desired High: 95.0
Desired Low: 75.0
Data Quality: 192
Appearance

Sparkline Chart "Sparkline Chart" (187,247) [150x40]

100% 142 / 455 mb

Sparkline Chart

Watch the Video

Next...

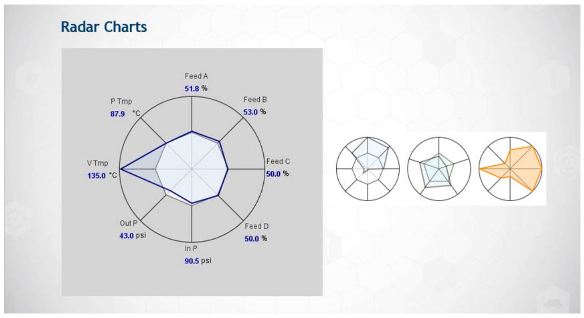
- Radar Chart Component

Radar Chart Component

Another high performance HMI technique is using the Radar Chart. Radar Charts (also know as, spider charts or spider plots) are useful for displaying several values at once. Each value is plotted on a separate axis. The chart draws a line between the different values, which create a shape that changes as those values change. Inside the chart, there is a circle that represents what the chart would look like if all of its values were in their ideal range. So the chart lets you quickly see where the values are in comparison to where they should ideally be.

Radar Chart

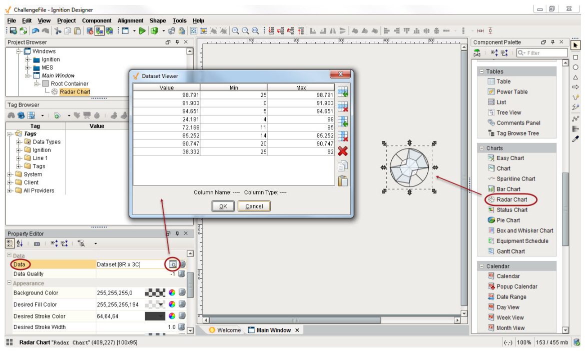
Watch the Video



To use the Radar Chart component

1. In the Designer, drag the **Radar Chart** from the **Charts** tab of the Component Palette to your workspace.

2. Click on the **Dataset Viewer** icon for the **Data** property in **Property Editor** to set the values. For each value you enter, an axis is created. For example, you can see below that there are 8 values in the Dataset Viewer which corresponds to the 8 axis on the Radar Chart. The Min and the Max represent the acceptable limits of the Value.



Value	Min	Max
98.791	25	98.791
91.903	0	91.903
94.651	5	94.651
24.181	4	88
72.168	11	85
85.252	14	85.252
90.747	20	90.747
38.332	25	82

You can show live tag values on the Radar Chart by binding the Data property to the Cell Update binding type. For example, suppose you had a group of motors. Each motor had a value for is current amps, a minimum number of amps, and a maximum number of amps. A Cell Update binding can represent each motor. The value, min, and max columns of the Radar Chart's dataset can be bound to each motors value, minimum, and maximum amps. The result is that different motors with different thresholds of tolerance can be visually represented on the Radar Chart.

Next...

Templates

Templates are a simple but a very powerful feature in Ignition that you can use with the Vision windows. The power comes from the ability to modify only the template in one location while affecting all of the instances used throughout the project.

HMI and SCADA systems typically have a lot of repetitions in their screens. You might use a group of the same components over and over within your project. The data driving each set of graphics is different, but the graphics themselves are copies of each other. You can make a single template and use instances of the template over and over again.

In this section ...

Template Basics

When using templates, you define the graphical display in one place, called the master template. You then use this master template many times in your project on multiple windows, thus making a number of template instances. Any changes made to the master template are then instantly reflected in all of the template instances. Using templates early in your project development for any repeating displays, can save a significant amount of time later on.

Without templates, the only way to do this is to copy-and-paste the components each time you want to represent lets say, a motor. This is simple, and it works, but it can cause major headaches and time consuming corrections later on. Because if you ever want to make a change to how motors are represented, you're stuck making the change to each copy of the group.

Template Properties

Because the primary use of templates are the ease of maintaining repeated user interface elements, correct use of parameters is of high importance. Parameters allow each template instance to reference different data elements of repeated data configurations.

This is very similar to the concept of Parameterized Popup Windows. In that case, any **Custom** property on the **Root Container** of the window is used as a parameter, and is passed into the window when it is opened. Templates take this idea one step further.

Template Parameters and Internal Properties

When you open the **Custom Properties** window (right-click the checkered-box of the template and select **Customizers > Custom Properties**), you'll notice it is different than all other components. There are two kinds of custom properties here, as follows:

- Template Parameters
  These parameters appear on each template instance, allowing each instance to be configured differently. Commonly, this is some sort of indirection. For example, if you have a template representing motors, you might have MotorNumber as a parameter property. Then you can use that property as an indirection variable in other bindings within the template. Parameter properties are not bindable from within the template master design. When you use the template to create a template instance, the property becomes bindable. This ensures that the property only has a single binding configured for it.

- Internal Properties
  These properties cannot be used as parameters. It shows up when designing the template master, but it does not show up on the template instances. Internal properties are bindable from within the template master design. These properties are intended to be used for the internal workings of the template.

Indirection and UDT Tags

There are two primary ways to achieve indirection when using templates. Let's continue to use the example of a motor. Your system has many motors in it, and your template is used to display the status of the motors and control the motor's running mode. The goal is to be able to drop instances of the template onto your windows, and configure them in a single step to point to the correct motor's tags.

If the tags representing the datapoints of each motor are arranged in an orderly way in folders or with a consistent naming convention, you can use standard indirection to configure your template. You can add a parameter such as MotorNum to the template. Then you configure the contents of the template using indirect tag binding, where the value of MotorNum is used for the indirection.

If your motors are represented by a custom tag data type, you can save some effort and use a property of that tag type directly. Make your indirection property the same type as your custom data type. Then you can use simple property bindings to configure the components inside your template. When you create a template instance, you can simply bind that property directly to the correct Motor tag, and all of the sub-tags of motor are correctly mapped through the property bindings.

The Drop Target Parameter

When you specify parameters in the **Custom Properties** window (right-click the checkered-box of the template and select **Customizers > Custom Properties**), you can set one of the parameters as the Drop Target. This allows you to drop tags onto your template instances to facilitate even quicker binding. For example, let's say that you have a parameter that is an integer and you've made it the drop target. If you drop an integer tag onto a window, your template appears in the menu dropdown list of components which is displayed. Choosing your template creates a template instance and binds that parameter to the tag.

This also works for UDT tags. Lets say you have a custom data type called Motor and a template with a Motor-typed parameter set as the drop target. If you drop a motor tag onto a window, it creates an instance of your template automatically. If you have more than one template configured with Motor drop targets, you have to choose which template to use.

In this section ...

About Templates

Templates are a simple but very powerful feature that you can use with your Vision windows. Using templates, you would define the graphics and parameters in one place, called the master template. You then create instances throughout your project that update whenever the master/parent template is changed.

INDUCTIVE UNIVERSITY

About Templates

Watch the Video

To create a Template

1. Right-click the **Templates** node in the **Project Browser** and choose **New Template** from the menu.
   Or you can choose **File > New > New Template** from the Designer's menu bar.

   A checkered box is displayed in the Designer's workspace.

2. Configure the master template the same way you would design windows: by adding components to it, and configuring those components using property bindings and scripting.

   See Basic Template for detailed information on creating a template.

Using the Template

Once you have a template, here are some of the things you can do with it.

Renaming the Template

You can rename your template by clicking on its name in the Project Browser and pressing F2, or changing the name in the Property Editor.

Creating the Template Instances

Once you've made your template, you can use it on any of the windows in your project by doing any one of the following steps:

- You can drag the template from the Project Browser into an open window just like you can drag components into the window for display.
- You can right-click on the template and choose **Use In Window**, which will let you place the template inside a window with another click.
- You can drag a tag from the Tag Browser to a window and from the pop-up menu, which is displayed, you can choose a template.

The template instance can then be moved and resized like any other component.

Binding Template Properties

Templates have the capability to incorporate custom properties. In this way, they are the same as any other Ignition components. The main difference between the custom properties for an Ignition component and a template is that the template has internal properties and template parameters.

- Internal Properties
  Internal properties help facilitate the bindings within a template in the same way that a window's root container will help facilitate bindings between components that make up the template. When a template is deployed onto a window the internal custom properties are not exposed to the world outside the template.

- Template Parameters
  The template parameters are the template's custom properties that are exposed to the outside world. In other words, when a template is deployed onto the root container of a window, the template parameters are available for binding with the objects on that window or to tags.

Editing a Template

You can open a template for editing by double-clicking on it in the Project Browser, or by double-clicking any instance of that template within a window. You design your template the same way you design windows: by adding components to it, and configuring those components using property bindings and scripting.

There are a few differences between templates and windows from an editing perspective. Templates, unlike windows, have a transparent background by default. This can be changed simply by editing the background color of the template. Templates also do not have the concept of the "Root Container" - the template itself acts like a container.

Once you change the master template, all the instances of that template are updated.

Next...

- Basic Template

Basic Template

Here we create a basic template. After the template is configured, you can create multiple instances of it throughout your project.

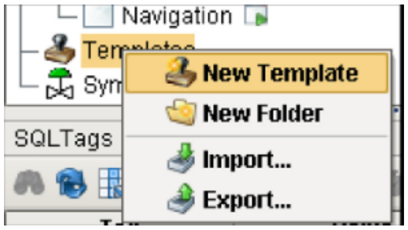
INDUCTIVE UNIVERSITY

Basic Template

Watch the Video

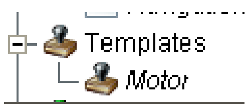
To create a Basic Template

1. In the Project Browser, right-click **Templates** and select **New Template**.

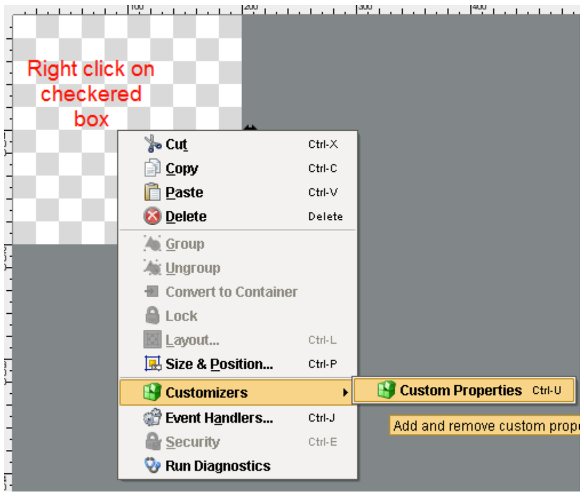


   A checkered box is displayed in the design space where you design your template. The checkered box means that the template is completely transparent. You can set a background if you want.

2. Right-click on ***New Template*** and click **Rename** to change its name to something else, lets say **Motor**.



3. Right-click on the checkered-box and select **Customizers > Custom Properties**.



   The **Custom Properties** window is displayed. Here you can set both the **Template Parameters** and **Internal Properties**. Template parameters appear on each template instance, allowing each instance to be configured differently, this is some sort of an indirection variable. Internal properties show up when designing the template master, but they do not show up on the template instances.

   Each template needs at least one Template Parameter property so that it can exist on every instance you put on the screens. Then each instance can bind the parameter property to a tag.
4. In the **Custom Properties** window, click the **+** icon in the Template Parameter area to add the values.

5. Enter the following and click **OK**.
   Name: **value**
   Type: **Integer**
   Drop Target: **select the check box**

6. Go to the checkered area of your template, from the Symbol Factory and Component Palette drag one or a few components such as a motor, a label, and a Level Indicator.

7. Resize the checkered area to fit all your components, and not larger, to remove all the unusable space.

   Now we need to link one of the components, lets say the Level Indicator, to the Template Parameter **value** we created earlier.

8. Select the Level Indicator and go to the Property Editor, click on the binding icon for the **Value** property.
   The Property Binding window is displayed.

9. Click the **Property** button, choose the **value** property, and click **OK**.

You now have configured a motor template, with a parameter property called **value** with an **integer** data type. Use this template to create many instances of it in your project.

Next...

- Custom Input Template

Custom Input Template

Sometimes you may need to have text fields in your project for the user to input data. Rather than copying and pasting these text fields into each window, you can create a template for them that includes a label and a text field. A template can contain scripting to validate the data in the text field.

Custom Input Template

Watch the Video

To create a custom input template

1. In the Project Browser, right-click **Templates** and select **New Template**.
2. Right-click on ***New Template*** and click **Rename** to change its name it to something, lets say **Tex t Entry**.
3. To add a parameter as you always do when creating a template, right-click the checkered area and select **Customizers > Custom Properties**.
4. In the **Custom Properties** window, add two **Template Parameters** by clicking the **+** icon twice and entering the following:
   1st Parameter
     Name: **display**
     Type: **String**
   2nd Parameter
     Name: **text**
     Type: **String**
5. Click **OK**.
6. From the Component Palette, drag a **Text Field** and a **Label** to the checkered area of the template.
7. Resize the size of the root container's area, the checkered area, so that the label and the text box occupy the majority of the space.
8. Select the **Label** component, go to **Property Editor**, and click on the binding icon of the **Text** property.
   The Property Binding window is displayed.
9. In the **Property Binding** window, click on **Property**, choose the **display** property.
10. Make sure the **Bidirectional** Box is not selected, and click **OK.**
    Don't worry if the label on the template disappears. Its is simply displaying the value of the display custom property which is currently blank.
11. Now select the **Text Field** component, go to **Property Editor**, and click on the binding icon of the **Text** property.
12. In the **Property Binding** window, click on **Property**, choose the **text** property, and click **OK**.
13. Make sure the **Bidirectional** Box is selected.

    Suppose you want data validation on the Text Field. We can have the background color of the Text Field change depending on whether the user has entered a text value or not. We can set the field to red when its empty, and white when it has a value.
14. While the **Text Field** component is still selected, go to **Property Editor**, and click on the binding icon of the **Background** property.
15. In the **Property Binding** window, click on **Expression**, enter the following expression:

```
//Inside the if statement is the len and the trim functions that are
availible when doing expression bindings.
//The trim function will trim the blank spaces from the text
therefore validating that there is actual text rather than spaces.
//The len function will count the lenth of the recently trimmed text.
//The if statements asks the question: Is the length of the trimmed
text greater than zero?
if(len(trim({textBox.Text Field.text}))>0,color(255,255,255),color
(255,0,0))
```

16. Click **OK**.

17. Close the template, click **Save** on the prompt window, and click **OK**.

18. To set data validation, drag the **Text Entry** template from **Project Browser** to the screen /window.
Notice that the text field is red.

19. In the **Property Editor**, go to the text custom property and enter your name.
Notice your name in the Text Field and that the background color changed to white.

Next...

- Template - Resizing and Enable Layout

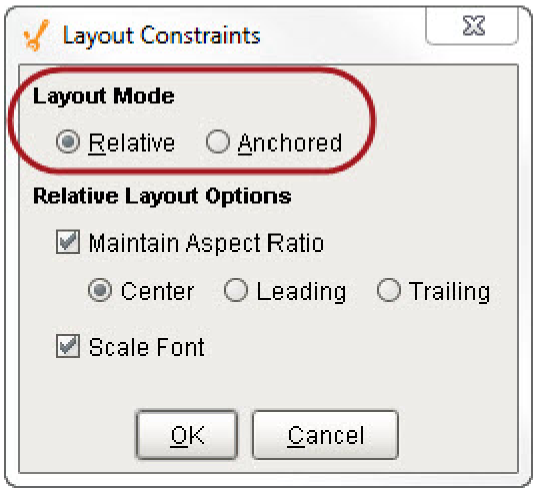
Template - Resizing and Enable Layout

You can configure the layout of each template so it resizes properly.

By default, when you drag a template into the window from the Project Browser, the size of the instance is exactly the same size as the master template. You can make the size larger or smaller. To go back to the same size as the master template, right-click on the instance and choose **Revert to Master size**.

For every component or template instance you add to the window, you can pick a Layout option as to how it is going to resize in the Client. Right-click on the instance, select **Layout** from the menu, the Layout Constraints window displays showing all the default settings.

To learn more about layout, **see the Component Layout page**.


Layout Constraints

Layout Mode

[✓ Relative] Anchored

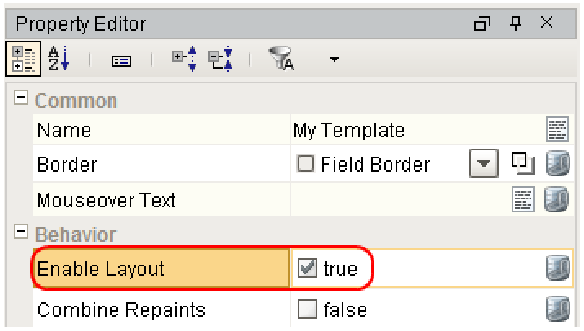
Relative Layout Options

[✓] Maintain Aspect Ratio

[✓ Center] Leading Trailing

[✓] Scale Font

If your template instance resizes in the client, then it will stretch all of the components inside it in the same way a Component Group works. That is: it will ignore any layout settings and stretch without maintaining aspect ratio. If you want the template instances to respect your layout settings, set the **Enabl e Layout** property to true in the template definition.


Property Editor

Common	
Name	My Template
Border	Field Border
Mouseover Text	
Behavior	
Enable Layout	[✓] true
Combine Repaints	[ ] false

Template - Resizing and Enable Layout

Watch the Video

Next...

- Template - Indirect Binding

Template - Indirect Binding

You can create templates that point indirectly to a set of tags based on a simple parameter. This is very helpful when you have a large number of UDTs with the same type of tags that only differ in one parameter. For example, lets say you have 100 Motor UDTs that all have the same kind and number of tags. The only thing that is different is the Motor number.

If each tag inside the UDT has a tag path that looks like this:

```
Motors/Motor 1/AMPS
```

You can create a Template Paramenter property on the Custom Properties window and make it to point to one of the tags indirectly, for example, you can call it `motorNumber`.

For every component within the template the tag path can be bound with indirection in such a way that the Custom property of `MotorNumber` substitutes the motor number therefore resulting in a valid tag path that points to the desired motor.

To learn about indirect tag binding, see Indirect Tag Binding.

INDUCTIVE UNIVERSIT

Template - Indirect Binding

Watch the Video

Next...

- Template - UDT Parameter

Template - UDT Parameter

User Defined Types (UDT) parameters are properties that you can bind the template components to. Since the UDT parameter is a Custom property that encapsulates the UDT tags and some meta properties you can access all this information through a custom property.

To add a component to the template and bind it to parts of UDT

1. Start with a blank template.
2. Add components to the template.
3. Create a **Custom** property on the template's Root Container, and make the Custom property's **Type** a **User Defined** type instead of a **Standard** data type.
4. Click **OK**.
5. Select a component on the template, in the component's **Property Editor**, click the binding icon next to the property that you want to influence.
6. Select the **Property** binding and choose the tag from the UDT that you would like the binding to occur.
7. Click **OK**.

Template – UDT Parameter

Watch the Video

Next...

- Template Composition

Template Composition

You can embed templates inside of other templates. The nested template behaves like a component.

To nest a template inside another template

1. Drag an existing template from the **Project Browser** into the template that you are currently working on.
2. Bind the nest template properties to the properties located on the parent template's **Custom** properties such as UDT parameters.

Template Composition

Watch the Video

Next...

Template Features

Some of the advanced template features in the Designer are: changing Template Path using Template Repeater and Cloud Templates.

In this section ...

Changing Template Path

An instance of a template on a window has a property called **Template Path**. You can change this property on a window dynamically, and it can be bound to anything that produces a valid template path.

For example, if there are two tank templates in a folder called Tanks, one template is called Tank A and the other is called Tank B. Each tank has a different look, but they have the same Custom properties. Their respective template paths are Tanks/Tank A and Tanks/Tank B. The template rendered on a window can swap between Tank A and Tank B by binding the template's property to any string reference that says Tanks/Tank A or Tanks/Tank B.

INDUCTIVE UNIVERSITY

Changing Template Path

Watch the Video

Next ...

- Template Repeater Component

Template Repeater Component

The **Template Repeater** component lets you easily create multiple instances of a master template for display on the HMI. Each instance shown in the Template Repeater has the same look, feel, and functionality of the master template. The instances can be arranged vertically, horizontally, or in a "flow" layout, which can either be top-to-bottom or left-to-right. If there are too many instances to fit, a scrollbar is added to the display.

Additional information on the Template Repeater can be found on the Template Repeater page in the appendix.

INDUCTIVE UNIVERSITY

Template Repeater

Watch the Video

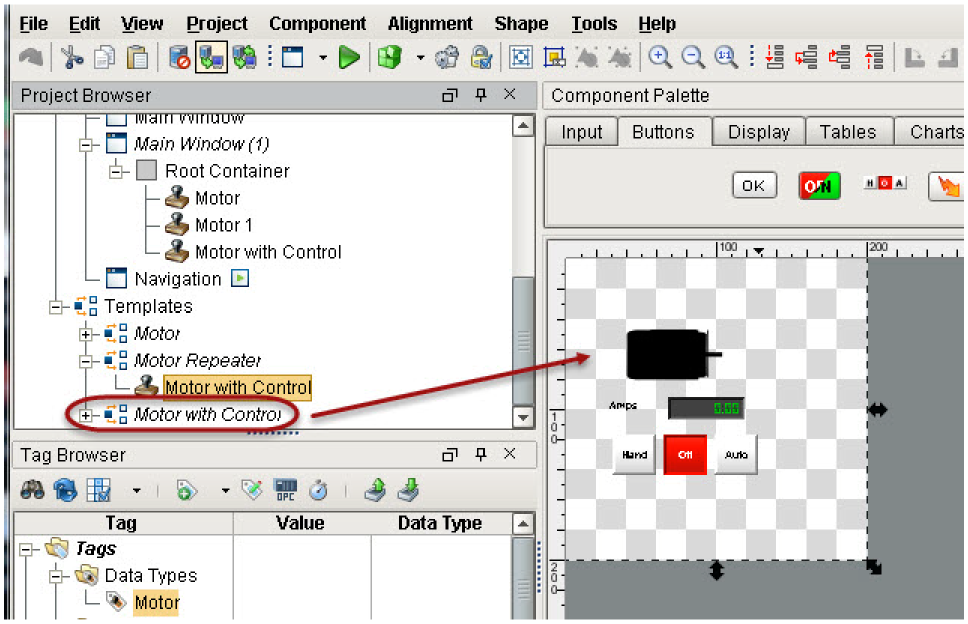
To customize a template for use in a Template Repeater

By default, the Template Repeater uses integers to manage the templates it displays. The template intended for use inside the Template Repeater must therefore be configured so that it can accept these integers as pass-through parameters from the Template Repeater.

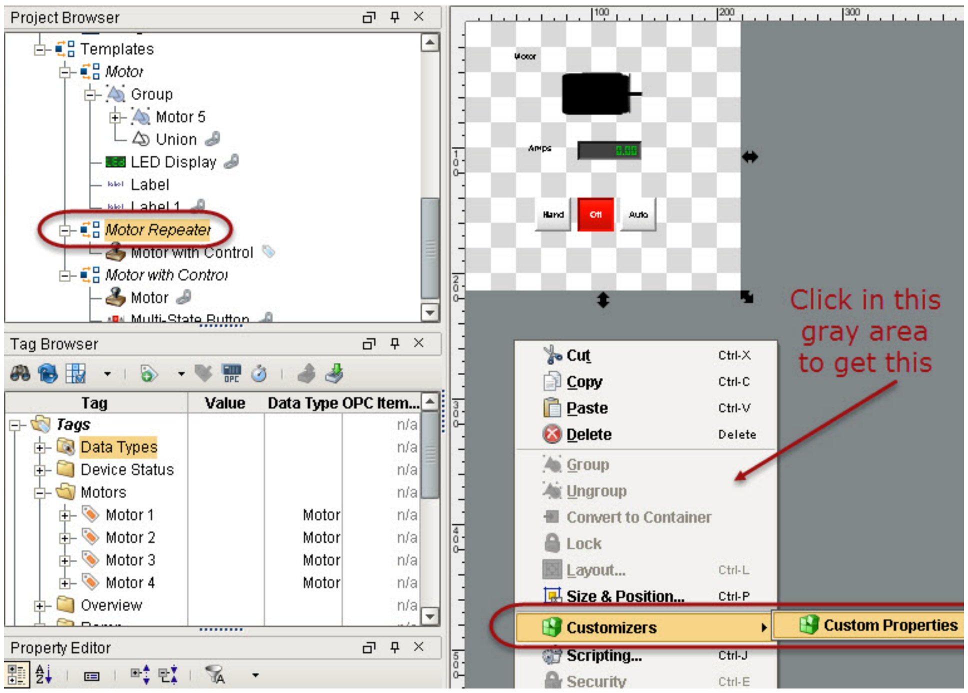
Let's customize a template so we can use it with the Template Repeater.

1. In **Project Browser**, right-click **Templates**, select **New Template** to create a new template.
2. Right-click on the New Template, select **Rename** and change the name to **Motor Repeater**.

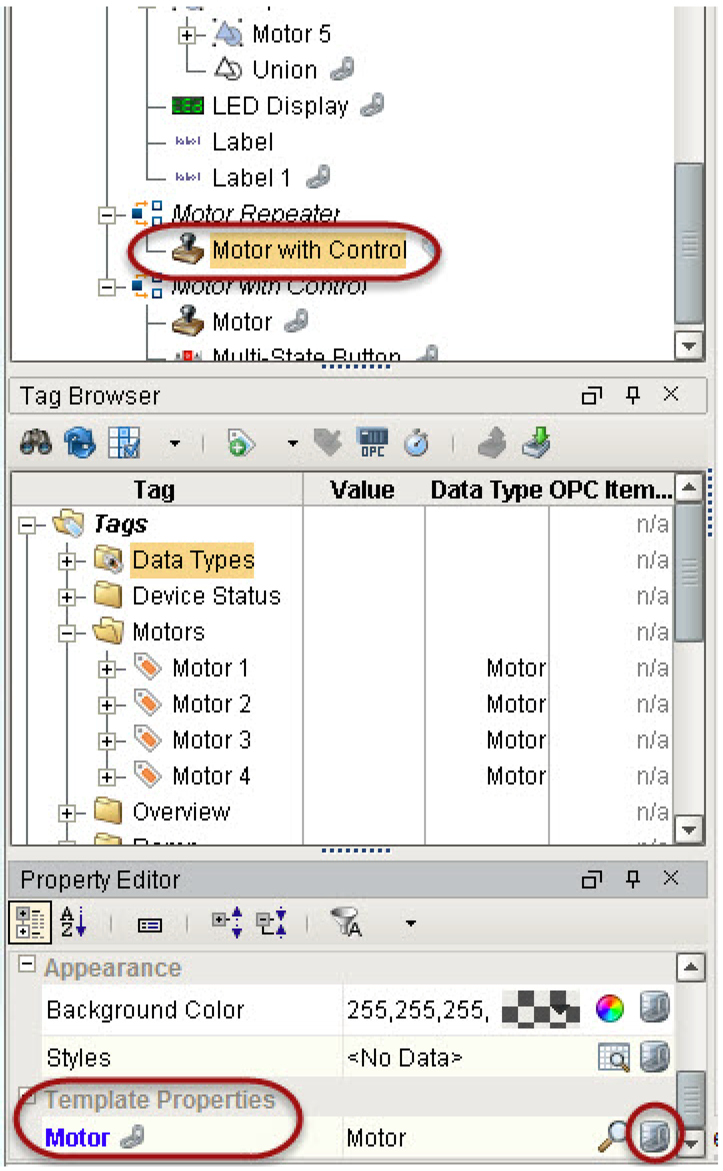
   We have previously created a **Motor with Control** template that includes a Motor (from the Symbol Factory), an LED Display, and a Multi-State Button.
3. Drag a **Motor with Control** template inside the new **Motor Repeater** template.



4. Select the **Motor Repeater**, right-click on the gray area of Motor Repeater, select **Customizer > Custom Properties** to create a new custom parameter.



5. In the Custom Properties window, click the **green +** icon for **Template Parameters** and type:
   Name: **MotorNumber**
   Type: **Integer**
   MotorNumber is the custom property that is required by the Template Repeater component to create and manage all the template instances.
6. Click **OK**.
7. Select the nested **Motor with Control** template.
8. In the Property Editor, under **Template Properties**, select the binding icon for the **Motor** UDT.



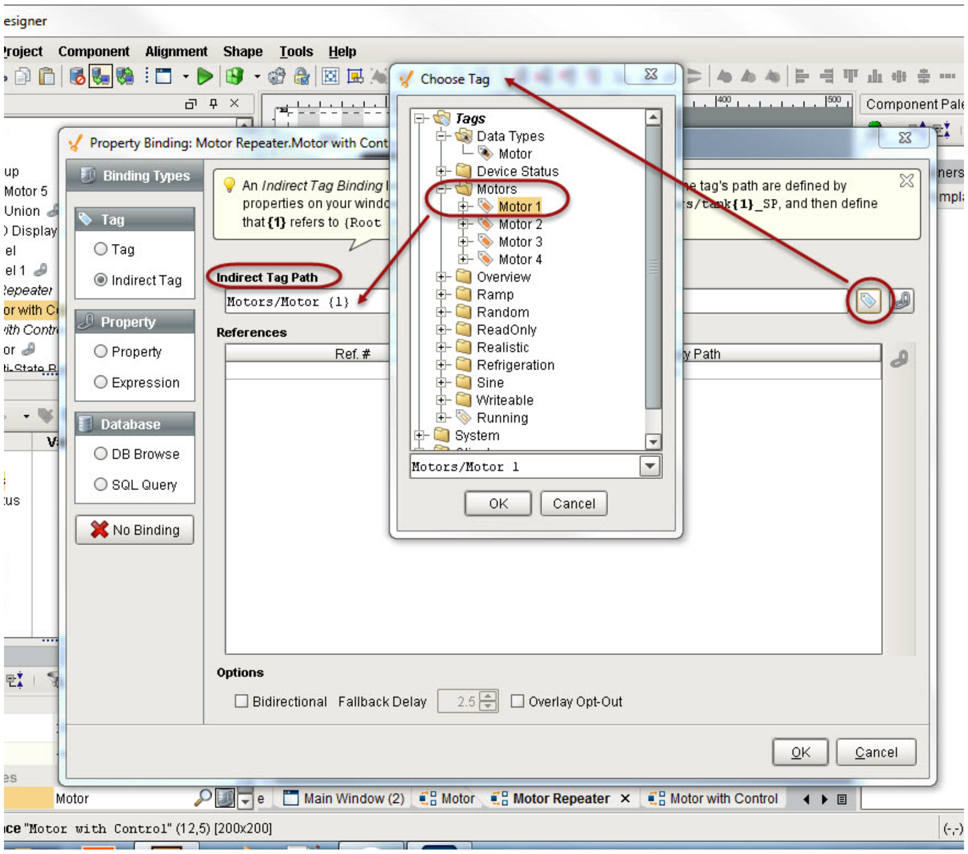
Tag	Value	Data Type	OPC Item...
Tags			n/a
Data Types			n/a
Device Status			n/a
Motors			n/a
Motor 1		Motor	n/a
Motor 2		Motor	n/a
Motor 3		Motor	n/a
Motor 4		Motor	n/a
Overview			n/a

Background Color	255,255,255,
Styles	<No Data>
Motor	Motor

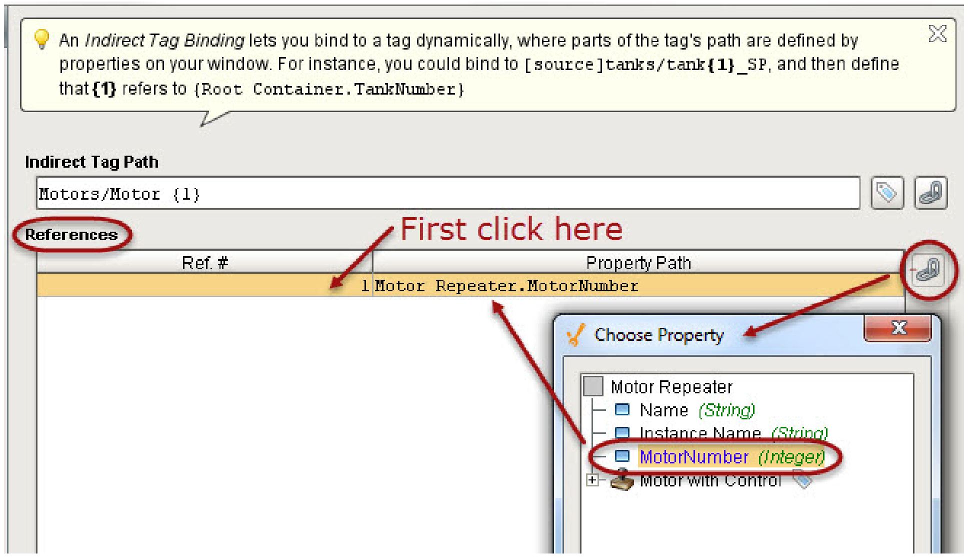
9. In the Property Binding window, click on the tag icon for the **Indirect Tag Path**, select **Motor/Motor 1**, and click **OK**.

10. Change **Motor/Motor 1** to **Motor/Motor {1}**.

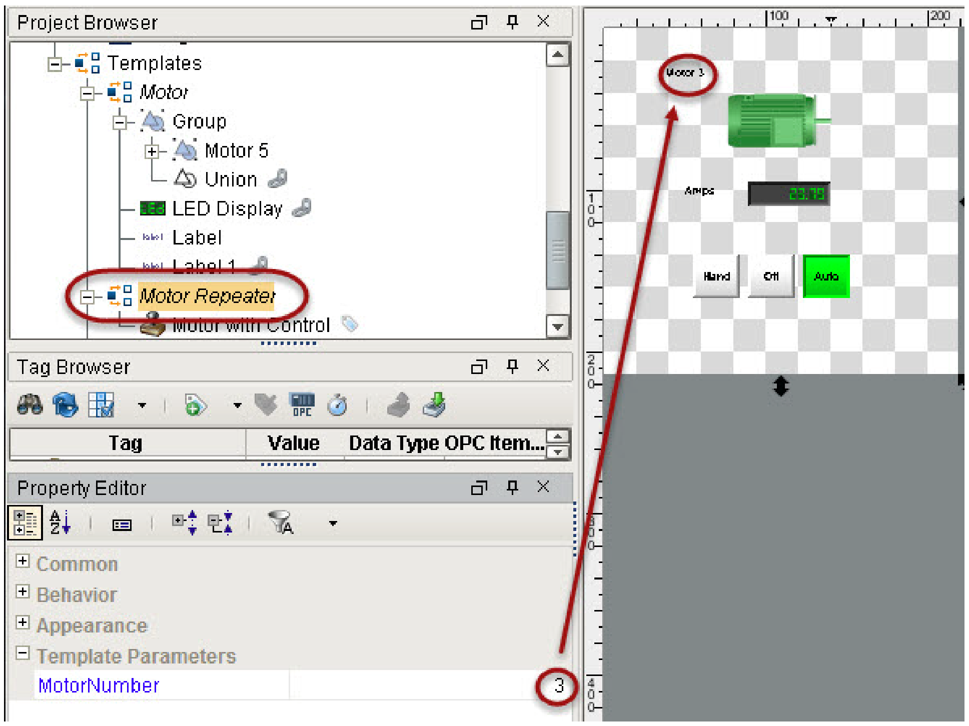
This indirectly binds the tag path to the address of the motor UDT with the indirection occurring on the number portion of the address.



11. In the **References** section, click in the area under Ref.# and then click the binding icon, select the **MotorNumber** property, and click **OK**. Now, when we pass a number into the Motor Repeater, the number will populate the nested template's information.


An *Indirect Tag Binding* lets you bind to a tag dynamically, where parts of the tag's path are defined by properties on your window. For instance, you could bind to `[source]tanks/tank{1}_SP`, and then define that **{1}** refers to `{Root Container.TankNumber}`

To test the new template, select the Motor Repeater template, change **MotorNumber** to a different number, and see the Motor number changes as well to the same number you specified.

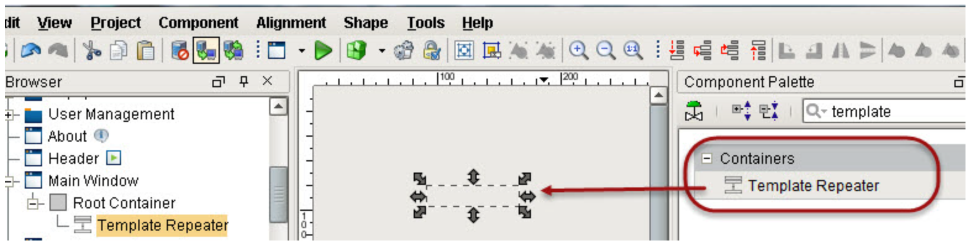


That's it! The Motor Repeater template is now configured to accept integers from the Template Repeater component.

To use a Template Repeater and fill it with a template

Let's use the Template Repeater component from the Component Palette and fill it with our newly created Motor Repeater template.

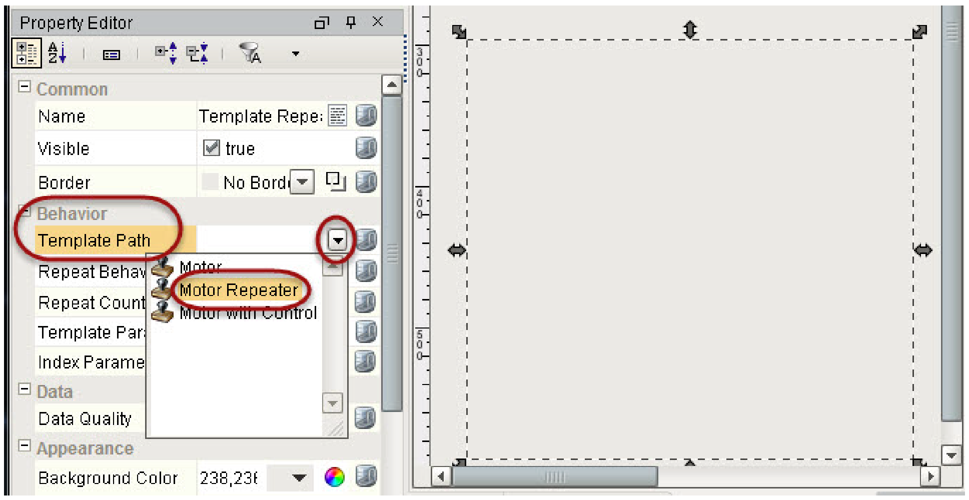
1. Go to a Main Window, and from the Component Palette, drag the Template Repeater component into the window.
   The Template Repeater displays as a small dashed box.



2. Resize it to at least twice the height and width of your Motor Repeater template.

3. While the Template Repeater is still selected, go to Property Editor.

4. Click the dropdown for **Template Path** and select the **Motor Repeater** template.
   The dropdown menu shows all three templates we created earlier.



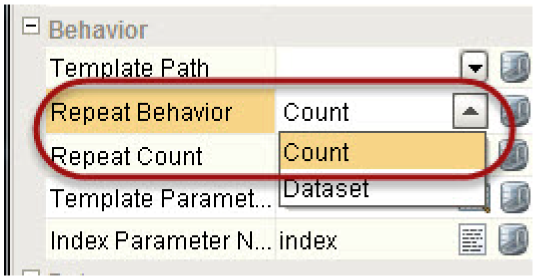
Below the **Template Path** property is the **Repeat Behavior** property which has a dropdown menu with two options, they are:

Count
Repeats the template a number of times, assigning each template an index number. The Count number starts at zero and increments upwards.

Dataset
Repeats the template once per row in the **Template Parameter** dataset.

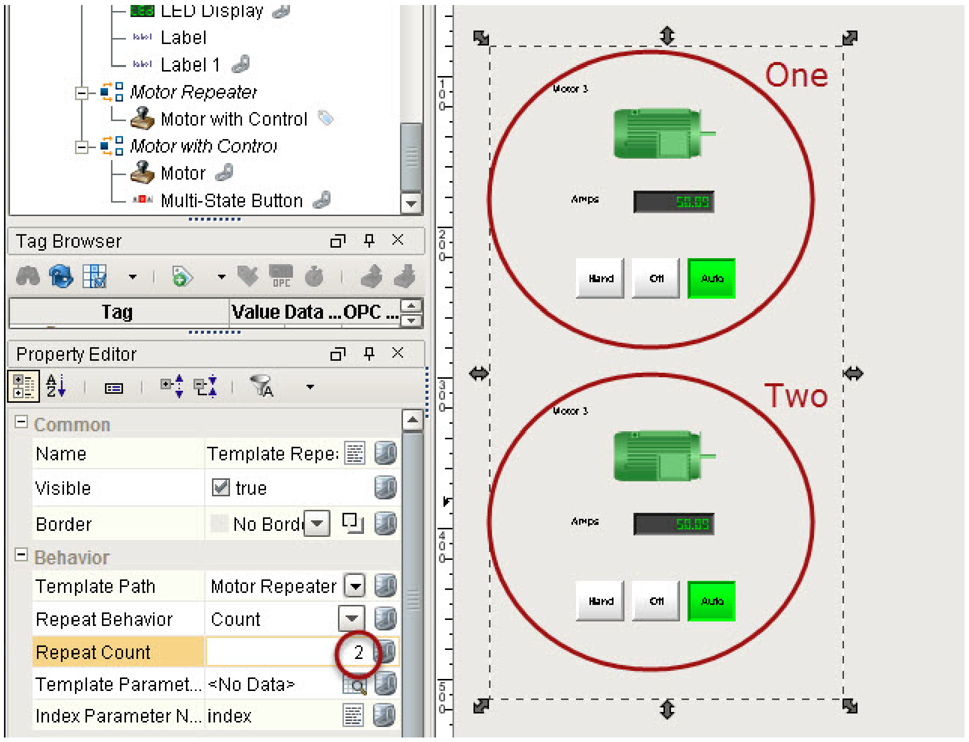
We will look at using both options. First, let's use the **Count** option.

5. Select **Count** from the dropdown menu of **Repeat Behavior**.



6. Set **Repeat Count** to the number of times you want the Template Repeater to repeat the template.

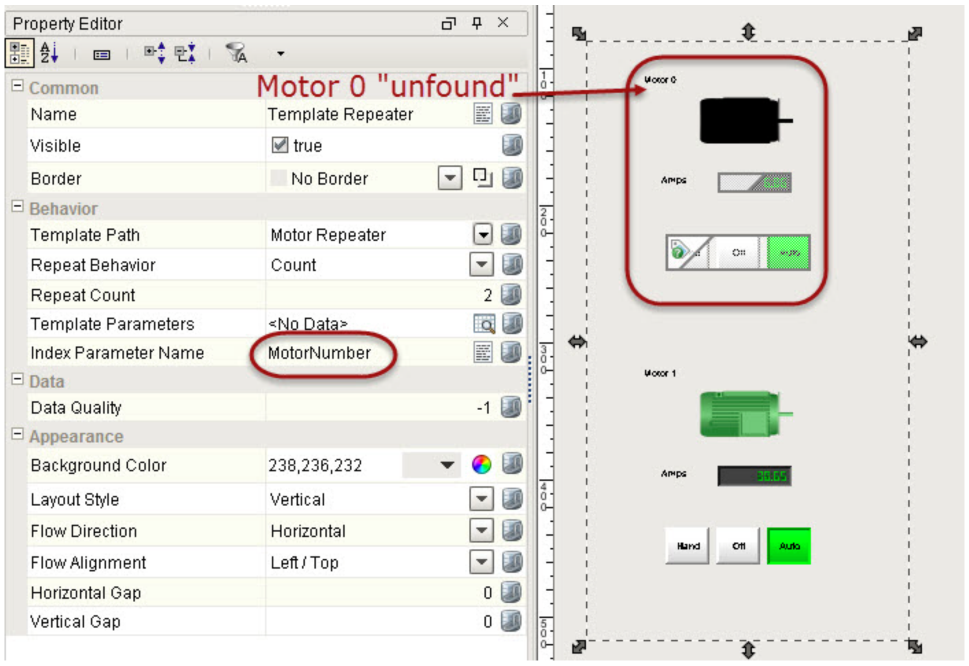
For example, here we set Repeat Count to 2, and so 2 templates are repeated in Template Repeater.



Common	
Name	Template Repe:
Visible	✓ true
Border	No Bord
Behavior	
Template Path	Motor Repeater
Repeat Behavior	Count
Repeat Count	2
Template Paramet...	<No Data>
Index Parameter N...	index

7. Change the **index** entry in **Index Parameter Name** to **MotoNumber** to apply the MotorNumber custom property to the Template Repeater. The Template Repeater now fills with the motors starting with Motor 0. As there is no Motor 0, the Template Repeater displays the motor as “unfound” and is displayed in black.

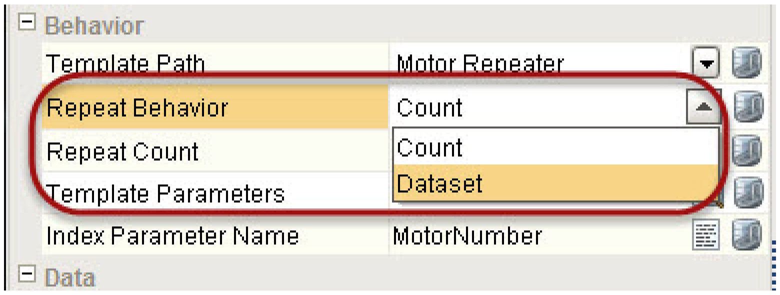
   Again, notice that the Count property begins its count at zero.



Common	
Name	Template Repeater
Visible	✓ true
Border	No Border
Behavior	
Template Path	Motor Repeater
Repeat Behavior	Count
Repeat Count	2
Template Parameters	<No Data>
Index Parameter Name	MotorNumber
Data	
Data Quality	-1
Appearance	
Background Color	238,236,232
Layout Style	Vertical
Flow Direction	Horizontal
Flow Alignment	Left / Top
Horizontal Gap	0
Vertical Gap	0

   Now lets use the Dataset option.

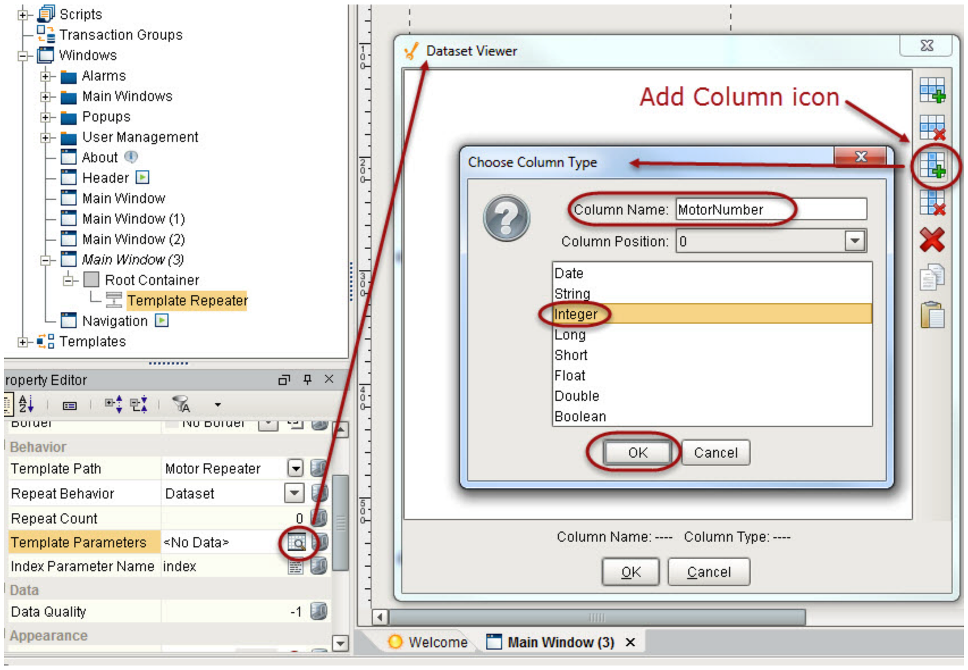
8. Select **Dataset** from the dropdown menu of the **Repeat Behavior**.
   This allows the Template Repeater to index through a dataset.



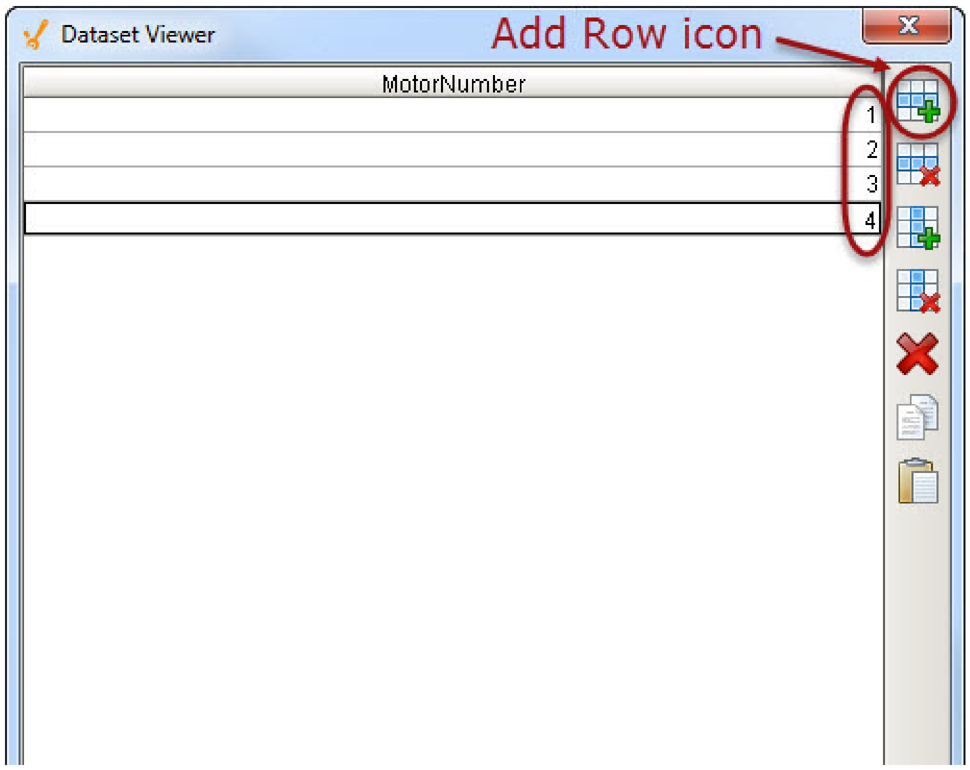
The Dataset option overcomes the “unfound” problem we had with the Count option and offers a lot of flexibility in how the motors are created and displayed in Template Repeater.

9. Click on the Dataset Viewer icon for **Template Parameters**.

10. In the Dataset Viewer window, click the Add Column icon, type **MotorNumber** as the Column Name and select Integer as the column type, then click **OK**.
Now we have a one column dataset with a header name that is the same as the Custom property on the Template Repeater.



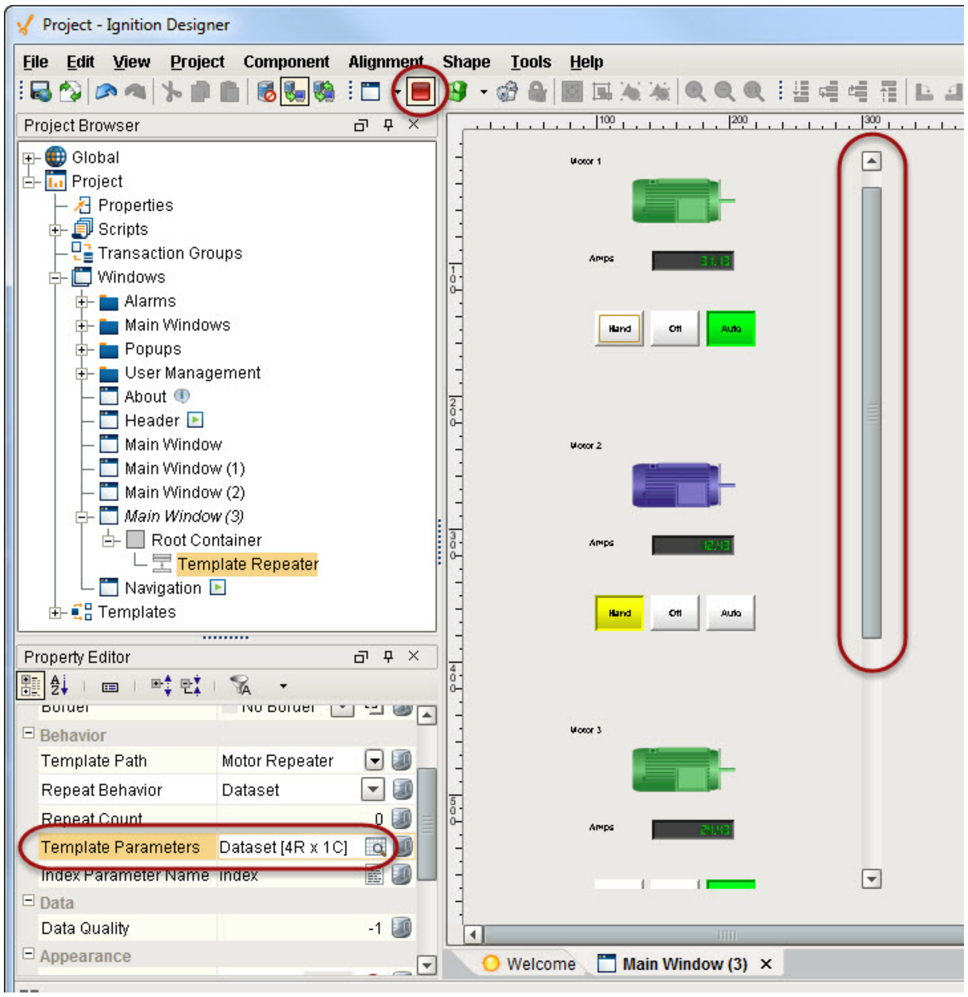
11. Click the Add Row icon four times to get four rows, and change the row numbers from 0 to 1-4 to display Motors 1-4, and click OK.



MotorNumber
1
2
3
4

The Template Repeater component now fills with the motors as you specified them. In this case, they are displayed as One column of 1-4.

12. Go to Preview Mode and notice that the Scroll bar allows for you to view all the Motors.



Behavior	
Template Path	Motor Repeater
Repeat Behavior	Dataset
Repeat Count	0
Template Parameters	Dataset [4R x 1C]
Index Parameter Name	index
Data	
Data Quality	-1
Appearance	

Next...

- Cloud Templates
- Template Repeater

Cloud Templates

Cloud Templates is a free online repository of templates hosted by Inductive Automation. You can share your templates publicly or store them privately as well as browse the library of templates to help get your project off the ground faster.

INDUCTIVE UNIVERSITY

Cloud Templates

Watch the Video

Importing Cloud Templates

Cloud Templates give the Ignition community access to pre-built templates and allows users to share their own templates. Using Cloud templates, an Ignition developer can store templates for private use or public access outside the local Gateway, and make the templates available from any Designer regardless of the project or Gateway.

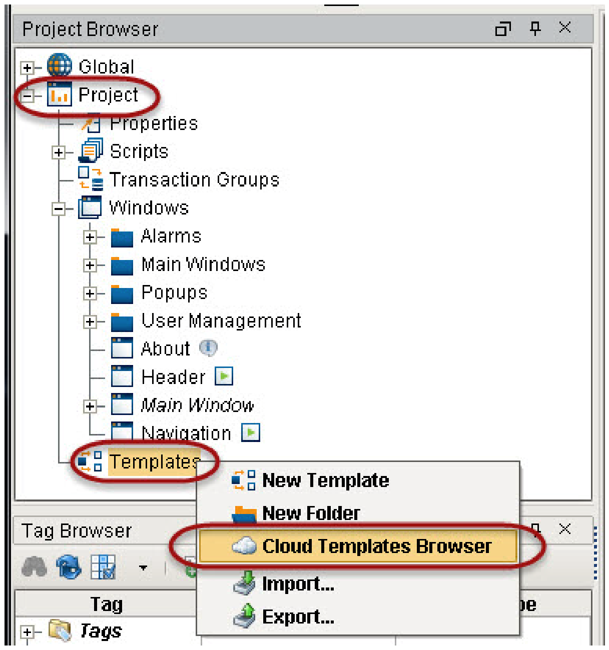
Because the templates are accessible from any Designer, there is no longer the need to manually port the templates from Gateway to Gateway.

The Cloud templates are entirely customizable after you import them from the Cloud. And unlike other Ignition templates, changes to the templates in the Cloud do not affect those template instances previously downloaded. Note that the public Cloud Templates are all free to use.

To import a Cloud template

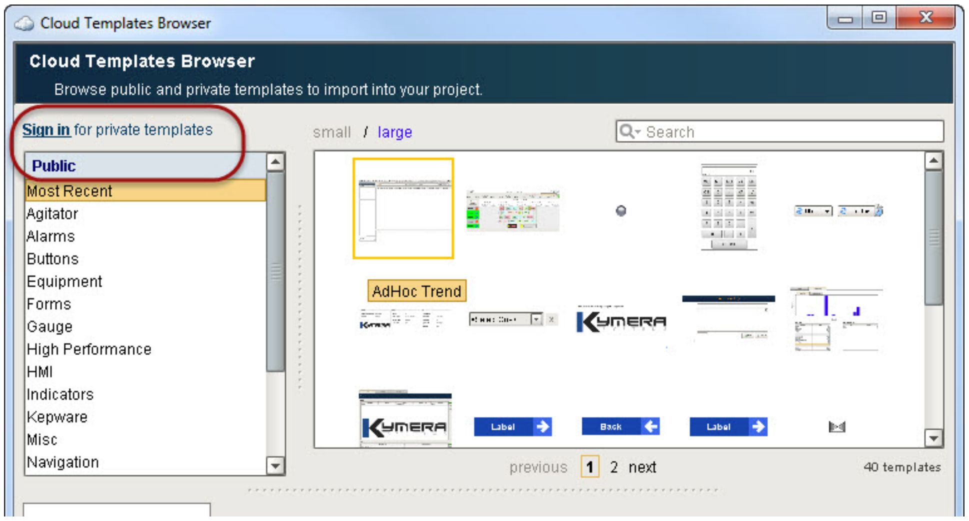
Let's import a template from the Cloud.

1. From the **Project Browser** expand **Project**.
2. Right-click on **Templates** and select **Cloud Templates Browser** from the popup menu.



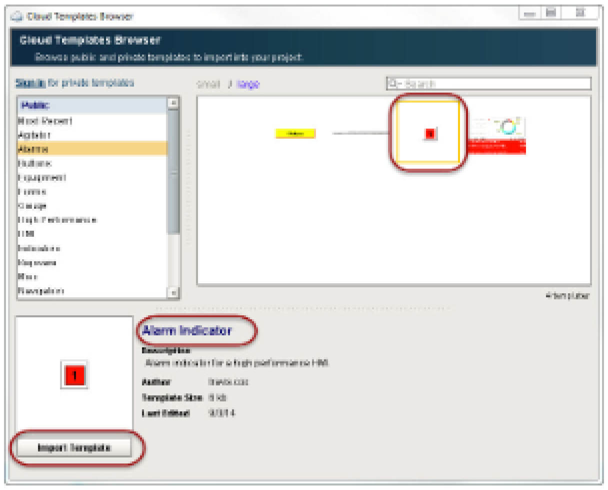
The **Cloud Templates Browser** is displayed.

The list of categories of Cloud templates for download are located on the left of the browser. This list is divided into **Private** and **Public** templates. The **Public** templates are displayed by default, but to see your **Private** templates, you must sign in with your Inductive Automation account.



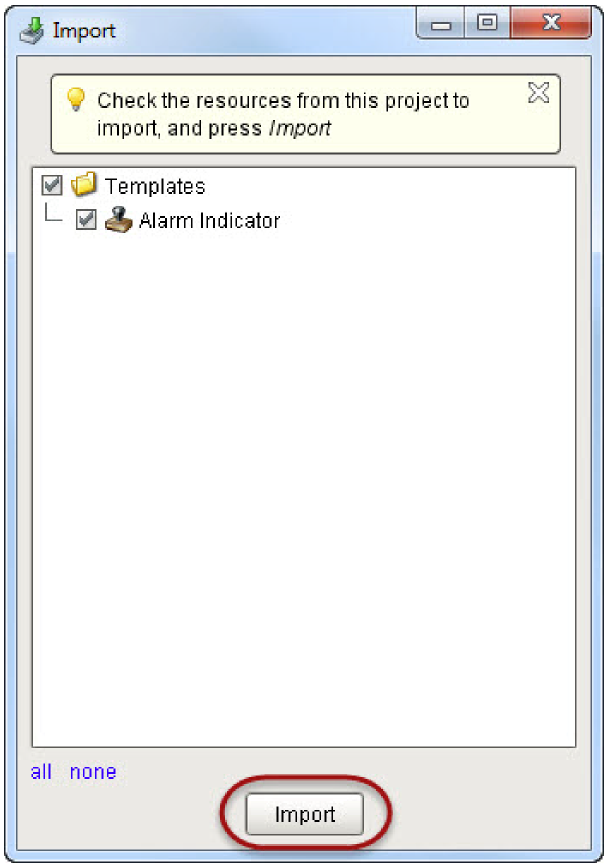
3. Select **Alarms** from the **Public** list.
   All publicly available templates for the Alarms category are displayed along with a description for the selected template.

4. Select the **Alarm Indicator** template and click **Import Template**.

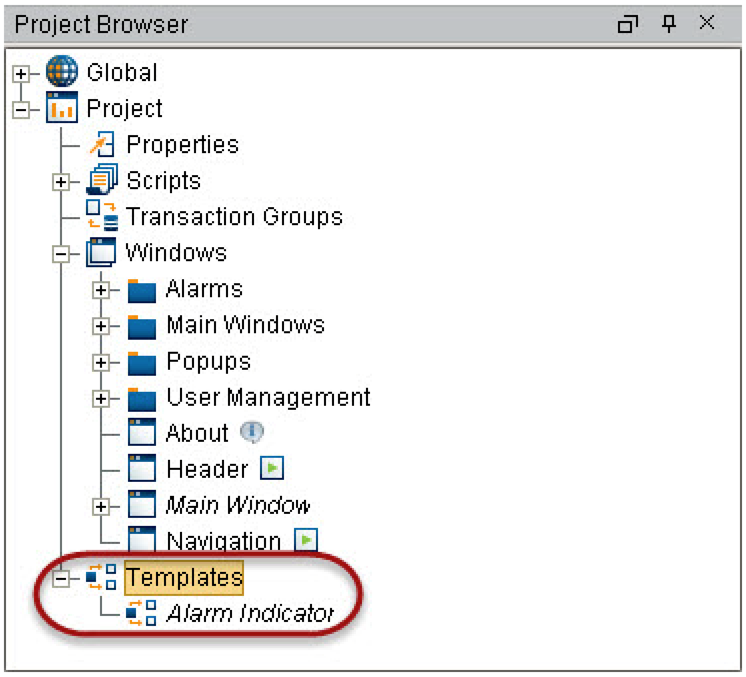


A new popup appears allowing you to confirm import on the template.

5. Click **Import** to confirm importing the template.


Import
Check the resources from this project to import, and press *Import*
Templates
Alarm Indicator
all
none
Import

The template is now imported and available for use within the **Templates** section of your **Project Browser**. You can drag your new template on a window or modify it however you like!



Next...

SQL in Ignition

For information about databases and how they are used in Ignition, see the Database Connections section.

What is SQL?

SQL or Structured Query Language is a standard computer language for accessing and manipulating database systems. SQL statements are used to create, maintain, and query relational databases.

Examples

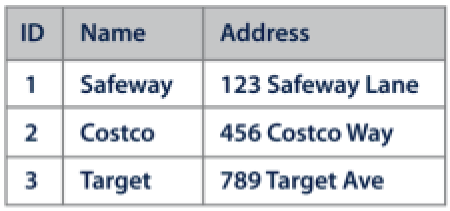
```
SELECT Firstname FROM Contacts WHERE Lastname = 'Smith'
INSERT INTO Contacts VALUES ('Joe', 'Smith')
UPDATE Contacts SET Lastname = 'Howard' WHERE ID = 1
DELETE FROM Contacts WHERE Lastname = 'Smith'
```

Tables

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 columns and rows.

The database table columns have their own unique names and have a 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

Primary Key

A primary key is a way to uniquely identify each row in a table. Every table must have a primary key. A primary key comprises of a single column or set of columns.
No two distinct rows in a table can have the same value (or combination of values) in those columns.

The Primary Key can be:

- Auto-incrementing
- Statically defined

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 each set of columns you specify 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.

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.

Examples

```
Supplier ( SupplierNumber, Name, Address, Type )
Invoices ( InvoiceNumber, SupplierNumber, Text )
```

NULL Value

NULL is a special marker used in SQL to indicate that a data value does not exist in the database.

SELECT Command

The SELECT statement is used to select data from a database. The result is stored in a result table, called the result-set.

Examples

```
SELECT * FROM Customers
SELECT Name FROM Customers
SELECT Name, Address FROM Customers
```

UPDATE Command

The UPDATE statement is used to update records in a table.

Examples

```
UPDATE Customers SET Name = 'Inductive Automation'
UPDATE Customers SET Address = '2110 21st Street' WHERE ID = 1
```

INSERT Command

The INSERT statement is used to insert a new row in a table.

Examples

```
INSERT INTO Customers (Name, Address, City, State, Zip, Country, Phone)
VALUES ('Inductive Automation', '2110 21st Street', 'Sacramento', 'CA',
'95818', NULL, '800-266-7798')
INSERT INTO Customers SELECT * FROM OtherCustomers
```

DELETE Command

The DELETE statement is used to delete records in a table.

Examples

```
DELETE FROM Customers
DELETE FROM Customers WHERE Name = 'Inductive Automation'
```

WHERE Clause

The WHERE clause is used to extract only those records that fulfill a specified criterion. Operators Allowed in the WHERE Clause are as follows:

= Equal
<> Not equal
> Greater than
< Less than
>= Greater than or equal
<= Less than or equal
BETWEEN Between an inclusive range
LIKE Search for a pattern
IN If you know the exact value you want to return for at least one of the columns

The AND & OR operators are used to filter records based on more than one condition.

Examples

```
SELECT * FROM Customers WHERE State = 'CA'
SELECT Name FROM Customers WHERE Address LIKE '%St%' AND State = 'CA'
```

ORDER BY Clause

The ORDER BY keyword is used to sort the result-set by a specified column. The ORDER BY keyword sort the records in ascending order by default. If you want to sort the records in a descending order, you can use the DESC keyword.

Examples

```
SELECT * FROM Customers ORDER BY Name ASC
SELECT * FROM Customers ORDER BY State ASC, Name DESC
```

Joins

The JOIN keyword is used in an SQL statement to query data from two or more tables, based on a relationship between certain columns in these tables. Tables in a database are often related to each other with keys. A primary key is a column (or a combination of columns) with a unique value for each row. Each primary key value must be unique within the table. The purpose is to bind data together, across tables, without repeating all of the data in every table.

JOIN: Return rows when there is at least one match in both tables
LEFT JOIN: Return all rows from the left table, even if there are no matches in the right table
RIGHT JOIN: Return all rows from the right table, even if there are no matches in the left table
FULL JOIN: Return rows when there is a match in one of the tables

Example

```
SELECT * FROM Contacts ct JOIN Customers cn ON cn.ID = ct.CustomerID
```

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.

Example

```
SELECT SUM(Duration) FROM Downtime GROUP BY CauseID
```

In this section ...

Querying, Adding, and Editing Data

Ignition offers a number of built-in scripting functions for you to use when running queries against the database. Depending on the type of your query and the sort of results you want, you will use a different function. The following functions are the ones you will use most:

Scripting Function	Description
system.db.runPrepQuery	Use this function to run basic SELECT queries against your database using a special placeholder (?) to allow for dynamic query building.
system.db.runPrepUpdate	Use this function to run queries that change the data in the database. This function also makes use of the (?) placeholder.
system.db.runScalarQuery	Use this function when you want only the value from the first column of the first row of your query to return.
system.db.createSProcCall and system.db.execSProcCall	Use these functions together to execute any stored procedures you may have in your database.

Note the options provided by each of the different functions. For example, the options that return auto-generated key for insert queries can be extremely helpful and eliminate the need to hit the database multiple times.

You can find examples of each of these functions in their linked topic sections.

In this section ...

Querying Data from Database

Use binding to bring back data from your database. By using the Polling Mode of the property binding, you can specify how often or at what rate you want to bring back the data.

To display data from the database on a table component

If a database is connected to your ignition server, you can display data from that database on a table component.

1. In the Designer, from the Component Palette, drag a **Table** or **Power Table** component to your window.
   While the component is selected, you can see all the properties that are related to the component in the Property Editor.

2. In the **Property Editor**, click the binding icon of the **Data** property.
   The Property Binding window is displayed.

3. Select the **DB Browse** database option.

4. Under the **Browse Database** box, find the table from the list of database connections and expand the table to inspect the table's columns.

5. By default, the query is built with a **`where`** clause. If you want to return all the data from the table, deselect the primary key option associated with the table's primary key by selecting the column and clicking the **Key** icon. To access the key column, select the column that is bold and then deselect the Key icon.

6. To choose which columns are returned, select individual columns from the column selector, or hold the shift button and select multiple columns.

7. Select the **Polling Mode** to specify how the Client will refresh the table data with new information from the database.
   When Polling Mode is set to OFF, this ensures the SQL query on a SQL binding only runs when the window is opened or a referenced property changes value.

As you build a query with this method, the resulting SQL statement is being built in the bottom part of the property binding management screen.

Querying Data from Database

Watch the Video

Next...

- Filter Data in Table

Filter Data in Table

Filtering table data by using a drop-down box is possible, if the **Table** component's **Data** property is bound to a SQL query, thus dynamically modifying the WHERE clause of a query, or creating a whole custom WHERE clause. To implement this solution, you must have a **Table** component and a **Dropdown List** component on a window.

Many SQL queries have WHERE clauses that filter the result returned from the database. When a SQL query examines the contents of a table it evaluates each row against the WHERE clause. If the WHERE clause is true, the row is included in the data-set, otherwise, it is not included in the data-set.

With SQL queries you can use the WHERE clause to your advantage when building interactive screens. For example, the following SQL query will filter the result set in such a way that the data-set returns only tanks from Area A.

```
SELECT * FROM tanks WHERE area = "Area A"
```

While the following query will return everything, but still will evaluate the query's WHERE clause.

```
SELECT * FROM tanks WHERE 1=1
```

In this case 1=1 will always evaluate as true, therefore the query will return every row.

Imagine if you had a drop-down component near the table, and the drop-down component's **Data** property was populated with the following data:

Value	Label
0	Area A
1	Area B
2	All

This data can help to create the WHERE clause. The first step is to create a **Custom** property on the drop-down list.

1. Right-click on the drop-down component and select **Customizers > Custom Properties**.
2. Create a new property and name it **WhereClause**, and it must be a **String** data type.
3. Click **OK**.
4. Select the binding icon for the new **Custom** property and select **Expression** binding.
5. The expression binding should look something like this:

```
if(
{Root Container.Dropdown.selectedStringValue}="All", //Evaluate this and
return a true or a false.
"1=1",  //Return 1=1 if the statement was true.
"Area = '"+{Root Container.Dropdown.selectedStringValue}+"'")
//Otherwise return a different where clause.
```

When you select a different option from the drop-down component the binding gets re-evaluated on the table's data binding resulting in the query executing with the new WHERE clause.

You may want to turn Polling Mode off on the tables Data property binding in order to limit the periodic querying of the database.

Next...

- Add Data to Database

Filter Data in Table

Watch the Video

Add Data to Database

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 to go into the database and then execute a SQL `INSERT` statement.

Suppose you had a window that had three components and a button. A script executed on the button's **actionPerformed** event handler would collect the relevant properties and insert them into a database. Here is some example code:

```
firstName = event.source.parent.getComponent('FirstName').text
lastName = event.source.parent.getComponent('LastName').text
age = event.source.parent.getComponent('Age').intValue

query = "INSERT INTO people (FirstName, LastName, Age) VALUES (?,?,?)"
args = [firstName, lastName, age]
system.db.runPrepUpdate(query, args)
```

This code assumes that there is a table that exist with the correct column names. Also, it probably has an auto-incrementing ID column as many tables do.

Add Data to Database

Watch the Video

Next...

- Edit Data in Database

Edit Data in Database

A table component 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. 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

Edit Data in Database

Watch the Video

The following is an example of using the **Table** component as opposed to the **Power Table** component.

1. Start with a window that has a **Table** on it.
2. Bind the Table's **Data** property to a SQL table.
   The data will appear in the table.
3. Right-click on the **Table** and select **Customizers > Table Customizer**.
4. Select the **Editable** check box for each of the table's columns that the user will be editing.
5. Right-click the table and select **Scripting**.
6. Create a script in the table's **`cell > cellEdited`** event handler.
   Using the following script will allow your table to be editable from a Client. The script works with a table that looks like the table above. Your table may be different.

```
id = event.source.data.getValueAt(event.row, 'id')        #Get the
id of the database column.
headers = system.dataset.getColumnHeaders(event.source.data)   #Get
the headers
query = "UPDATE User SET %s = ? WHERE id = ?" % (headers[event.
column])
args = [event.newValue, id]
system.db.runPrepUpdate(query, args)
```

The following is an example of using the **Power Table** component.

1. Start with a window that has a **Table** on it.
2. Bind the Table's **Data** property to a SQL table.
   The data will appear in the table.
3. Right-click on the **Table** and select **Customizers > Table Customizer**.
4. Select the **Editable** check box for each of the table's columns that the user will be editing.
5. Create a script in the table's **`onCellEdited`** extension function by selecting the **`onCellEdited`** extension function. Enable the function.
   Using the following script will allow your table to be editable from a Client. The script works with a table that looks like the table above. Your table may be different.

```
id = self.data.getValueAt(rowIndex, 'id')        #Get the id of the
database column.
query = "UPDATE User SET %s = ? WHERE id = ?" % (colName)
args = [newValue, id]
system.db.runPrepUpdate(query, args)
system.db.refresh(self,"data")
```

Next...

- Refreshing SQL Data on Component

Refreshing SQL Data on Component

The SQL query that populates a data 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 most common way to do this is to have a button near the component that instructs the component's data property to refresh. Any property that is refreshed can be any property that has a binding to a SQL database.

Refreshing SQL Data on Component

Watch the Video

To update a component's property with scripting

1. Select the property on the component that you want to manually update.
2. Create the SQL binding on that component, and for **Polling Mode**, select **Off**.
3. Create a **Button** inside the same **Root Container** as the component that has the SQL binding.
4. Access the Button's scripting by double-clicking on the **Button** and selecting the **Script Editor** tab.
5. Using the system.db.refresh() function, enter your script that updates the **Data** property on the component with the SQL binding. This required you to select a reference to the component, and the name of the property to refresh. IE:

   ```
   system.db.refresh(component, "propertyName")
   ```

6. Click **OK**.

Next...

- Deleting Multi-Selected Rows from Table

Deleting Multi-Selected Rows from Table

A common user experience is to select multiple rows of a table and then delete those rows from the database by clicking a Delete button. When the user selects more than one row in a table, there is a special function available on the table that makes accessible 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 as shown in this example code that can be called from a button's **actionPerformed** event handler.

```
data = event.source.parent.getComponent('Table').data   #Get the data from
the table and assign it to the variable called data.
rows = event.source.parent.getComponent('Table').getSelectedRows()   #Get
the rows of the data that the user has currently highlighted.

for row in rows:        #Iterate through each row of the list that is
associated with the rows variable.
        id = data.getValueAt(row, "id") #Get the value associated with the
current row and the the column called "id"
        system.db.runPrepUpdate("DELETE FROM tableName WHERE id = ?",
[id]) #Run the query to delete from the database.
```

This code makes a few assumptions. First that there is table named **tableName** in your database, and that the table's Data property is bound to the same table.

Next...

Deleting Multi-Selected Rows from Table

Watch the Video

Database Examples

Here are examples on how to store information in a database to do more with Ignition.

In this section ...

Comments Panel Component

The following feature is new in Ignition version **7.8.1**
Click here to check out the other new features

Looking for documentation on the legacy Comments Panel component? Please see the Legacy Comments Panel page.

Not sure which version you are looking at? The Legacy version of this component has several properties that the new one does not: "Insert Query 1", "Insert Query 2", "Delete Query", "Unstick Query", and "Download Attachment Query".

There are many ways to set up the **Comments Panel** component, but the same first few steps exist in all cases to get started. To set up a simple **Comments Panel** example, here are the basic steps:

1. Add the **Comments Panel** component to a window.
2. Create or locate a database table to store notes in. You can follow the sample table in the "data" property of the **Comments Panel** Component.
3. Add a SQL query to the "data" property of the component to fetch the notes. Make sure to sort the records by timestamp so your newest comments are on top.
4. Save your project and test it out!

By default, a certain database structure is expected. If you want to use different tables, the Extension Functions on this component must be modified.

Additional documentation on the Comments Panel component can be found in the Appendix

Example

Three-Table (default) Configuration

The following section assumes the default configuration: all Extension Functions on the component are disabled.

Three-Table Configuration: Required Database Tables

The default behavior of the component expects three database tables be present under the same database connection, and each table needs to have certain columns with specific names.

Table:Notes

Column Name	Description	Data Type
id	An auto-incrementing integer that is the primary key. This maps to the ID field in the dataset.	Integer
whoID	A mapping to the Username field in the dataset	Integer
tStamp	A mapping to the Timestamp field in the dataset	Date or Datetime
note	A mapping to the NoteText field in the dataset	Varchar
filename	A mapping to the AttachmentFilename in the dataset	Varchar
sticky	A mapping to the Sticky field in the dataset	Boolean or Integer

attachment	A column to hold the attachment data. LongBlobs do not exist in MSSQL, so a varbinary type must be used	LongBlob or Varbinary (depending on database)

Table: ItemNotes

Column Name	Description	Data Type
accountId	An automatically generated UUID for the Comment Panel instance. You can use the accountId in a WHERE clause on the data property so that the component only shows notes from a particular Comments Panel in the project.	Varchar
noteId	An integer that maps to the ID column on the Notes table	Integer

Tables: Users

Column Name	Description	Data Type
id	An integer that is inserted into the whoID column on the Notes table	Integer
username	The username of the user that created the note	Varchar

Three-Table Configuration: Configuring the Data property

This component expects that its dataset is populated with the following columns. The names do need to be exact, but different names can be used as long as the query that builds the dataset uses aliases. The data type for each column in your notes table must match the table below.

Column Name	Description	Data Type
id	an integer that should be the primary key for the notes table. Used for deleting and looking up attachments	integer
username (or whoid)	the user who added the note	string/varchar
tstamp	when the note was added	date or dateTime
note	The text of the note itself	string/varchar
filename	filename for a file attached to the note	string/varchar
sticky	0 or 1 indicating whether or note the note is "sticky", which means it gets highlighted and put at the top	boolean or integer

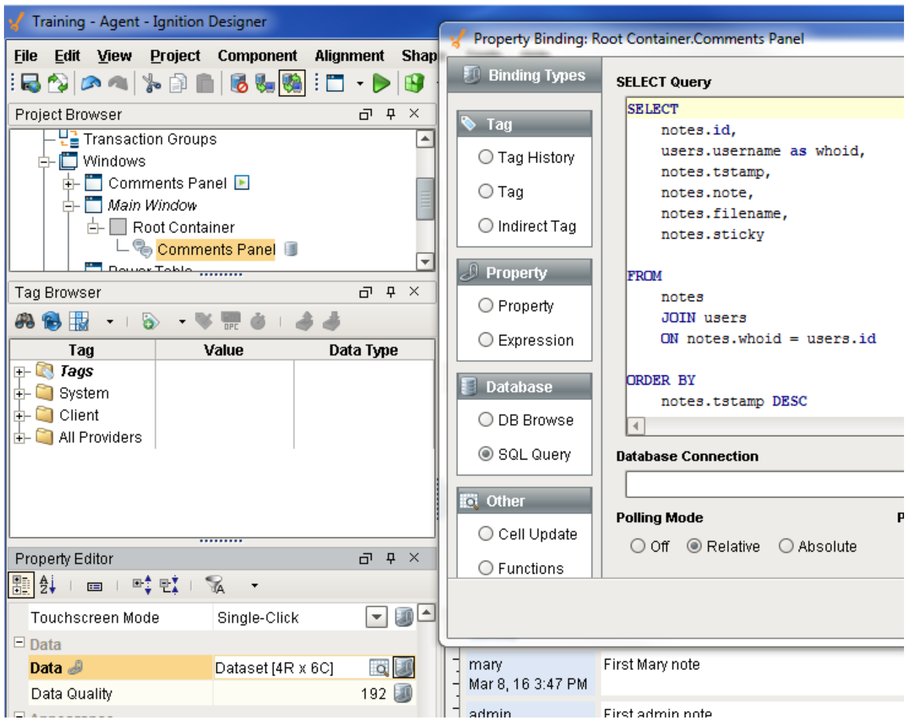
The comments panel also has a set of default queries to handle adding, deleting, and unsticking comments. The default queries expect certain database tables and columns to be set up. If your database is set up differently, or your dataset is different than the default, remember to enable the Extension Functions and provide the correct names.

Example

The following query returns note data from the above tables, and displays the data on a Comments Panel component. This query should be placed in a SQL Query binding on the Data property

```
 SELECT
        notes.id,
        users.username
as whoid,
        notes.tstamp,
        notes.note,
        notes.filename,
        notes.sticky

FROM
        notes
        JOIN users
        ON notes.whoid
= users.id
```

```
ORDER BY
        notes.tstamp
DESC
```



```
SELECT
    notes.id,
    users.username as whoid,
    notes.tstamp,
    notes.note,
    notes.filename,
    notes.sticky

FROM
    notes
    JOIN users
    ON notes.whoid = users.id

ORDER BY
    notes.tstamp DESC
```

Binding Types: SQL Query; Polling Mode: Relative

Tag	Value	Data Type
Tags		
System		
Client		
All Providers		

Touchscreen Mode	Single-Click
Data	Dataset [4R x 6C]
Data Quality	192

Next...

- Storing Files in a Database
- Comments Panel

Storing Files in a Database

Ignition can display and store PDF files from data stored in databases.

Displaying PDF files from the database

Ignition can render the PDF inside the PDF viewer component. The PDF viewer component is included with the Reporting Components. To view PDF files in the Client, your Ignition server must have this module. Once the module is installed, you can load the bytes from the database into the PDF viewer component with the following script which is enacted on a drop-down component property event handler.

```
id = event.newValue  #First get the id from the dropdown.
bytes = system.db.runScalarQuery("SELECT data FROM documents WHERE id = %
d" % id)  #Query for the raw pdf data by using the id.
name = system.db.runScalarQuery("SELECT name FROM documents WHERE id = %d"
% id)   #Query for the file's name by using the id.
event.source.parent.getComponent('PDF Viewer').loadPDFBytes(bytes, name)
#Load the bytes and the name into the PDF viewer component.
```

Uploading PDF files to the database

Ignition can upload PDF documents to a database. Imagine the user selecting a button, a file browser pops up, the user selects the PDF file, presses enter and the file gets uploaded into the database. The following script demonstrates how this might work:

```
path = system.file.openFile("pdf")   #Find the path to the PDF file.
data = system.file.readFileAsBytes(path)   #Read the file as bytes.
name = system.gui.inputBox("Enter a name for the file")   #Have the user
enter a name for the file.
system.db.runPrepUpdate("INSERT INTO documents (name, data) VALUES (?,?)",
[name,data])  #Insert the data and name into the database.
```

INDUCTIVE
UNIVERSIT

Storing Files in a Database

Watch the Video

In a database, each column is a certain datatype. The datatype where you can store PDF bytes is a LONGBLOB datatype, therefore to insert the PDF file's bytes to a column the column's datatype must be a LONGBLOB.

Next...

Scripting

Scripting is used in many places in Ignition to add a significant degree of flexibility and customization where pre-canned options fall short. There are two major scripting languages in Ignition, Python and the Expression Language. It is important to understand the differences between the two, and to know where each is used.

What is Python?

Most of the time when we talk about "scripting" we're talking about Python scripting. Python is a general purpose programming language that was developed in the early 90's and has gained significant popularity in the 2000's. 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.

Since Python is such a popular and well-regarded language, there are many high-quality tutorials available on the web. The official Python tutorial, written by the inventor of Python himself, Guido van Rossum, is very good. https://docs.python.org/2/tutorial/index.html

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.

Why not VBA?

Many HMI/SCADA packages use VBA, or Visual Basic for Applications. As such, many engineers switching to our software inquire about it. There are a variety of reasons we don't use VBA:

- It is not compatible with Java, the language that Ignition is written in. This also means that it is not cross-platform.
- It is a dying language (Microsoft has phased it out as of July, 2007).
- It is full of security holes.
- It is an ugly language!

Where is Python Used?

Python is used in many places in Ignition. The most apparent place is in component event handlers. Project event scripts are another major place where Python is used.

Which version of Python is Used?

Ignition uses Jython 2.5. 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 of the documentation.

What is the Expression Language and how is it different from Scripting?

The Expression Language is a simple 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 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 is similar. It is used to define dynamic values for tags and component properties. The Expressions and Python are often confused but they are used in very different spaces. The Expression language is most commonly used in expression bindings, and the Python language is used in Ignitions Events.

In this section ...

Scripting Basics

In this section ...

Scripting Intro

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". We've included a short tutorial here which should help get you started. If you find yourself doing a lot of scripting, you may want to pick up a basic reference book about Python.

INDUCTIVE UNIVERSITY

Scripting Console

Watch the Video

Getting Help with Scripting

Scripting is one of the topics in Ignition that users frequently need help with, because it is used to achieve some of the most complex requirements of a project. If you get stuck designing a script, or would like help getting started, don't hesitate to get some help from our user forum at http://www.inductiveautomation.com/forum, it is by far the best place for scripting help.

When asking for scripting help - be precise and complete. If you're working through an error - include the text of the error, the circumstances, and the offending code. If you're stuck on something, it is helpful to describe the broader goals of what you're trying to accomplish - there is often an easy way and a hard way. Don't be shy to simply ask for some direction getting started.

Under the Hood - Python in Java

The implementation of Python included in Ignition is Jython 2.5. While this is not the "latest" version of Python, it supports the vast majority of what people consider "python". The latest version of Python, version 3, is actually more of a branch, with many significant changes. The vast majority of sample code to be found was written against version 2.

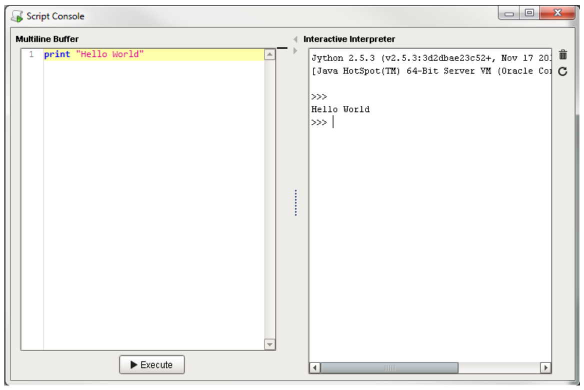
One of the powerful things about using Jython is that your script has access to the entire Java standard library. In the Client, this will be Java 5 or above. When running under the Gateway, this will be Java 6 and above. For more information, see Accessing Java.

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.

Next...

- Scripting Console

Scripting Console



```
print "Hello World"
```

```
Jython 2.5.3 (v2.5.3:3d2dbae23c52+, Nov 17 20
[Java HotSpot(TM) 64-Bit Server VM (Oracle Co

>>>
Hello World
>>>
```

INDUCTIVE UNIVERSITY

Scripting Console

Watch the Video

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.

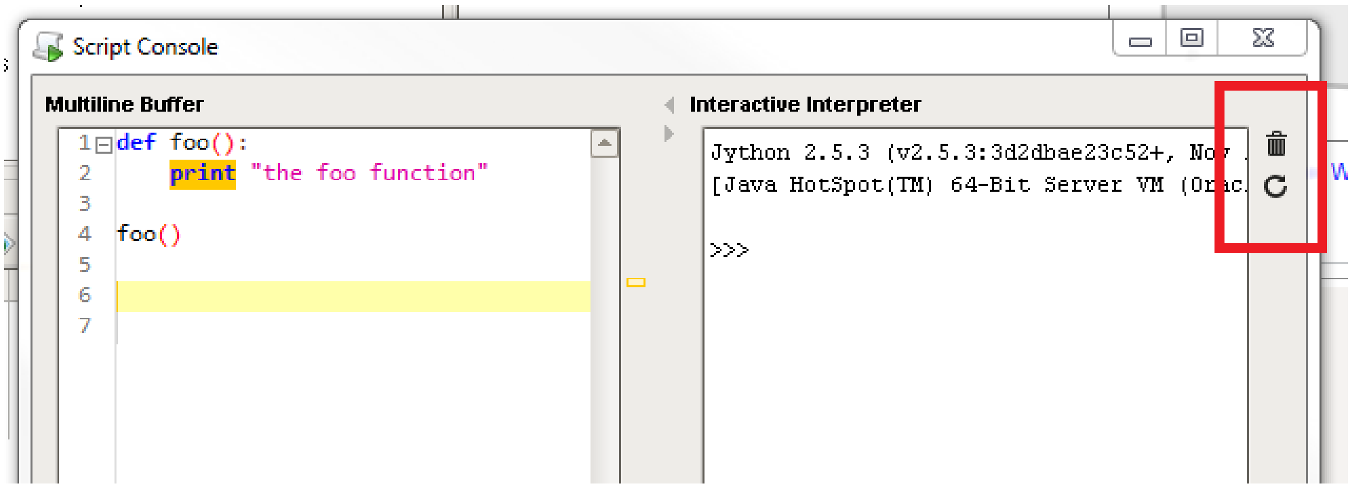
Due to the scope of the Script Console, it can not interact with components on a window.

It can call Project and Shared scripts. If a Project or Shared script was recently added, then the conosle will need to be reset before they may be called. This can be accomplished by clicking on the Reset button.

> ⚠ Gateway scoped information will not appear in either the Script Console or Output Console. Instead they will output 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.

Features

There are two buttons located on the right side of the console.



```
def foo():
    print "the foo function"

foo()
```

```
Jython 2.5.3 (v2.5.3:3d2dbae23c52+, Nov
[Java HotSpot(TM) 64-Bit Server VM (Orac

>>>
```

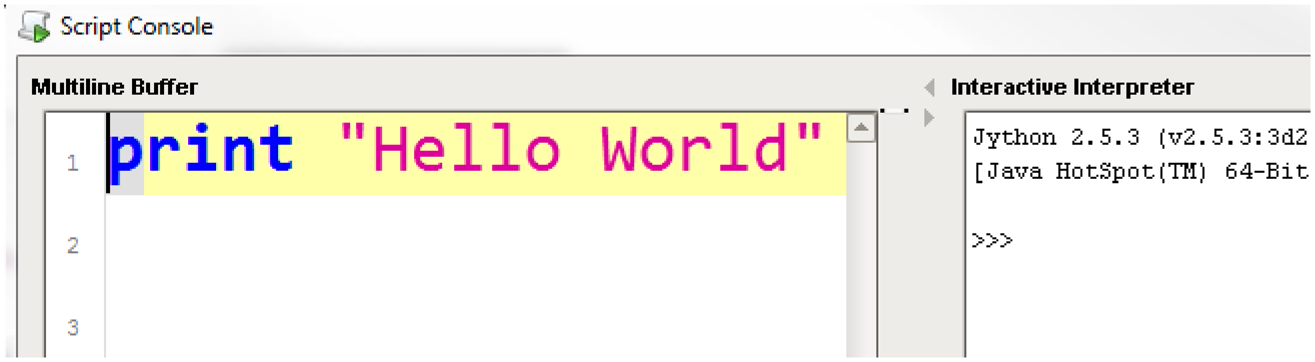
- Clear clears the text from the Interactive Interpreter.
- Reset clears the text and deletes all user defined objects (variables and functions) from the interactive Interpreter.

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 statments will output to the Interactive Interpreter. Supports code folding for function definitions and comments.

Font Size Adjustment

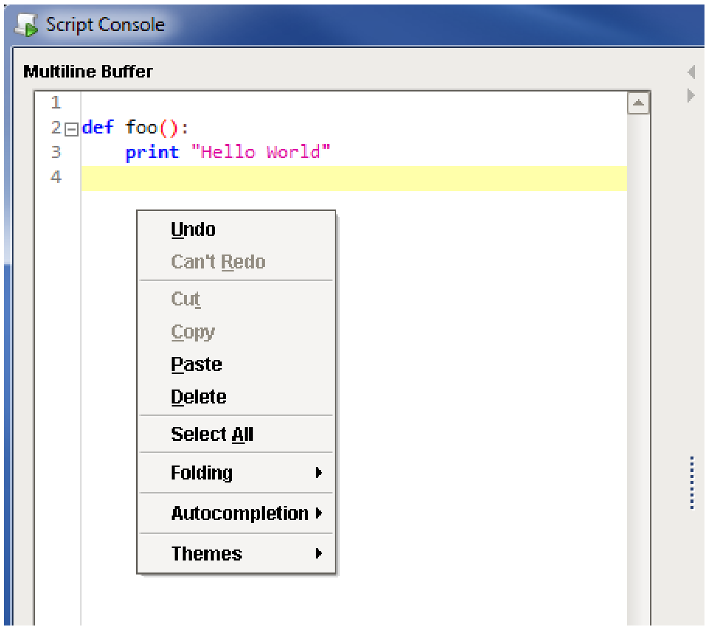
The Font size in the Multiline Buffer can be adjusted by holding Ctrl and scrolling the mouse wheel



```
print "Hello World"
```

```
Jython 2.5.3 (v2.5.3:3d2
[Java HotSpot(TM) 64-Bit

>>>
```

Right Click Menu



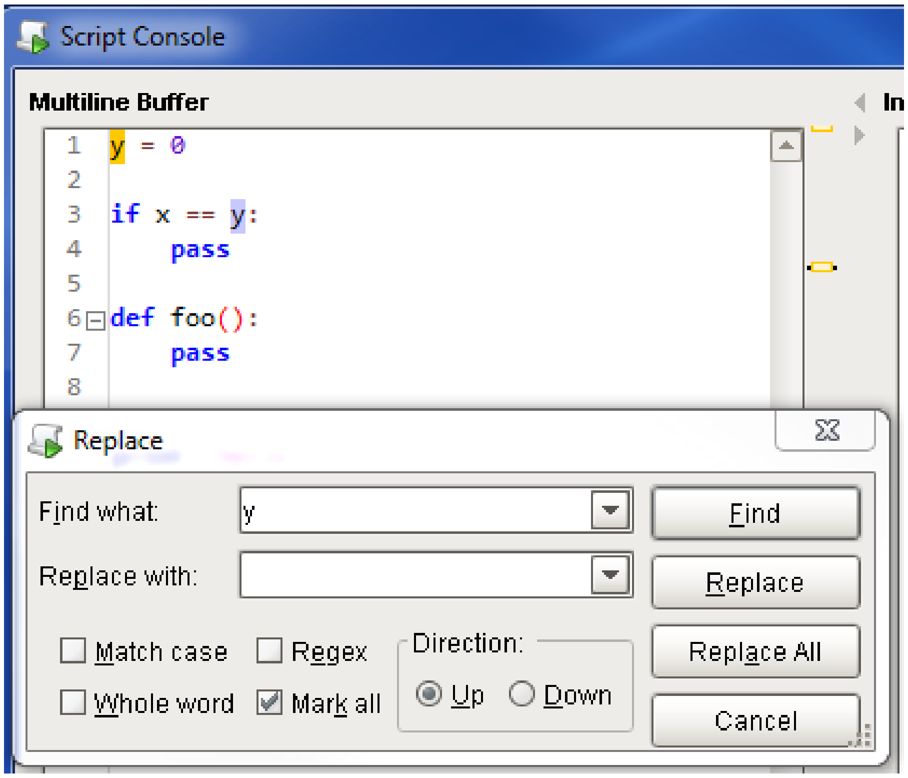
```

def foo():
    print "Hello World"

```

Right clicking on the Multiline Buffer opens a menu. Most of the options in the menu are self explanatory. The rest are defined below:

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	Collapse all expanded folds.

All Folds	
Expand All Folds	Expand all collapsed 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



```
y = 0

if x == y:
    pass

def foo():
    pass

```

Find what: y

Replace with:

[ ] Match case [ ] Regex

[ ] Whole word [✓ Mark all]

Direction: [✓ Up] Down

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.

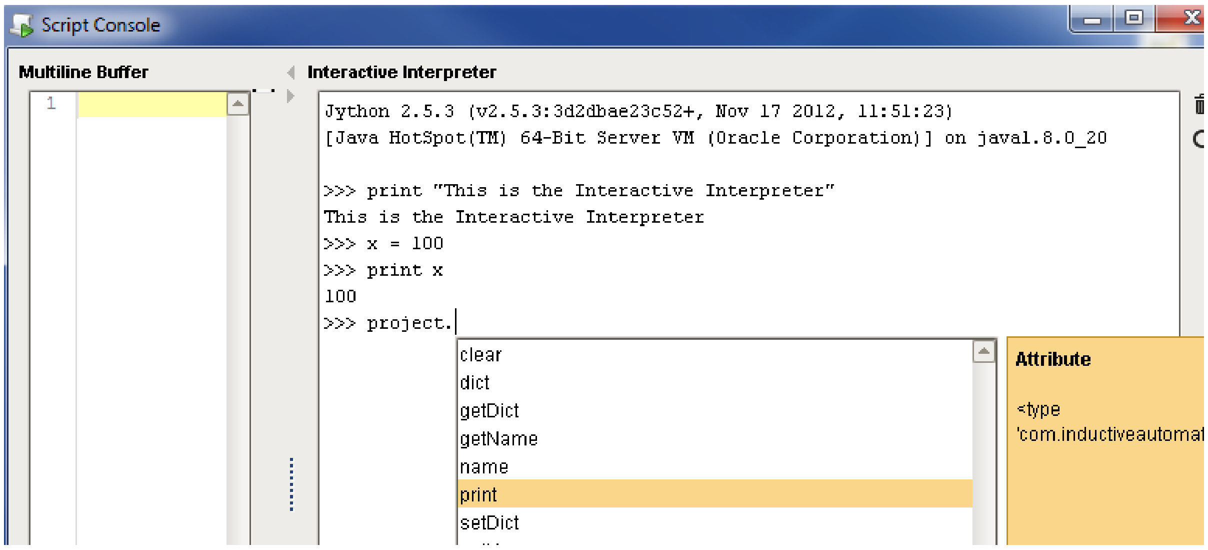
Keyboard Shortcuts

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

Interactive Interpreter



```
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.
```

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 Enter key. Print statements from both sides of the Script Console will appear in the Interactive Interpreter.

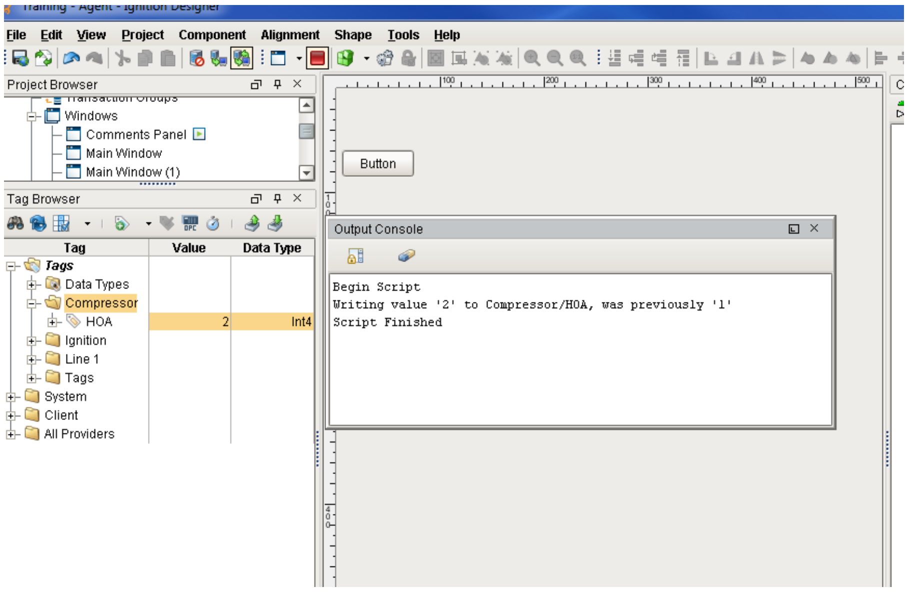
The Autocompletion window available in the Interactive Interpreter has access to the current working environment, so items such as Project and Shared scripts will appear. This will automatically appear, but can be called manually.

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 "**<object>.**" has been typed (assuming a valid object such as "system" or "project" if 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

Output Console



```
Begin Script
Writing value '2' to Compressor/HOA, was previously '1'
Script Finished
```

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** keyb oard shortcut.

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:

```
# 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'
```

Note: The output console is also available in the Vision Client from **Help > Diagnostics**.

Next...

- Python Variables

Python 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

```
x=5
y=3
print x*y
```

Python Variables

Watch the Video

Next...

- Python Datatypes

Python Datatypes

Strings, Numbers, and Booleans

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. 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. There is no harm in marking all strings as unicode strings. The following prints a unicode string:

```
print u'été'
```

The None Value

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.

List

In Python, lists (arrays) are a built-in type that contains multiple other values. Lists can contain any type of items, and the items in a list do not all need to be the same type. You can create a list by enclosing multiple items in square brackets ([]), separated with commas. You can pull items out of a list with the square-bracket list index notation. Note that lists are zero-indexed, meaning that the first item in the list is at position 0. This code will print out "a list".

Basic Operators

Python has all of the normal arithmetic operators you'd expect, addition(+), subtraction(-), division(/), multiplication(*), modulus(%), etc.

The comparison operators are just like in C: equals(==), not equals(!=) greater than (>), greater than or equal(>=), etc.

The logical operators are just typed in plain text: and, or, not.

These are just the basics. There are other operators, like bit shift operators. Read about them at: http://docs.python.org/library/stdtypes.html

White Space

Perhaps its most unique feature, logical blocks are defined by indentation in Python. A colon (:) starts a new block, and the next line must be indented (typically using a tab of 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". The final print is not part of the loop, because it isn't indented.

```
countdown=5
while countdown > 0:
   print countdown,
   countdown = countdown - 1
print "Blast-off!"
```

Next ...

- Control Flow Logic

Control Flow Logic

Control flow are the parts of a language that make it do things differently based upon various conditions. In other words: ifs and loops. Python has all of the basic control flow statements that you'd expect.

`if` Statements

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. 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. There is no harm in marking all strings as unicode strings. The following prints a unicode string:

```
print u'été'
```

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 if a certain condition is true. For example, this script won't do anything.

```
x = 15
if x < 10:
   print "this will never show"
```

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. This script will print out **`this will show!`**

```
x = 15
if x < 10:
   print "this will never show"
else:
   print "this will show!"
```

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`**:

```
x = 3
if x == 1:
   print "one"
elif x == 2:
   print "two"
elif x == 3:
   print "three"
else:
   print "not 1-3"
```

Next ...

- Control Flow Loops

Control Flow Loops

While Loop

A while loop will repeat a block of statements while a condition is true. This code will print out the contents of the items in the list. This code uses a function called len, which is a built-in function that returns the length of a list or string.

```
listOfFruit = ['Apples', 'Oranges', 'Bananas']
x = 0
while x < len(listOfFruit):
   print listOfFruit[x]
   x = x + 1
```

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. So, we could re-write the example above using a for loop eliminating the counter x:

```
listOfFruit = ['Apples', 'Oranges', 'Bananas']
for item in listOfFruit:
   print item
```

Much more graceful! You'll often see the for loop used instead of the while loop, even when you simply want to iterate a given number of times. To do this with the for loop, you can use the built-in function range. The range function returns a variable-size list of integers starting at zero. Calling range(4) will return the list [0, 1, 2, 3]. So, to have a for loop repeat 4 times, you simply can do:

```
for x in range(4):
   print "this will print 4 times"
```

Break and Continue In Loops

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".

```
for x in range(10):
   if x >= 2:
      break
   print "Loop"
print "Finished"
```

You can use the continue statement to make a loop stop executing its current iteration and skip to the next one. The following code will print out the numbers 0-9, skipping 4

```
for x in range(10):
   if x == 4:
      continue
   print x
```

Next ...

- Functions

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 libraray.

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.

Defining Functions

Functions are defined using the def keyword. A function needs a name, and needs a list of the arguments that it can be passed. For example, this code defines a function that tests whether or not a number is odd. It returns a true value (1) if the number is odd. It is then used in a loop to print out the odd numbers between 0 and 9.

```
def isOdd(num):
   return num % 2 == 1 # uses the modulus (or remainder) operator

for x in range(10):
   if isOdd(x):
      print x
```

Function 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. In the example above, the value of x inside the for loop gets passed to the isOdd function, and becomes the value of the num argument.

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 calledcap, which will take a number, and make sure it is within an upper and lower limit. The limits default to 0 and 100.

```
def cap(x, min=0, max=100):
   if x < min:
      return min
   elif x > max:
      return max
   else:
      return x

# This will print out "0"
print cap(-1)

# This will print out "100"
print cap(150)

# this will print out "150", because it uses a max of 200
print cap(150, 0, 200)
```

Keyword Arguments

Arguments can also be specified by keyword instead of by position. In the above example, 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.

```
print cap(150, max=200)
```

Note that not all functions in the standard library and the Ignition library can be called with keyword invocation. Functions that accept keyword invocation, like system.tag.queryTagHistory, will say so in their documentation.

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++).

Lets go back to the isOdd example above. Suppose we wanted a more general way to filter a list. Maybe sometimes we want the odd entries, while other times we want even ones, or entries less than 3, etc. We can define a function called extract that takes a list and another function, and returns only entries that "pass" through the other function.

```
def isOdd(num):
   return num % 2 == 1

def isEven(num):
   return num % 2 == 0

def isLessThan(num, max=3):
   return num < max

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))
```

Now, it just so happens that Python has a built-in function that does exactly what our extract function does - its called filter.

We would also be remiss at this point if we didn't mention another language feature called list comprehensions. This is a great little bit of syntax that helps make new lists out of other lists. Instead of using our filter function, we could have simply done this:

```
def isEven(num):
   return num % 2 == 0

print [x for x in range(10) if isEven(x)]
```

If that looks cool to you - read more about list comprehensions at http://docs.python.org/tutorial/datastructures.html#list-comprehensions

In Ignition, you'll most commonly see functions used as objects when using the system.util.invokelater function. This function takes a function and executes it after all pending event handling has finished processing.

Next ...

- Python Comments

Python Comments

Comments start with a hash sign. By adding comments to your code, you know what the code is trying to do when you go back to it after a long time.

```
# Prints out 'Hello World' 5 times.
for x in range(5):
   print 'Hello world'
```

Next ...

- System Library

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.

```
value = system.tag.value("tagPath").value
```

The appendix is full of built-in functions such as this.

Component Scripting

Event handling allows you to use scripting to respond to a wide variety of events that components fire. This lets you configure windows that are very interactive, and are an important part of project design in the Vision module.

Events

An event can be many things, like a mouse click, a key press, or simply a property changing. Whenever these events occur, a script can be called to "handle" the event. Different components can fire different types of events. For example, mouse events are very common and are fired by almost all components. The **cellEdited** event, on the other hand, is only fired by the Table component.

Configuring Handlers

To configure event handlers for a component, right-click on the component and choose the **Scripting** from the pop-up window. You can also get to this button from the toolbar ( ) or the Component menu. From the component scripting window, you can pick the appropriate event handler.

Script Builders

All events are handled with scripting, but you frequently don't need to write the scripts by hand. This is where the Script Builders come in. For each event, you can choose a common way of handling the event. This can be a navigation action, setting a tag value, and so on. To write an arbitrary script, choose the **Script Editor** tab.

For example, one of the most common uses of event handlers is to open a window when a button is pushed. To do this, simply select the `actionPerformed` event, and select the **Navigation** tab. Here you can simply pick the navigation action **Open**, and choose the window to open. If you're curious, you can peek over at the **Script Editor** tab to see the underlying code that makes this action work, but you certainly don't have to.

Common Component Functions

The following functions are common to all components in Ignition:

`requestFocusInWindow()` - requests that the component be given input focus. See also: `focus` Events.
`setPropertyValue(name, value)` - sets the value of a component's Custom property.
`getPropertyValue(name)` - gets the value of a Custom property.

Moving/Resizing Components and Windows

You can use scripting to move and resize a component at runtime. The functions system.gui.moveComponent, system.gui.reshapeComponent and system.gui.resizeComponent are used for this.

They simply take a component, and a new size, location, or both. Locations are always measured in pixels from the upper left point of the component's parent container.

Note that if you're moving a component that is set to relative layout, your coordinates will act as if they were coordinates to the sizes of the relevant containers last time they were saved in the Designer, not the current real coordinates of the runtime. This is very helpful for creating animations. In effect what this means is that the coordinates fed into these functions "respect" the relative layout system automatically.
You can move and resize windows as well. If you have a reference to a window, you can set its size and location directly with setSize() and setLocation(). For example, if you wanted to move a window named "Popup" to a certain location:

setSize and setLocation

```
try:
        window = system.gui.getWindow("Popup")
        window.setSize(250,600)
        window.setLocation(0,0)
except ValueError:
        pass # ignore error with a pass keyword pass
```

In this section ...

Component Event Handlers

When you're writing component event handlers, you'll do a lot of work with components. You'll need to reference various components on the window or on other windows, you'll need to reference and set properties of the component, you may even want to move components around on the screen.

Finding Components

When you have an `event` object, that object becomes your window into the entire component hierarchy. `event.source` references the component that fired whatever event you're responding to. `event.source.parent` references the container that component is in `event.source.parent.getComponent("Name")` finds a sibling component with a certain name. The manual page for the event object covers this topic in more detail.

These are all of the event types that are fired by the various components in the Vision module. Events are organized into event sets. For example, the mouse event set includes `mouseClicked`, `mousePressed`, and `mouseReleased`. All of the events in an event set share the same properties for their `event` object.

Event Sets

Each component can have its own set of events, however most components share events, therefore, understanding the way one event works with one component will result in understanding how it works will all other components that have the same event.

The most common event sets are as follows:

- action
- cell
- focus
- internalFrame
- item
- key
- mouse
- mouseMotion
- paint
- propertyChange

`action` Events

Events

`actionPerformed`

Properties in `'event'`

`source`

The `actionPerformed` event is fired when an "action" occurs. What that "action" is depends on the component. The most common example is the Button component. You should always use the action event on a button instead of a mouse click, because it will be fired whenever the button is pressed, whether it is via the mouse or the keyboard (via a mnemonic shortcut or tabbing over to the button and pressing enter or space). The Timer component is another example of a component that fires an action event. In this case, the action is the timer firing.

`cell` Events

Events

`cellEdited`

Properties in `'event'`

`source`
`oldValue` - the previous value in the cell
`newValue` - the newly entered value for the cell
`row`
`column`

Cell events are fired by a Table component that has editable columns. When a user edits a cell, this event will fire. The `oldValue` and `newValue` properties in the event can be used to determine what value the cell used to hold, and what new value the user has entered. The `row` and `column` properties, both integers, show what position in the table's data property the edit occurred at.

Example

Commonly, the event handler for a cell event will issue a SQL update query to persist changes to the table back to an external database. You can use the row to determine what the primary keys were for the row that was edited by looking at the table's data property. You can use the column index to find the column name of the edited column.

```
columnName = event.source.data.getColumnName(event.column)
primaryKeyValue = event.source.data.getValueAt(event.row, "keycolumn")
query = "UPDATE mytable SET %s=? WHERE keycolumn=?" % columnName
system.db.runPrepUpdate(query, [event.newValue, primaryKeyValue])
```

`focus` Events

Events

`focusGained`
`focusLost`

Properties in `'event'`

`source`
`oppositeComponent` - the component that either gave up focus to this component, or took it away

Focus events are fired for components that can receive input focus. For both the focus gained and focus lost events, you can also access the "opposite" component. For a focus gain, this is the component that previously had the focus. For a focus lost event, the opposite component is the component that took the focus away.

You can programatically request that focus be given to a component by calling the function `requestFocusInWindow()` on that function. This function is actually defined by Java's JComponent class, from which all Vision components extend.

If you are trying to alter the focus from within a focus event handler, you must wrap your code in a call to `system.util.invokeLater`. This will let your focus change be processed after the current focus change event that is being processed has a chance to finish.

`internalFrame` Events

Events

`internalFrameActivated` - fired when the window becomes the focused window
`internalFrameClosed` - fired after the window is closed
`internalFrameClosing` - fired just before the window is closed
`internalFrameDeactivated` - fired when the window loses focus
`internalFrameOpened` - fired the first time a window is opened after not being in the cache

Properties in `'event'`

`source`

Internal frame events are fired by windows. (They are known as "internal frames" in the underlying Java windowing system that the Vision component uses). Note that the source of these events is the window itself. To get the root container of the window, use `event.source.rootContainer`, not `event.source.getComponent("Root Container")`.

The Activated/Deactivated events get fired when the component receives or loses input focus. The Activated event is a more reliable event to use as a window startup event than the Opened event, because the Opened event will not be called if the window was opened when it was already cached.

See also: Window's Cache Policy

`item` Events

Events

`itemStateChanged`

Properties in `'event'`

`source`

`stateChange` - a code that will be equal to either the SELECTED or DESELECTED constants.

`SELECTED` - a constant representing a selection event.

`DESELECTED` - a constant representing a deselection event.

The `itemStateChanged` event is used by components that choose between a selected or deselected state. For example, a Check Box or Radio Button. You can respond to this event to be notified when the state has changed (via any mechanism - click, keyboard, property bindings, and so on). To check whether the event represents a selection or a deselection, you compare the event's stateChange property with the SELECTED or DESELECTED constants, like this:

```
if event.stateChange == event.SELECTED:
   print "Turned ON"
else:
   print "Turned OFF"
```

key Events

Events

`keyPressed` - fires when a key is pressed while the source component has input focus. Works for all keyboard keys.

`keyReleased` - fires when a key is released while the source component has input focus. Works for all keyboard keys.

`keyTyped` - fired when a character key is pressed and then released while a component has input focus.

Properties in `'event'`

`source`

`keyCode` - an integer code representing the key that was pressed or released. Only valid on `keyPressed` and `keyReleased` events. See table below.

`keyChar` - a string that represents the character that was typed, if applicable (e.g. used for letters, but not an F-key). Only valid on keyTyped event.

`keyLocation` - the location of the key. E.g. to differentiate between left shift from right shift.

`altDown` - true (1) if the alt key was held down during this event, false (0) otherwise.

`controlDown` - true (1) if the control key was held down during this event, false (0) otherwise.

`shiftDown` - true (1) if the shift key was held down during this event, false (0) otherwise.

Key events are used to respond to keyboard input. They will only be fired on components that receive input focus. Handling key events often involves checking exactly what key was pressed. These events make a distinction between character keys (`A,B,C...`) and non-printable keys (`F3,Esc, Enter`). All keys will get `keyPressed` and `keyReleased` events, but only character keys will get `keyTyped` events. For `keyTyped` events, checking what key was pressed is relatively simple, you can simply do a comparison on `keyChar`, like `event.keyChar == 'a'`. For other keys, however, you need to compare the keyCode to a constant, enumerated below. These constants can be referenced through the event object itself, like: `event.keyCode == event.VK_ENTER`.

Key Code Constants		
VK_0 - VK_9	VK_END	VK_PAGE_UP
VK_A - VK_Z	VK_ENTER	VK_RIGHT
VK_F1 - VK_F24	VK_HOME	VK_SHIFT
VK_ALT	VK_INSERT	VK_SPACE
VK_CONTROL	VK_LEFT	VK_TAB

VK_DOWN	VK_PAGE_DOWN	VK_UP
VK_ESCAPE		
Location Code Constants		
KEY_LOCATION_LEFT	KEY_LOCATION_RIGHT	KEY_LOCATION_UNKNOWN
KEY_LOCATION_NUMPAD	KEY_LOCATION_STANDARD	

All of this information comes straight out of the Java documentation for `java.awt.KeyEvent`.
See http://java.sun.com/j2se/1.5.0/docs/api/java/awt/event/KeyEvent.html

`mouse` Events

Events

`mouseClicked` - fired when the mouse is pressed and released in the same spot on the component.
`mouseEntered` - fired when the mouse is moved so that it is hovering over the component
`mouseExited` - fired when the mouse had been hovering over the component and exits
`mousePressed` - fired when the mouse is pressed within the bounds of the component
`mouseReleased` - fired when the mouse is released after having been pressed within the bounds of the component

Properties in `'event'`

`source`

`button` - an integer code representing the button that was clicked. Use the constants `event.BUTTON1`, `event.BUTTON2`, and `event.BUTTON3`.

`clickCount` - an integer count of the number of successive clicks.

Click Count

```
#Use code like this to ensure a specific number of clicks before your code runs.
if event.clickCount == 2:
        print "I was clicked twice!"
```

`x` - the x-axis location of the mouse click, with (0,0) being the upper left corner of the component.
`y` - the y-axis location of the mouse click, with (0,0) being the upper left corner of the component.
`popupTrigger` - true(1) if this mouse event should pop up a context menu. Meaning is OS-dependent. On windows, it is a release of `BUTTON3`.
`altDown` - true (1) if the alt key was held down during this event, false (0) otherwise.
`controlDown` - true (1) if the control key was held down during this event, false (0) otherwise.
`shiftDown` - true (1) if the shift key was held down during this event, false (0) otherwise.

mouseMotion Events

mouseMotion events will not trigger when the project is viewed from a mobile project as these gestures are used by the browser/device to zoom or pan.

Events

`mouseDragged` - fires when the mouse is pressed within the component, and then moved. Will continue to fire until the button is released, even if the mouse moves outside the component.
`mouseMoved` - fired when the mouse moves over the component.

Properties in `'event'`

See mouse events.

`paint` Events

Events

`repaint`

Properties in `'event'`

`source`

`graphics` - An instance of `java.awt.Graphics2D` that can be used to paint this component. The point (0,0) is located at the upper left of the component.
`width` - The width of the paintable area of the component. This takes into account the component's border.
`height` - The height of the paintable area of the component. This takes into account the component's border.

This event is fired by the Paintable Canvas component. This component is provided for highly scriptliterate users, and is decidedly not user-friendly. Don't say you weren't warned. It allows you to use Java2D through Python to programatically "paint" your own dynamic, vector-based component. This event is called every time the component needs to repaint. It will repaint when any of its custom properties change, or when `.repaint()` is called on it. Drop a Paintable Canvas onto a window and look at the paint event handler for an example.

`propertyChange` Events

Events

`propertyChange`

Properties in `'event'`

`source`

`newValue` - The new value of the property.

`oldValue` - The previous value of the property. Not all properties provide this information.

`propertyName` - The name of the property that has changed.

The `propertyChange` event is called any time a bindable property changes on a component. This includes all custom properties. This can be a very useful tool, allowing you to respond via scripting when a property changes. Because this one event handler is called for multiple properties, it is typical for a handler to first have to filter based on the `propertyName`, so that it is responding to a specific property changing.

Example

```
#This script might go on a Table whose data must be filled in before continuing
if event.propertyName == "data":
   newData = system.db.toPyDataSet(event.newValue)
   if len(newData)>0:
     # Data exists - let the user know they may proceed
     system.gui.messageBox("You may proceed.")
```

Next ...

- Accessing Component Properties

Script Builders

When creating an event handler, you can use one of the handy **script builders** instead of writing the script by hand. In the Event handlers configuration window, the script builders are accessible as tabs along the top. The last tab, **Script Editor**, lets you write an event handler by hand. You can also use it to view the script that was generated by the **script builder**, which is a good way to get started learning how to write event handlers by hand.

INDUCTIVE UNIVERSITY

Script Builders

Watch the Video

Action Qualifiers

All of the script builders allow you to put security and/or confirmation qualifiers onto the event handler. The security qualifier lets you restrict the event handler from running if the current user doesn't possess a set of roles. Use CTRL-select to pick multiple roles. The confirmation qualified will prompt the user with a popup **Yes/No** box. The action will only be executed if the user chooses **Yes**.

Navigation

The navigation script builder has various functions that deal with opening and closing windows.

Open / Swap

Opening is a very straight-forward operation - it simply opens the specified window. You are also given options to center that window within the Client, and to close the window that the event was fired from.

Swapping is the practice of opening another window in the same size, location, and state as the current window, and closing the current window. This gives the appearance of one window simply swapping into another, seamlessly. The navigation builder uses the swapWindow version of swapping, but most "by hand" script authors will us the swapTo version. This last version relies on the fact that the windows being swapped are both maximized windows. See the typical navigation strategy section for more information.

You can also pass parameters to the opened or swapped-to window. The names of these parameters must match names of custom properties on the root container of the target window. The values can either be literals or values of other properties from the source window. To use a property, highlight an empty cell in the Value column of the parameter table, and press the Insert Property () button. See the parameterized windows section for more information.

Forward / Back

These action give you a simple way of implementing *browser*-style forward/back buttons in your client. Note that you must be using the default navigation strategy for this to work, because these functions rely on calls to `system.nav.swapTo` in order to keep track of what the sequence of recent windows has been.

Closing Windows

These options allow for an easy way to have an event handler close the window that it is a part of, or any other window. Refer to the following sections for more information:

- Parameterized Windows
- Navigation Strategies
- system.nav.openWindow
- system.nav.swapWindow

Set Tag Value

This event handler script builder will respond to an event by setting the value of a Tag. You can set the tag to either a literal value directly typed in, or you can use the Insert Property () button to have the handler use the value of another property from the same window.

SQL Update

This script builder helps you build an update query using a database browsing interface. Choose a spot in your target database and the update query will be built for you. By setting columns as key columns, you can have the filter correctly filter to the right row. You may use either literal values or property values by using the Insert Property () button next to the Update Value text box.

Set Property

This script builder will respond to an event by altering a property in the window. You must choose the property to alter, and the value that you wish to assign to it. The value can be a literal value or the value of any other property on the window by using the Insert Property () button.

Next ...

- Event Object

Event Object

Event handling scripts are just regular Python scripts except for one important detail. They all have a special variable defined in their namespace called `event`. This is an object that represents information about the event that just occurred. For example, the `event` object for a mouse click will have the x and y coordinates where the click occurred. A key press event, on the other hand, will have a keycode, but not a coordinate.

In addition to information about the event that has just occurred, the `event` object has a `source` propert y. The source of an event is the component that fired it. This is a crucial concept to understand. The reference to the component is your handle into the entire hierarchy of the window that your script is contained in.

Event Object

Watch the Video

Example

Suppose you're handling the mouse-pressed event of a **label** component. The following script would print out the coordinates of the click, as well as the text of the label:

```
currentText = event.source.text
print 'Mouse clicked on label "%s" at %dx%d' % (currentText, event.x,
event.y)
```

The output would look like this if the label's text was "this is my label":

```
Mouse clicked on label "this is my label" at 27x99
```

Using the `event` object to access the component hierarchy

Because `event.source` is the component that fired the event, you can use this reference to access the entire hierarchy of your window. This means you can access properties of any other component in the window. You just need to know how to navigate up and down the component tree.

To navigate **up** the component tree (going from a component to its parent container), simply use the parent property.

To navigate **down** the component tree (going from a container to one if its children), use the `getComponent(name)` function.

To navigate **sideways** (getting a reference to a sibling component), simply go up one level and then back down.

Example

Suppose the component hierarchy in our window looked like this:

```
Root Container
   HeaderLabel
   StartButton
   Options
      ProductCode
      BatchSize
   PreviewTable
```

This window has a start button, a header, some options, and a preview table. Lets say that it is a window that lets the operator start a new batch. It has some options that are grouped into their own container. Lets say that the Root Container also has some parameters that our start button needs to know about.

The following table shows some script expressions and what they will evaluate to if you're writing an event handler for the **StartButton** component:

```
event.source #The start button
event.source.parent # The root container
event.source.parent.MyProperty #The value of custom property "MyProperty" on the Root Container
event.source.parent.getComponent("Options") #The Options container
event.source.parent.getComponent("Options").getComponent("ProductCode").selectedValue #The selected value of
```

```
the ProductCode dropdown component
event.source.parent.getComponent("PreviewTable").selectedRow #The index of the selected row in the
PreviewTable
```

There is one exception to the pattern of using `.parent` to go up the hierarchy and using `.getComponent(name)` to go down. The parent of a root container is not the window, and a reference to the window does not have a `.getComponent(name)` function. To get a reference to a window, simply use system.gui.getParentWindow with any component's `event` object as the parameter. Once you have a reference to a window, you can use its `.rootContainer` property to get to the root of the component hierarchy, and from here you can follow the rules laid out above.

Next ...

- Accessing Component Properties

Accessing Component Properties

The properties of a component can be influenced through scripting. Each component property comes with a scripting reference. Once you have a reference to a component, you can treat it just like any Python object. You can call functions on it, and you can reference its properties, both standard and dynamic, with the "." accessor.

Accessing Component Properties

Watch the Video

Example 1

To change the visibility of a component, access the component and set the property scripting name to either True or False to make the component visible or invisible.

```
event.source.parent.getComponent('Some Component').visible = False
```

Example 2

You could put this in a button next to the table, which would tell the user which row was selected, then clear the selection, and then print the table.

```
table = event.source.parent.getComponent("Table")

# Referencing properties of a component
row = table.selectedRow
system.gui.messageBox("The selected row is : %d" % row)

# Setting properties of a component.
table.selectedRow = -1

# Calling functions on components
table.print()
```

Next...

- Finding Components on Other Windows

Finding Components on Other Windows

In Ignition, you can obtain and use information from components on other windows. To access the information, first you need to know the window name, component name, and the property that you want to obtain. In addition, if the window is not open an error will occur. This error can be handled with normal exception handling.

The following example demonstrates how to access the contents of a text field on a different window from where the scripting occurred.

```
try:
        window = system.gui.getWindow("Other Window")
        text = window.rootContainer.getComponent("Text Field").text
        print text
except ValueError:
        print "Other window is not open"
```

Sometimes you may want to reference components on other windows. Or maybe you don't have an 'event' object because you're writing a project event script. In this case, you'll need to look up the containing window first. You can do this with the system.gui.getWindow function. This function will throw a ValueError if the given window isn't open, so you should make sure your code handles that gracefully. Once you have a Window, you can use its `rootContainer` property to get into the standard component hierarchy. This code will look up the HeaderLabel on a window named `"Overview"` and set its text and foreground color.

```
try:
   window = system.gui.getWindow("Overview")
   label = window.rootContainer.getComponent("HeaderLabel")
   label.text = "Notice Me!"
   label.foreground = (255,0,0)
except ValueError:
   # ignore error with a pass keyword
   pass
```

INDUCTIVE UNIVERSITY

Finding Components on Other Windows

Watch the Video

Next...

- Reading and Writing Tags

Reading and Writing Tags

Using the system functions, you can read from a tag and write to a tag in Ignition.

Reading from a tag

Reading a tag with scripting using a `system.tag.read()` function requires the tag path and will return a 'Qualified Value.' A qualified value is the tag value that has three attributes; Value, Quality, TimeStamp.

```
value = system.tag.read("tagPath").value
quality = system.tag.read("tagPath").quality
timestamp= system.tag.read("tagPath").timestamp
```

Writing to a tag

Writing to a tag with the `system.tag.write()` function requires the tag path and the value to write to the tag.

```
system.tag.write("tagPath","Hello World")
```

Reading and Writing Tags

Watch the Video

Next...

- Navigation Functions

Navigation Functions

The system.nav functions allow the user to navigate between windows.

Navigation Functions

Watch the Video

Next...

- Retargeting

Retargeting

Retargeting is accomplished through scripting, usually as a response to a button press or other component event. The **system.util.retarget** function allows you to 'retarget' the Client to a different project and/or different Gateway. You can have it switch to another project on the same Gateway, or another Gateway entirely, even across a WAN. This feature makes the vision of a seamless, enterprise-wide SCADA application a reality.

The retarget feature will attempt to transfer the current user credentials over to the new project / Gateway. If the credentials fail on that project, the user will be prompted for a valid username and password. Once valid authentication has been achieved, the currently running project is shut down, and the new project is loaded.

You can pass any information to the other project through the parameters dictionary. All entries in this dictionary will be set in the global scripting namespace in the other project. Even if you don't specify any parameters, the system will set the variable _RETARGET_FROM_PROJECT to the name of the current project and _RETARGET_FROM_Gateway to the address of the current Gateway.

INDUCTIVE UNIVERSITY

Retargeting

Watch the Video

Next...

- Working With Different Datatypes

Working With Different Datatypes

You'll encounter lots of different datatypes when scripting in Python.

This guide can help you through the snags of working with some of the more complex types. The University Video shows how you can work with colors and dates in the scripting language.

INDUCTIVE UNIVERSITY

Working with Different Datatypes

Watch the Video

Numbers

Working with numbers is very easy in Python, and requires no special considerations. You can use the built-in function `int()` to attempt to coerce values to integers, and `float()` to coerce values to floating-point values. Both will throw `ValueError` if the coercion fails.

> If you are new to programming, the following might throw you off. Python, like nearly all programming languages, uses integer division when dividing two integers. This means that `1/2` will result in `0`. This is because both 1 and 2 are integers, so the result of the division must be an integer. The result of `0.5` gets truncated to `0`. If either operand is a float, the result will be a float, so `1/2.0` will result in `0.5`.

Strings

Strings are used frequently in scripting. Strings can be defined using double quotes or single quotes. Learning how to use String Formatting is a very useful technique. You can user the built-in function `str()` to coerce values into strings.

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:

```
label = event.source
label.foreground = (255,0,0)
```

Dates

Dates are one of the trickier datatypes to deal with in scripting. It turns out that it is easier to deal with dates using the Java classes `java.util.Date` and `java.util.Calendar` than it is to use Python's `time` module.

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:

```
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`.

```
from java.util import Calendar
cal = Calendar.getInstance()
cal.setTime(myDate)
cal.add(Calendar.HOUR, -8)
myNewDate = cal.getTime()
```

Date Formatting

To format a date as a String, you can use the system function `system.db.dateFormat`. This function uses a *format string* to give it a hint as to how you want your date formatted. The format string is full of various placeholders that will display different parts of the date. These are case-sensitive! The most common placeholders are:

y	Year
M	Month
d	Day
E	Day of Week
a	am/pm marker
H	Hour of day (0-23)
h	Hour in am/pm (1-12)
m	Minute
s	Second
S	Millisecond
z	Time zone

These placeholders can be repeated for a different effect. For example, M will give you 1-12, MM will give you 01-12, MMM will give you Jan-Dec, MMMM will give you January-December. Here are some examples:

```
from java.util import Date
now = Date() # creates a new date, for right now

# Common format for databases
print system.db.dateFormat(now, "yyyy-MM-dd H:mm:ss")

# Nice human-readable format for just the date
print system.db.dateFormat(now, "MMM d, yyyy")

# Formating just the time in am/pm style
print system.db.dateFormat("h:mm a")
```

Next...

- Working with Datasets

Working with Datasets

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 `system.dataset` library provides various functions for manipulating and creating datasets.

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 represents datasets. When you get the data property out of a Table, for example, you'll get a DataSet. The PyDataSet is a wrapper type that you can use to make DataSets more accessible in Python. You can convert between the two with `system.dataset.toPyDataSet` and `system.dataset.toDataSet`.

Working with Datasets

Watch the Video

Accessing data in a DataSet

DataSets have various properties and functions that you can access through Python.

`rowCount`
Returns the number of rows in the dataset.

`columnCount`
Returns the number of columns in the dataset.

`getColumnName(index)`
Returns the name of the column at the given index.

`getValueAt(row, column)`
Returns the value from the dataset at the given location. `column` can be either an integer or a column name, which is treated case-insensitive.

For example, you could iterate through every item in a DataSet in scripting like this:

```
# Pull the dataset property off a Table component
data = event.source.getComponent("Table").data

for row in range(data.rowCount):
   for col in range(data.columnCount):
      print data.getValueAt(row, col)
```

or you could find specific values from each row in a DataSet like this:

```
# Pull the dataset property off a Table component
data = event.source.getComponent("Table").data

for row in range(data.rowCount):
   temp = data.getValueAt(row, "Temperature")
   speed = data.getValueAt(row, "Speed")
   print temp, speed
```

Accessing data in a PyDataSet

You can convert a dataset to a PyDataSet, which lets you use it more like a Python sequence. You don't have to do this, its purely a convenience. A PyDataSet is like a list of dictionaries, and so it can use the normal for loop syntax. These examples are equivalent to the examples above.

Iterating through a PyDataSet

```
# Pull the dataset property off a Table component
data = event.source.getComponent("Table").data

# Convert to a PyDataSet
pds = system.dataset.toPyDataSet(data)

for row in pds:
   for value in row:
      print value
```

Finding specific values in a PyDataSet

```
# Pull the dataset property off a Table component
data = event.source.getComponent("Table").data

# Convert to a PyDataSet
pds = system.dataset.toPyDataSet(data)

for row in pds:
   temp = row["Temperature"]
   speed = row["Speed"]
   print temp, speed
```

Accessing specific elements in a PyDataSet

```
# Pull the dataset property off a Table component
data = event.source.getComponent("Table").data

# Convert to PyDataSet
pds = system.dataset.toPyDataSet(data)

# Grab the first item of the first row
value = pds[0][0]
print value
```

Altering Datasets

Technically, you cannot alter a dataset. Datasets are immutable, meaning they cannot change. You can, however, create new datasets. So to change a dataset, you really create a new one and then replace the old one with the new one. Because this is so common, there are special functions under `system.dataset` that are designed for this.

You can use the following functions to create datasets that are altered version of existing datasets:

- `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 would set the **Quantity** column in the selected row of a table to 15.8:

```
table = event.source.parent.getComponent("Table")
selRow = table.selectedRow
if selRow != -1:
   # Create a new dataset
   newData = system.dataset.setValue(table.data, selRow, "Quantity", 15.8)

   # Replace the Table's data property with the new dataset
   table.data = newData
```

Creating Datasets

Sometimes you'll want to create a new dataset from scratch. This can be easily done with the `system.dataset.toDataSet` function. All it needs are the column headers and a list of the rows in the dataset. Each row must have the same number of elements as the header list.

For example, the following code would create a dataset that contained some information about US cities:

```
headers = ["City", "Population", "Timezone", "GMTOffset"]
data = []

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])

cities = system.dataset.toDataSet(headers, data)
```

Next...

- Exporting Data to CSV

Exporting Data to CSV

You can export a dataset from a query or table to a CSV file.

To export a dataset to a CSV file

1. Identify the dataset that you want to export to a CSV file.
   The source of the dataset is irrelevant because Ignition will treat all datasets the same.

2. Create a Button on the window.

3. Use the system.dataset.exportCSV scripting function to export the dataset to a CSV file.

INDUCTIVE UNIVERSITY

Exporting Data to CSV

Watch the Video

Next...

- Component Extension Functions

Component Extension Functions

Some advanced components can be customized by using the extension functions. These functions will be called by the component itself when appropriate. For example, the Power Table component exposes an extension function called `getBackgroundColorAt()`. By implementing this function, you can control the background color of each cell of the table component using scripting.

Extension functions allow you in a loose sense to create a custom "sub-class" of the base component type, from an object-oriented point of view. Your sub-class can then override/implement parts of the functionality of the component itself, in python. This is a very powerful, if somewhat advanced, way of configuring the component. Each component extension function comes with its own documentation built-into the function's default implementation using a standard Python "doc-string". You will find that you are unable to edit the function's signature or docstring. Changing the method's signature (arguments or function name) would prevent the component from calling it correctly. Changing the docstring could be misleading or confusing as you'd lose the documentation for how your implementation of the function should work.

Following Python object-oriented methodology, you'll notice that each extension function's first argument is called "self". That is because these are methods belong to the component's class itself. That is, they are instance methods. The value of `self` will always be the component itself. Notice that this is different than component event handler scripts. In those scripts, you are given an `event` object in your scope. The component that fired the event will be under `event.source`. When you write an extension function, there is no `event` object. The component is given to you as the `self` object instead.

Component Extension Functions

Watch the Video

Next...

- Component Custom Methods

Component Custom Methods

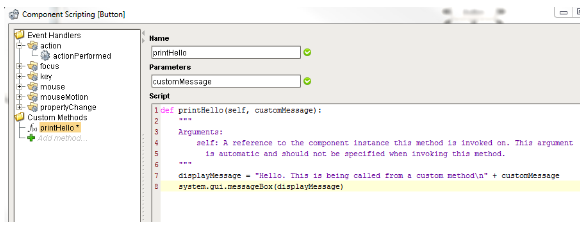
INDUCTIVE UNIVERSITY

Component Custom Methods

Watch the Video

Custom methods can be added to components and templates. Custom methods allow scripts to be defined on a component, and call the same script from another event.

In the image below, a Button component has a custom method called printHello. The method takes a single parameter called customMessage, concatenates the value of customMessage with a static string, and calls system.gui.messageBox to show the message.


Component Scripting [Button]

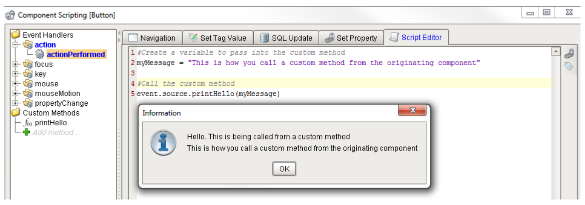
Name: printHello

Parameters: customMessage

Script

```
def printHello(self, customMessage):
    """
    Arguments:
        self: A reference to the component instance this method is invoked on. This argument
          is automatic and should not be specified when invoking this method.
    """
    displayMessage = "Hello. This is being called from a custom method\n" + customMessage
    system.gui.messageBox(displayMessage)
```

Calling a custom method requires a **path to the component where the method resides**, as well as the name of the method. To call the printHello method from the actionPerformed event on the same button, the script would look like the following:

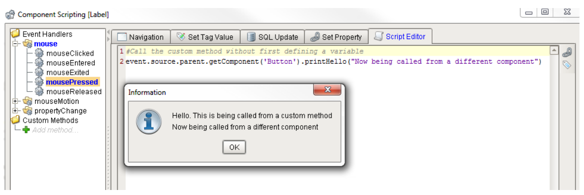

Component Scripting [Button]

```
#Create a variable to pass into the custom method
myMessage = "This is how you call a custom method from the originating component"

#Call the custom method
event.source.printHello(myMessage)
```

Information

Hello. This is being called from a custom method
This is how you call a custom method from the originating component

Custom methods can be called from scripts on other events and components. A script on a Label component located in the same container as the Button can call the printHello method.


Component Scripting [Label]

```
#Call the custom method without first defining a variable
event.source.parent.getComponent('Button').printHello("Now being called from a different component")
```

Information

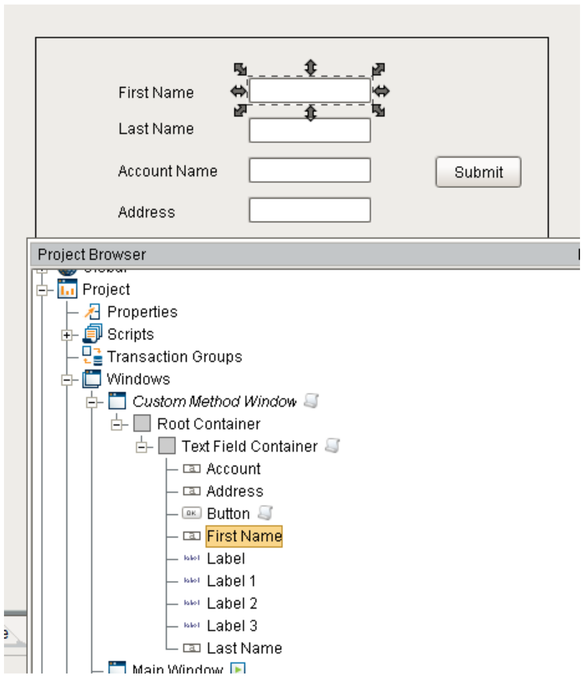
Hello. This is being called from a custom method
Now being called from a different component

Example - Error Checking User Input

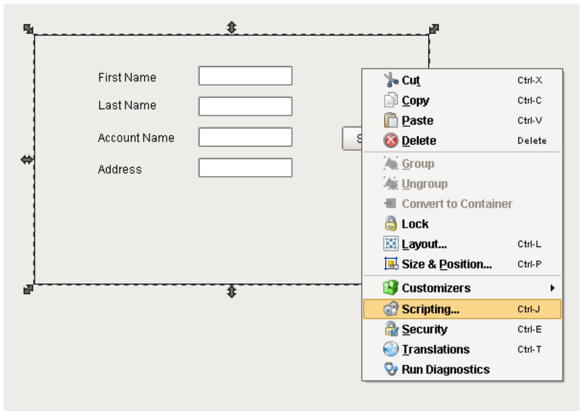
The following example will create a custom method that reads input from several components, and checks the values to make sure valid data was entered.

Step 1 - Creating the Custom Method

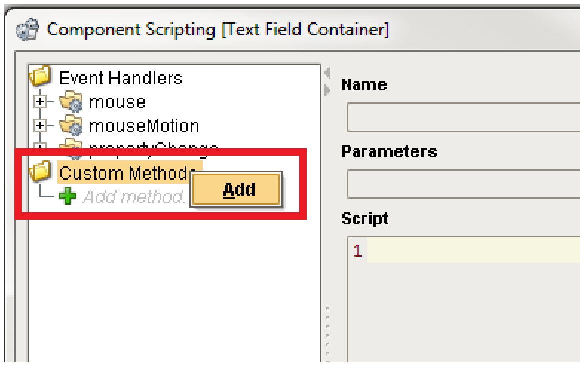
The image below shows a container named Text Field Container. Inside are four Text Field components can enter data into. The container also has a Button component that will read the values from each component.



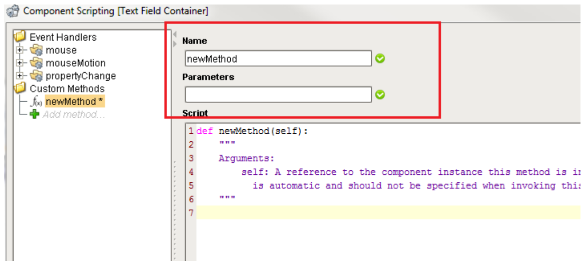
The error checking method will be placed on Text Field Container since all of the components that will be checked are inside this container. To create a custom method, right-click the Text Field Container component, and select Scripting



Right-click on Custom Methods, or double-clicking the Add method. green plus button will also create a new custom method.



A new custom method will appear. The new method can be renamed, and parameter names for the method can be defined.



Name: newMethod

Parameters:

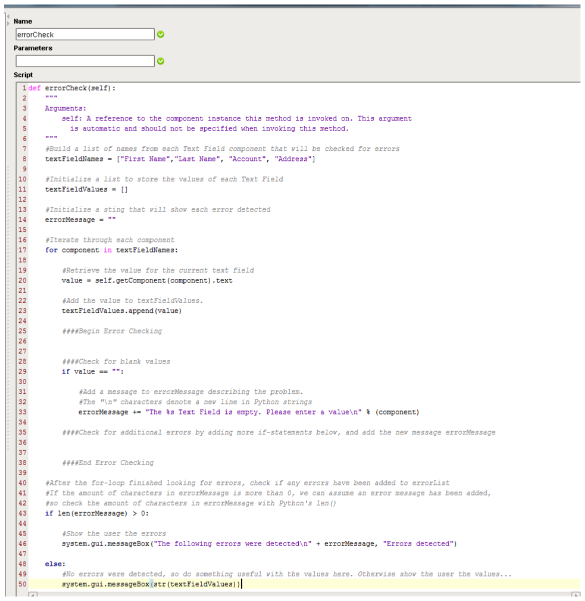
Script

```
def newMethod(self):
    """
    Arguments:
        self: A reference to the component instance this method is i
          is automatic and should not be specified when invoking thi
    """
```

Rename the method to errorChecking. This method will not accept parameters, so leave the Parameters Text Field blank. Expand and copy the example below and paste it into the Script text area. Make sure the indentation matches the image below.

errorCheck

```
        #Build a list of names from each Text Field component that will be
checked for errors
        textFieldNames = ["First Name","Last Name", "Account", "Address"]

        #Initialize a list to store the values of each Text Field
        textFieldValues = []

        #Initialize a sting that will show each error detected
        errorMessage = ""

        #Iterate through each component
        for component in textFieldNames:

                #Retrieve the value for the current text field
                value = self.getComponent(component).text

                #Add the value to textFieldValues.
                textFieldValues.append(value)

                ####Begin Error Checking


                ####Check for blank values
                if value == "":

                        #Add a message to errorMessage describing the
problem.
                        #The "\n" characters denote a new line in Python
strings
                        errorMessage += "The %s Text Field is empty.
Please enter a value\n" % (component)

                ####Check for additional errors by adding more if-
statements below, and add the new message errorMessage


                ####End Error Checking

        #After the for-loop finished looking for errors, check if any
errors have been added to errorList
        #If the amount of characters in errorMessage is more than 0, we
can assume an error message has been added,
        #so check the amount of characters in errorMessage with Python's
len()
        if len(errorMessage) > 0:
                #Show the user the errors
                system.gui.messageBox("The following errors were
detected\n" + errorMessage, "Errors detected")

        else:
                #No errors were detected, so do something useful with the
values here. Otherwise show the user the values...
                system.gui.messageBox(str(textFieldValues))
```



Name: errorCheck

Parameters:

Script

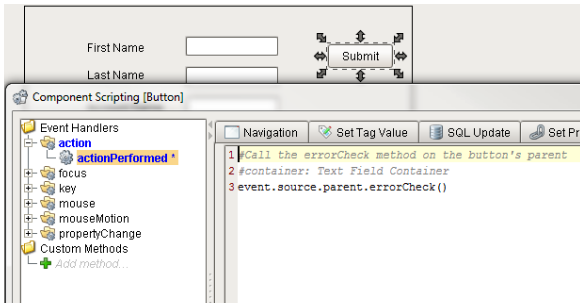
```
def errorCheck(self):
    """
    Arguments:
        self: A reference to the component instance this method is invoked on. This argument
          is automatic and should not be specified when invoking this method.
    """
    #Build a list of names from each Text Field component that will be checked for errors
    textFieldNames = ["First Name","Last Name", "Account", "Address"]

    #Initialize a list to store the values of each Text Field
    textFieldValues = []

    #Initialize a sting that will show each error detected
    errorMessage = ""

    #Iterate through each component
    for component in textFieldNames:

        #Retrieve the value for the current text field
        value = self.getComponent(component).text

        #Add the value to textFieldValues.
        textFieldValues.append(value)

        ####Begin Error Checking


        ####Check for blank values
        if value == "":

            #Add a message to errorMessage describing the problem.
            #The "\n" characters denote a new line in Python strings
            errorMessage += "The %s Text Field is empty. Please enter a value\n" % (component)

        ####Check for additional errors by adding more if-statements below, and add the new message errorMessage


        ####End Error Checking

    #After the for-loop finished looking for errors, check if any errors have been added to errorList
    #If the amount of characters in errorMessage is more than 0, we can assume an error message has been added,
    #so check the amount of characters in errorMessage with Python's len()
    if len(errorMessage) > 0:

        #Show the user the errors
        system.gui.messageBox("The following errors were detected\n" + errorMessage, "Errors detected")

    else:
        #No errors were detected, so do something useful with the values here. Otherwise show the user the values...
        system.gui.messageBox(str(textFieldValues))
```

Step 2 - Calling the Custom Method

Now that a custom method had been created, a script needs to call the method. Add a script to the button in Text Field Container on the actionPerformed event.

Invoke errorCheck

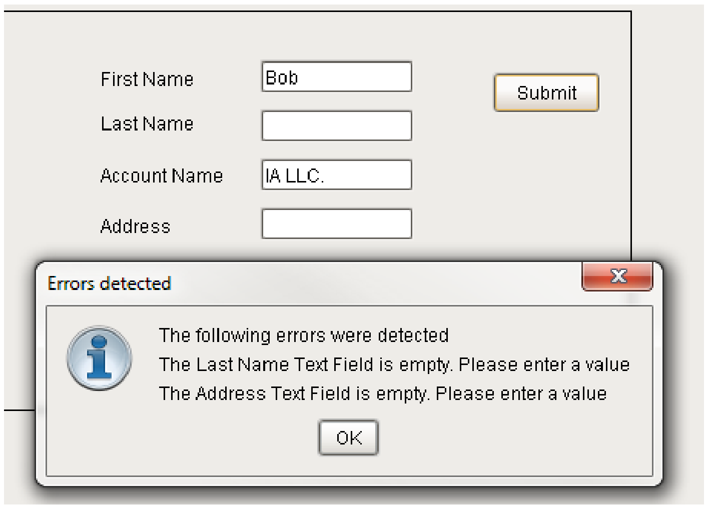
```
#Call the errorCheck method on the button's parent
#container: Text Field Container
event.source.parent.errorCheck()
```



```
#Call the errorCheck method on the button's parent
#container: Text Field Container
event.source.parent.errorCheck()
```

Step 3 - Testing the Custom Method

Click OK to close the Component Scripting window, save the project, and put the designer into preview mode. Add some data to some of the text fields, but leave at least one blank. Clicking on the button will display which text fields were empty. The image below shows the Last Name and Address text fields are empty, and the script detected this.



The following errors were detected
The Last Name Text Field is empty. Please enter a value
The Address Text Field is empty. Please enter a value

Now any script that has access to this window can check the text fields for errors. This example could be expanded to look for additional errors, such as checking for prefixes in some or all of the values.

Next...

- Exporting Data to CSV

Advanced Scripting

There are many places scripting can be used in Ignition, there are Gateway and Client scripts, startup, shutdown, time scripts and more.

Script Modules

A project's Script Modules are a global library of scripts that can be called from anywhere within the scope of a project. These scripts are organized as named modules that all live under the `app` module. To open the Script Module Editor double-click on the **Configuration > Script Modules** node in the **Project Browser** or navigate to the **Project > Script Modules** menu.

Rule of Thumb: Never Copy-and-Paste a Script

If you're unsure of when to put scripts in a script module vs. embedding the script directly in an event handler, follow this simple rule. If you ever find yourself copying a script from one event handler to another, stop and refactor the script into a global script module! Then simply call your new module from the event handler. This rule will help prevent code duplication across your project, a major maintenance liability.

How to use Script Modules

To add a script module, simply right click the `script library[project]` package and click the **New Module** button. Each module is a python script that may define many functions. You may also organize modules in subpackages if you'd like. Lets suppose you added a script module named `myfuncs`, whose body was:

```
def callMe(message):
    system.gui.messageBox(message)
```

Now, anywhere in your project you can call:

```
project.myfuncs.callMe('Hello World')
```

To import System or not to import System

Frequently in Ignition, your scripts get `system` (the built-in library package in Ignition) and `shared or project` imported for you automatically.

Subsequent versions of Ignition beyond version 7 no longer need import system every time you create a new scope. However versions prior to version 7 still require an import system every time a system associated function is called inside a function. Ignition versions prior to version 7 still need the import system when system functions are called inside functions. Regardless, typing import system in any version of Ignition will result in the system library being available for use inside the function.

See also: Scripting Intro, Scope and Import

In this section ...

Gateway vs. Client Event Scripts

Your projects can execute scripts in response to a variety of events and actions that occur within the project's lifecycle. There are two major scopes for scripting: Gateway scripts and Client scripts.

Gateway Scripts

Gateway scripts execute on the Ignition Gateway, which means that they always execute in one place.

Client Scripts

Client scripts execute in the running Client, which means that they may never execute (if no clients are running), or they may execute many times. Client scripts will also execute in the Designer, but only in Preview Mode.

Because Clients are full-fledged applications, the scripts run on the computer running the Client, not on the Gateway's host server computer. This means that they don't have access to the Gateway's file system, and so on. It also means that if no clients are launched, the scripts will not be executing.

Note that these global project event scripts are not to be confused with the component event handler scrip ts.

Gateway vs Client Event Scripts

Watch the Video

Next...

- Startup Scripts

Startup Scripts

These script types are available in both Gateway and Client scopes. These scripts will be run when the project starts up or shuts down.

In the Gateway scripting scope, this means that the script will run when the Gateway starts up or is shut down, and whenever the scripting configuration changes via a Designer save action. This means that while designing, the startup and shutdown events may happen frequently.

In the Client scripting scope, these scripts run after a user successfully logs in or out, or when the client is closed.

Startup Scripts

Watch the Video

Next...

- Shutdown Scripts

Shutdown Scripts

This script type is only available in the Client scope. This is a special script that will be called when the user tries to exit or close the client, that is, shutdown the Client. This script is run with a special event variable in its namespace. When the script terminates, if `event.cancel` is 1, the shutdown will be aborted, and the client will remain open. Otherwise, the normal shutdown script will be called, and the client will close.

Example

```
if "SuperUser" not in system.security.getRoles():
        system.gui.warningBox("Exit not allowed for non-admin user.")
        event.cancel=1
```

Shutdown Scripts

Watch the Video

Next...

- Keystroke Scripts

Keystroke Scripts

Keystroke scripts are only available in the Client scope. These are scripts that run on a certain key combination. You may add as many keystroke scripts as you'd like, as long as each one has a unique key combination. The keystroke event scripts fire on the Client when a user performs a certain key stroke.

When choosing a keystroke, you may choose any number of modifiers, which are keys or mouse buttons that must be down to activate the keystroke. You can also choose whether or not the keystroke is on the pressed or released event of a keyboard key, or upon the typing of a character. Special keys like the F-keys, ESC, etc, are only available in the pressed and released actions.

INDUCTIVE UNIVERSITY

Keystroke Scripts

Watch the Video

Next...

- Tag Change Scripts

Timer Scripts

Timer scripts are available in both Gateway and Client scopes. These scripts execute periodically on a timer at a fixed delay or rate. Remember that Client timer scripts may never execute (if no clients are open) or may execute many times (once per open client). If you need scripting logic that occurs centrally, make sure you use Gateway scoped scripts.

Fixed delay or fixed rate?

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.

Shared thread or dedicated thread?

All timer scripts for a given project that choose **Run in shared thread** will all execute in the same thread. This is usually desirable, to prevent creating lots of unnecessary threads. However, if your script takes a long time to run, it will block other timer tasks on the shared thread. The rule of thumb here is that quick-running tasks should run in the shared thread, and long-running tasks should get their own thread.

INDUCTIVE UNIVERSITY

Timer Scripts

Watch the Video

Next...

Tag Change Scripts

Tag Change scripts are available in both Gateway and Client scopes. Each tag change script can be given a list of tag paths. Whenever one of these tags changes, the tag change script will execute. They will also get an initial execution whenever the scripting system starts up.

Each tag change script can be given a name for organizational purposes. To specify multiple tag for a given script, enter them one per line in the tag paths text area. To quickly import many tags, you can drag-and-drop tags from the Tag Browser window onto this text area.

These scripts receive three special variables in their namespace when they are run: `event`, `initialChange`, and `newValue`. The `intialChange` variable is a flag (0 or 1) that indicates whether or not this event is due to initially subscribing or not. The event variable is a `TagChangeEvent` object, which itself contains the properties: `tag`, `tagPath`, and `tagProperty`. The third, `newValue`, is the new value for the tag property that is subscribed. These values are objects themselves that contain a value, quality, and timestamp. The following example script should be a good starting point:

```
print "Received tag change event for %s" % event.tagPath
value = newValue.value
quality = newValue.quality
timestamp = newValue.timestamp
print "value=%s, quality=%s, timestamp=%s" %(value, quality, timestamp)
```

The TagPath object that you access via `event.tagPath` is itself a complex object. You can turn it into a string if you want the whole tag path by using the `str()` function. You can also access individual parts of the tag path. The most useful is usually the `itemName` property, which is the name of the tag represented by the path. To get the name of the tag, you can use `event.tagPath.itemName`.

INDUCTIVE UNIVERSITY

Tag Change Scripts

Watch the Video

Next...

- Tag Scripts

Tag Event Scripts

Each tag has five special events, each with their own event handler that can run a script when the event occurs similar to event handlers on components. 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 of 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 logs. Keep that in consideration when doing any testing.

INDUCTIVE UNIVERSIT

Tag Scripts

Watch the Video

Getting Started

Each tag has the option to have tag event scripts on it. When you edit a tag, you can navigate to the Tag Events section to see a list of all of the tag scripts. You can then select which event you would like to write a script for. You can even write a script for multiple events if you would like.

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:

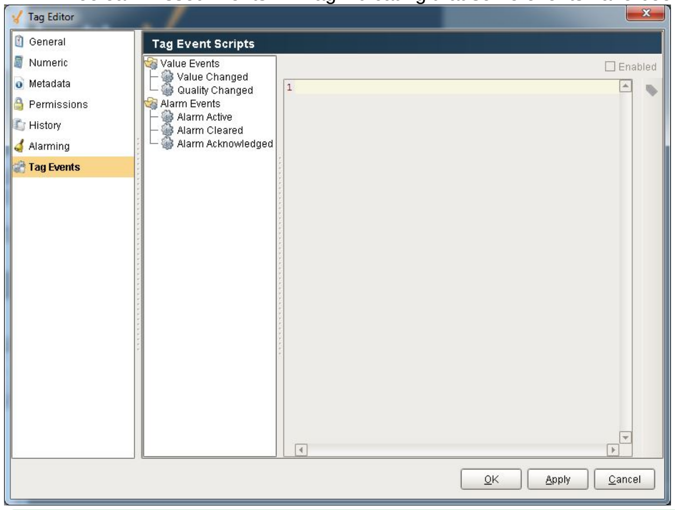
String tagPath - The full path to the tag.

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 first execution of the initial subscription.

Boolean missedEvents - A flag indicating that some events have been skipped due to an event overflow.



The currentValue and previousValue arguments are qualified values. This means that to get to the actual current value, it would look like 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.

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 first execution of the initial subscription.

Boolean missedEvents - A flag indicating that some events have been skipped due to an event overflow.

Alarm Active

The Alarm Active event fires whenever an alarm becomes active on the tag.This event has a variety of arguments available for use in the script:

String tagPath - The full path to the 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:

- id
- source
- name
- priority
- displayPath
- displayPathOrSource (the display path if it is set, otherwise the source)
- state (the current state, active/cleared + acked/unacked)
- lastEventState (the last transition)
- cleared (a boolean if the alarm is currently cleared)
- acked (a boolean if the alarm is currently acknowledged)
- shelved (a boolean if the alarm is currently acknowledged)
- 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.

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.

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.

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.write("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.read("HighestTempTag").value
if currentValue.value > highestRecordedTemp:
        system.tag.write("HighestTempTag", currentValue.value)
```

Next...

- Menubar

Menubar

The Client's menu bar is configured and customized through the Client Event Scripts dialog box. Each node in the menu bar that does not have children executes a script when the user presses it. Most commonly, these scripts will execute navigation actions, opening, or swapping a window.

See also: Navigation Strategies, Client/User Interface Properties

INDUCTIVE UNIVERSITY

Menu Bar

Watch the Video

Next...

- Shared Scripts

Shared Scripts

You can use the "shared" script library to add script functions that your whole Gateway can use.

INDUCTIVE UNIVERSITY

Shared Scripts

Watch the Video

Next...

- Project Scripts

Project Scripts

Learn to use the **`project`** script library to consolidate frequently used script logic into one place.

INDUCTIVE UNIVERSIT

Project Scripts

Watch the Video

Next...

- Script Messaging

Script Messaging

Message handler scripts run when system.util.sendMessage() is called on a remote Client or on the Gateway to send a message. Message handlers can be created for both Gateway and Client scopes. To create a message handler within a Designer, navigate to Project -> Gateway Event Scripts or Project -> Client Event Scripts. Click on the Message type on the left side, and then click on the '+' icon at the bottom to add a new script.

Threading
By default, all message handlers for a project execute on a single shared background thread. This arrangement is normally the most efficient. However, if a message handler is performing a lengthy operation, messages may become backed up while waiting for the shared thread. In order to prevent this, you can specify that a message handler runs on its own dedicated thread by setting the Threading value to "Dedicated".

WARNING: Both the shared thread and dedicated threads run in the background, meaning that if you need to update a component in the GUI (window navigation, setting and getting component properties, showing error/message popups, etc) you must wrap the code in a function and use system.util.invokeLater() to run the function as shown in the example below. An alternative is to set the threading value to "EDT". This value will execute the message handler code directly on the Event Dispatch Thread (EDT). The disadvantage of this approach is that your entire Client will freeze if the message handler script does not execute its code quickly.

Message payload
A message sent using system.util.sendMessage() may optionally contain a message payload Dictionary object to access custom message parameters. This Dictionary can be accessed from the "payload" variable as shown in the example below. If no Dictionary was passed when sending the message, then the "payload" variable will be an empty Dictionary.

Example send script

```
payloadDict = {}
payloadDict['first']="Hello"
payloadDict['second']="World"
system.util.sendMessage("ProjectX","myHandler",payloadDict)
```

Example message handler using invokeLater()

```
def handleMessage(payload):
        """
        message = "Message received by ProjectX myHandler. "
        if payload.has_key('first'):
                message += "First param=" + str(payload['first']) + ". "

        if payload.has_key('second'):
                message += "Second param=" + str(payload['second']) + ". "
        def handleMessage(theMessage=message):
                import system
                window = system.gui.getWindow("Main Window")
                rootContainer = window.getRootContainer()
                textarea = rootContainer.getComponent("receivedMessages")
                textarea.text += theMessage + "\n"
        system.util.invokeLater(handleMessage)
```

INDUCTIVE UNIVERSITY

Script Messaging

Watch the Video

Next...

- Timer Scripts

Schedule and Holiday Scripting

Schedule and Holiday Scripting

Watch the Video

Next...

Web Services & SUDS

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. Other programs interact with the service through an interface defined by a WSDL (Web Services Description Language) file. This WSDL describes how to talk with the device and what should be expected back in response. Messages to and from the web service are formatted XML and while you need very little knowledge of XML to use the SUDS library, many times a web service will return a formatted XML string that you will have to parse through manually in order to make the data presentable.

Did you know?

SUDS doesn't actually stand for anything! It's a lightweight version of SOAP, hence the name SUDS. SOAP stands for Simple Object Access Protocol, and is commonly used for Internet communication.

What if I do not have the SUDS library?

Some installations of Ignition may not have the SUDS library, but it can easily be added following the steps below:

1. Download the SUDS library: https://fedorahosted.org/suds/
2. Place the library into Ignition's installation directory: Knowledge Base - Importing 3rd party modules

When working with Web Services, the SUDS documentation and the PyDocs are a good place to start.

In this section ...

Overview & Simple Arguments

The SUDS Library

The information below assumes the SUDS library is already in Ignition's installation directory. If the example below fails because SUDS is not present, click here to learn about importing 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. 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.

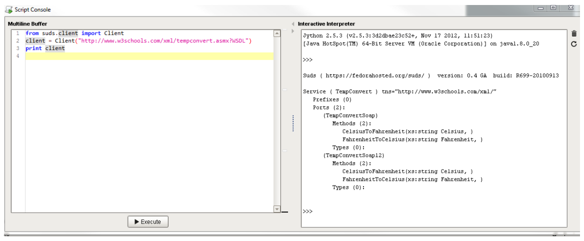
A Simple Example

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.

Example Code

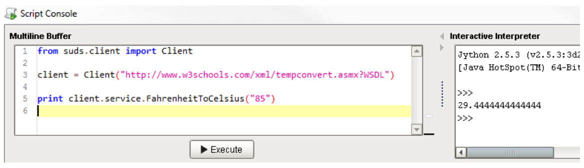
```
from suds.client import Client
client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")
print client
```



```
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 Celsius) and FahrenheitToCelsius (string Fahrenheit). 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.



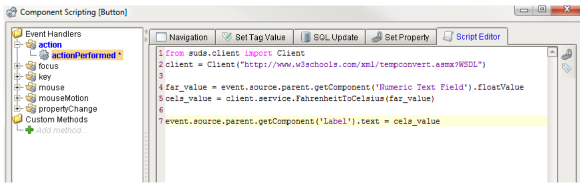
```
from suds.client import Client

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 two buttons, a numeric textbox, and a lable to a window. Then on the button to be used to calculate a Fahrenheit to Celsius calculation you would place something like the following:

F -> C Example Code:

```
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
```

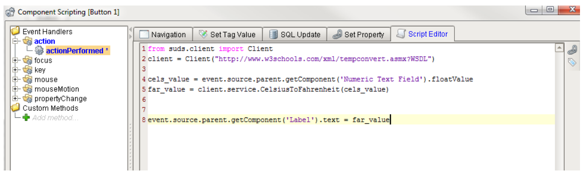


```
from suds.client import Client
client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")

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 on the second button do the opposite.

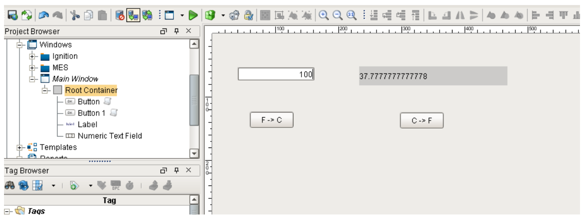
C -> F Example Code:

```
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

This example is extremely simple and rather straight forward. The main steps to remember when attempting to use the SUDS library in scripting are as follows:

```
#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)
```

Next...

- Complex Arguments

Complex Arguments

In the overview example 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. A very similar example can be found in the SUDS documentation.

Say we create and print the client object we associated with our web service in the following manner

```
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:

```
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:

```
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:

```
(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.

```
phone = client.factory.create('Phone')
phone.arecode = '916'
phone.number = '5557777'

name = client.factory.create('Name')
name.first = 'John'
name.last = 'Doe'

contact.name = name
contact.phone.append(phone)

client.service.addContact(contact)
```

After execution a new contact will have been added via the web service. There is also a way to use python dictionaries to specify complex arguments that is detailed in the SUDS documentation.

Steps to remember when using complex types:

```
#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
```

Next...

- Parsing XML Results

Parsing XML Results

It's quite easy deal with return values of method calls when they are simple none structured values like floats or integers. However it's not always the case that you will have simple single values returned from a method call. Many times web services will return values that have some sort of structure to them like a dataset. Since there is no way for the web service to know how this value should be represented in the language/environment that called the method it is common that the result will be returned in an XML formatted string. Dealing with these strings is not really part of the SUDS library but an example of XML string handling is included here for completeness. This example makes use of the xml.dom.minidom python module to parse the XML string and pull out the data.

Periodic Table Example - Using xml.dom.minidom XML parsing

One of the public web services that are available has a method for returning the elements in the periodic table in a structured string. This example will show you how you can take that string result, parse through it and then create an Ignition dataset that can be displayed in a table component. The xml.dom.minidom library provides the functionality for parsing XML strings and returning what is called an XML Document (in fact DOM stands for Document Object Model). This Document has functions that allow you access the elements within the document by specifying their respective XML tags. Most of this library is out of the scope of this document but if you have any questions about the functions being used they can be found here and here.

1. The call to the web service to get the list of elements

```
from suds.client import Client
from xml.dom.minidom import parseString
client = Client("http://www.webservicex.net/periodictable.asmx?WSDL")
elements = client.service.GetAtoms()
```

2. The elements variable will now contain a structured string of the following format. We take note of this format so we will know how to parse the string.

```
 <NewDataSet>
 <Table>
  <ElementName>Actinium</ElementName>
 </Table>
 ...
</NewDataSet>
```

3. Now we can parse the string and create a list of the element data

```
dom = parseString(xml_string)
xmlTags = dom.getElementsByTagName('ElementName')
xmlData = []
for tag in xmlTags:
        xmlData.append(tag.firstChild.data)
```

4. The xmlData list now contains all of the names of the elements from the periodic table. Now let's get separate lists for all of the atomic numbers, atomic weights, and element numbers in the same order as our element list

```
element_weights = []
element_numbers = []
element_symbols = []
for name in element_names:
        weight = (parseString(client.service.GetAtomicWeight(name)).getElementsByTagName('AtomicWeight')[0].
firstChild.data)
        number = (parseString(client.service.GetAtomicNumber(name)).getElementsByTagName('AtomicNumber')[0].
firstChild.data)
        symbol = (parseString(client.service.GetElementSymbol(name)).getElementsByTagName('Symbol')[0].
firstChild.data)
        element_weights.append(weight)
        element_numbers.append(number)
        element_symbols.append(symbol)
```

5. With our four lists we can now go about building a dataset and then use the dataset to set the data property on a table

```
headers = ["Name", "Symbol", "Weight", "Number"]
i = 0
rows = []
for name in element_names:
        oneRow = [name,element_symbols[i],element_weights[i],element_numbers[i]]
        rows.append(oneRow)
        i += 1
data = system.dataset.toDataSet(headers, rows)
event.source.parent.getComponent('Table').data = data
```

6. Put all of that together and throw it on a button's actionPerformed event inside a window with a table on it and test away! It should be noted that this script will take a couple minutes to run since there is no function to get all of the weights, numbers or symbols in one method call. Be patient.

Next...

Security

Users, Roles, and User Sources

Ignition uses the concept of role-based security throughout. Role-based security is 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. This allows users to be reassigned, removed, and added without affecting the logic of the security policy.

The users and their roles are defined 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 configuration web interface might be governed by one User Source, while the security for a project is governed by another.

There are many different 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 policies can be defined for many different parts of the system. For example:

- You can alter the roles required to log into the Gateway configuration section.
- You can define roles required to write to or even read from a Tag.
- You can define roles required to view a Component.
- You can access the security system in a script to restrict the operation of the script to authorized users.

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 know what kind of User Source is governing what, do the following steps:

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 **Configuration > 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, you go to the **Configuration > Projects** section. Look at the project in question and you can find its User Source listed there.

4. Now that you know what User Source you need to manage, you can find out what kind it is under the **Security > Users, Roles** section.

Contact Information and Schedules

User Sources are also used for other aspects of the system besides security. For example, the alarm notification system also uses users from User Sources to know who to send alarm notification messages to. For this reason, more information can be associated with a user. Contact info can be added to support the alarm notification system. A schedule can be defined on a user which can control when they are able to log in and receive alarm notification messages. Language preferences can be defined on a per-user basis to better support individual user's preferred language.

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 config interface under **Configure > Security > Users, Roles**. Click on the **manage** link next to 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.

In this section ...

Gateway Security

The Gateway's web interface can be secured in two ways:

- Password protected (see Changing Admin Password)
- Encrypted communication (see Requiring SSL)

In this section ...

Requiring 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

To turn on SSL

1. Go to the **Configure** section of the Gateway.
2. Choose **Configuration > Gateway Settings** from the menus on the left.
3. Select the check box for **Use SSL** and click on **Save Changes** at the very bottom of the page. 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 to the standard SSL port of 443. To do this, follow the directions in Setting the Port.

Enabling SSL

Enable SSL communications in Ignition to set up secure communication to the Gateway webpage as well as Client/Designer communication with the Gateway. You need to acquire and install an SSL Certificate for Ignition. It is highly recommended that you purchase an SSL certificate from a certificate authority if you turn this feature on, see Installing a Genuine SSL Certificate.

Installing a Genuine SSL Certificate

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 SSL certificates have to be purchased individually from a certificate authority, such as Verisign, GoDaddy, or Comodo.

This guide shows you how to purchase and install a real SSL certificate from a certificate authority and install it in Ignition. You'll need to be comfortable executing command-line programs to complete this guide. The examples in this guide assume a Windows environment, but the general procedure would be identical in Linux.

1. Install the Java Development Kit (JDK).
   There are some command-line tools you'll need to use to create a certificate request and to install your certificate. These tools come with the Java Development Kit (JDK). It is likely that you only have the Java Runtime Environment (JRE) installed. Go to http://java.oracle.com and click on Java SE. Download the Java SE 6 JDK and install it.
2. Open a command prompt.
   Open a command prompt from **Start > Run > cmd** and change directory into your JDK tools directory.
   ```
   cd C:\Program Files\Java\jdk1.6.0_24\bin
   ```
3. Create your KeyStore.
   SSL certificates for Ignition are stored in a file called a `KeyStore`. You'll need to create your own `KeyStore` file with a certificate in it before you can purchase the SSL certificate.
   a. Enter the following command:
      ```
      keytool -genkey -alias tomcat -keyalg RSA -keysize 2048 -keystore C:\ssl.key
      ```
      you can put the file wherever you want for now, but you should call it `ssl.key`
   b. At the prompt, enter a password. Use the password: `ignition`
   c. At the prompt, you are asked for your `first and last name`. Do not use your first and last name. This value must be one of the following for your Ignition Gateway:
      i. Fully Qualified Domain Name, for example `secure.yourdomain.com`
      ii. Public IP address, for example `202.144.8.10`
      iii. Full Server Name of your internal server, for example `scadaserver`
      iv. Private IP address, for example `192.168.0.1`
   d. At the prompt, you are asked for information about your company. Input all data accurately, as the certificate authority needs to verify this information.

   e. Lastly, at the prompt, you are asked for the password for `alias <tomcat>`. Press RETURN to use the same password as the `Key Store` file.

4. Generate a Certificate Signing Request.
   You have a `KeyStore` file named `ssl.key` at the root of your `C:\` drive (or wherever you specified it to be in step 3a).

   In your command prompt window, enter this command:
   keytool -certreq -alias tomcat -file C:\csr.txt -keystore C:\ssl.key

   It will prompt you for the `KeyStore` password (`ignition`). You now have a certificate request file at `C:\csr.txt`

5. Buy the SSL certificate.
   Now you need to get your SSL certificate signed by a certificate authority. When you go to a certificate authority (Verisign, GoDaddy, Comodo, and so on), they'll ask for your CSR, which is the `csr.txt` file that you created in step 4. Typically they'll ask you to paste your CSR into their web form. Open `csr.txt` in notepad, and copy-and-paste it into the certificate authority's form.
   If prompted what software generated the CSR, choose Tomcat or Java. After the certificate authority has processed your payment and reviewed your CSR, they will send you your certificate via email.

6. Install the SSL certificate.
   After your SSL certificate has been emailed to you, follow the instructions provided for installing the certificate into a Java KeyStore. Your certificate authority provides these instructions. The following is the procedure for installing a Comodo SSL certificate, provided as an example:

   a. Extract the certificate files that were emailed to you, in this example they were extracted to `C:\cert`

   b. Install the root certificate with the following command:
   ```
   keytool -import -trustcacerts -alias root -file C:\cert\AddTrustExternalCARoot.crt -keystore C:
   \ssl.key
   ```

   c. Install the COMODO intermediate certificate:
   ```
   keytool -import -trustcacerts -alias INTER -file C:\cert\COMODOHigh-AssuranceSecureServerCA.crt -
   keystore C:\ssl.key
   ```

   d. Install your server's certificate:
   ```
   keytool -import -trustcacerts -alias tomcat -file C:\cert\192_168_1_7.crt -keystore C:\ssl.key
   ```

7. Replace Ignition's default `KeyStore`.
   You now have a `KeyStore` file at `C:\ssl.key` that holds your SSL certificate. The certificate alias is **`tomcat`** and the password is **`ignition`**. You can now replace the `KeyStore` file that ships with Ignition with your file. Make a backup of the file at `C:\Program Files\Inductive Automation\Ignition\tomcat\ssl.key` and replace it with your `KeyStore` file. You need to restart the Ignition service after replacing this file.
   Make sure your SSL port is allowed through your server's firewall. The default SSL port is 8043, and can be changed to the standard SSL port (443) through the Gateway Control Utility (GCU).
   If you have a redundant installation, you'll need to repeat this procedure on your backup server and buy a second certificate for it.

Next...

- Changing Admin Password

Changing Admin Password

The first step in securing your Ignition installation is to change the default username/password.

Password Protection

By default, the **Configure** section of the Gateway is password protected, and this cannot be removed. You can also optionally protect the **Status** and the **Home** sections. You can also change the roles that are required to access any of these sections under **Configuration > Gateway Settings**.

Changing admin Password

Watch the Video

Next...

- Role-based Access

Role-based Access

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. Each user can have no role or many roles.

INDUCTIVE UNIVERSITY

Role-based Access

Watch the Video

To create a role

1. Go to the **Configure** section of the Gateway.
2. Choose **Security > Users, Roles** from the menu on the left.
   The User Sources page is displayed.
3. Click on the **manage users** link for the **User Source** you want to manage.
4. Click the **Roles** tab, look for the orange arrow and click the **Add Role** link.
5. Name the role by entering it in **Role Name** and click on **Add Role** to add the role.
   The role is now available to be associated with specific users.

To assign roles to users

1. Go to the **Configure** section of the Gateway.
2. Choose **Security > Users, Roles** from the menu on the left.
   The User Sources page is displayed.
3. Click on the **manage users** link for the **User Source** you want to manage.
4. Click on the **Edit** link for the User Source you want to edit, or look for the orange arrow and click the **Add User** link.
5. Click the **Roles** tab, select which roles you want this user to have from the roles section of the user properties.
   The **Administrator** role is the only role until more roles are added, therefore no other roles will appear until they are created. Once they are created, they appear as check boxes just like the Adminstrator option.
6. Click **Save Changes**.
   The user now has the privileges associated with each role.

Next...

- Internal Authentication

Internal Authentication

Ignition's default authentication profile stores its user information internally. This video shows how to add and edit users and roles to the profile using the Gateway interface and the User Management component. The Gateway Allow User Admin setting must be enabled to make changes using the User Management component in the Designer or Client.

INDUCTIVE UNIVERSITY

Internal Authentication

Watch the Video

Internal User Source

The internal User Source is very easy to use. You get this User Source when you first installed Ignition and had the default user/password combination of admin/password. 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.

Next...

- Database Authentication

Database Authentication

Add an additional authentication profile to store user information using an existing database connection. This video addresses the internal authentication profile Failover Source and the differences between a Hard and Soft Failover mode. It also includes information about the automatically created database tables.

Database Authentication

Watch the Video

Database User Source

The database user source uses an external database connection to find its users, their passwords, and their roles. When you create a database user source, you have the option of setting it up in automatic or manual modes.

Automatic Mode

In this mode, the user source will create and manage the database tables for you. In this mode, the user source will be fully manageable from the Gateway and the Vision Clients using the user management component. You can specify a prefix for the tables that are created, but their names after the prefix are chosen by the user source.

Manual Mode

In this 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 yourself directly through the database. Examples for each of the queries are given on the user source setup page. Make sure to return the columns that are defined in each query's description.

Next...

- Active Directory Authentication

Active Directory Authentication

Let's add an additional authentication profile to fetch user information from Active Directory (AD). This video explains the Domain and LDAP settings, and using SSO (Single Sign On) and enabling it in each project. Active Directory Groups are used as Ignition's roles and user-role mappings.

Active Directory User Source

The active directory User Source will communicate with a Microsoft Active Directory server through the LDAP protocol. Administration of the users and roles must be done through Active Directory's management tools. This User Source is a good choice when integration with a corporate authentication scheme is a requirement.

To set up 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 for authentication for when it queries the list of roles.

This style of User Source is not manageable within Ignition. You'll need to administer the users through Active Directory itself.

Next...

- AD Database Hybrid

AD Database Hybrid

Add an additional authentication profile to fetch the user list from Active Directory, but store all role information in a database. This authentication profile stores users, roles, and mappings in the database similar to the Database Authentication profile, but validates against passwords stored in Active Directory.

INDUCTIVE UNIVERSITY

AD Database Hybrid

Watch the Video

AD/Database User Source

The active directory/database hybrid profile type is a combination of the active directory User Source and the database (manual mode) User Source. This means that for any username/password combination, active directory gets to decide whether that user is a valid user, and if they are considered valid, then the other information for that user (roles, contact info, and so on) are retrieved from an external database connection.

Just like the AD/Internal User Source, this is very handy for projects that are required to integrate with IT's centrally managed security, but negotiating the management of roles with IT would be too cumbersome. This 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.

Next...

- AD Internal Hybrid

AD Internal Hybrid

Let's add an additional authentication profile to fetch the user list from Active Directory, but store all role information internally in Ignition. This authentication profile stores users, roles, and mappings in Ignition similar to the Internal Authentication profile, but validates against passwords stored in Active Directory.

INDUCTIVE UNIVERSITY

AD Internal Hybrid

Watch the Video

AD/Internal User Source

The active directory/internal hybrid 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, all of this information is then stored as if it were an internal User Source.

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 I.T. department, who typically control 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. Users cannot be added or removed. Their usernames and passwords cannot be changed. This is because this information resides in Active Directory, not within Ignition. Other information, such as a user's roles, contact info, schedule, and so on, is manageable.

Next...

- Verifying an Authentication Profile

Verifying an Authentication Profile

You can verify that a user exists in a given authentication profile and if the password is correct.

To verify a user

1. Go to the **Configure** section of the Gateway.
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.
4. In the resulting menu, choose the **Profile**, **Username**, enter the **Password**, and click **Test Login**.
   Ignition will test the credentials then display the results of the validity test.

Verifying an Authentication Profile

Watch the Video

Next...

- Locking the Gateway

Locking the Gateway

You can set up security for the Gateway webpage sections and Designer. The Gateway can use any existing authentication profile independent of any projects. Each of the Status, Home, and Configure pages can be restricted by role. Outside of logging into a Client, the Designer as well can require a special role.

INDUCTIVE UNIVERSIT

Locking the Gateway

Watch the Video

To setup security for the Gateway

1. Go to the **Configure** section of the Gateway.
2. Choose **Configuration > Gateway Settings** from the menu on the left.
   The **Gateway Settings** page is displayed.
3. Enter the roles the user must have in order to access the **Gateway Config Roles**, **Status Page Roles**, **Home Page Roles**, and **Designer Roles**.
   Each option can accept any number of roles as long as they are separated by commas. Also, if the option is blank, any user with any role can log in. The **Global Resource Protection** will shelter the Global Resources from edits by users with roles other than what is stated here.

Next...

Project Security in Client

You configure security in the Client on each component, window, or tag. Or you may simply want to make the entire Client read-only without having to setup security on each component.

In this section ...

Project Security

You can change the Authentication profile used by a project and add the required user roles for a Client. A user must have at least one of the roles to open the Client.

INDUCTIVE UNIVERSITY

Project Security

Watch the Video

To add a new role for a Client

1. From the Designer, go to **Project > Properties** and go to the **General** section.
2. In the Project Properties window, under **Security Settings**, select the User Source from the drop-down menu.
3. Enter the **Required Client Roles** the Client will have to have in order to access the Client.
4. Click **OK** to save the settings.

Next...

- Tag Security

Tag Security

You can add read/write security to individual tags through the Designer. Custom Access Rights must be set to use the Role-based permissions.

To set the access rights to a tag

1. Edit a tag by double-clicking on the tag.
2. In the **General** tab, from the **Access Rights** drop-down, select **Custom**, and click **OK**.
3. Go to the **Permissions** tab, from the **Add a Role** drop-down select the role, and click the **Add** button.
4. Change each roles **Access** rights by selecting the **Read/Write** or **Read Only** options.
5. Click **OK**.
   Now the user must have appropriate permissions to write to interact with the tag.

INDUCTIVE UNIVERSITY

Tag Security

Watch the Video

Next...

- Component and Window Security

Component and Window Security

Component security works on the component, group, container, and window levels. Also, each one of these has special categories of security that help with tuning security to various design considerations.

Component and Window Security

Watch the Video

To change the security settings on a component

1. Right-click on a component and select **Security**.
   The Security Settings panel is displayed.
2. Deselect the **Inherit Permissions** check box.
3. Select the role that the user must have. If the roles do not appear, right-click the area and select **Refresh**.
4. In the **Restrictions** section, select the option that will enact if the user does not have the selected role.
5. Keep the panel or close it, the security settings are immediate.

Next...

- Securing Event Handlers

Securing Event Handlers

You can set up security in an event script using the `system.security.getRoles` function, and use the function in any script in Ignition that runs in a Client.

To ensure the person invoking the script has sufficient privilages, use the following code:

```
if u'Administrator'in system.security.getRoles():
        #This part of the script will run if the user has the correct
priviliges. For example:"
        print "this script will run if the user has a the administrator
role."
else:
        system.gui.errorBox('Insufficient security privileges.')
```

If the person does not have sufficient privilages, the script will show an error box. The script can be embeded into any event handler.

Securing Event Handlers

Watch the Video

Next...

- Setting Client Read-Only

Setting Client Read-Only

There are times when it is best to open a Client in a read-only mode to eliminate the possibility that a Client will affect a device or database. The Client event start up script to set the Client mode to read-only is an easy way to accomplish this. Similar to the buttons in the Designer, this function can be used to set Disconnected, Read Only, and Read/Write modes in any script in Ignition that runs in a Client.

To set the Client to be Read-Only

1. From the Designer, go to **Project Browser > Scripts > Client Event Scripts**.
   The Client Event Scripts window is displayed.
2. In the **Startup** script enter this code: `system.util.setConnectionMode(2)` where 2 means read-only.
3. Click **OK**.
   The start-up script will run the next time a user opens the Client, resulting in the Client being read-only.

Setting Client Read-Only

Watch the Video

Next...

- Custom Security

Custom Security

Normally, security is handled according to the users roles, however sometimes it is important to control access by the location of the Client computer. For example, Clients with certain IP addresses can have a certain level of access, but Clients from computers with different IP addresses can have limited access. Likewise, the computer's host name is also a option.

This practice makes use of the system tags located in these location:

```
[System]Client/Network/Hostname
[System]Client/Network/IPAddress
```

Each Client will have 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.

A Client startup script can make the comparison and make the appropriate decision. For example, it can set the connection state to read-only with the following function:

```
system.util.setConnectionState(2)
```

Or it can prevent a Client from logging in with the following script:

```
system.security.logout()
```

Next...

INDUCTIVE UNIVERSITY

Custom Security

Watch the Video

Project Security in Designer and Gateway

You can place security on Ignition itself. The security placed on Ignition Gateway Configure page, affects how people can login to the Configure page, Designer, Home page, and Status page.

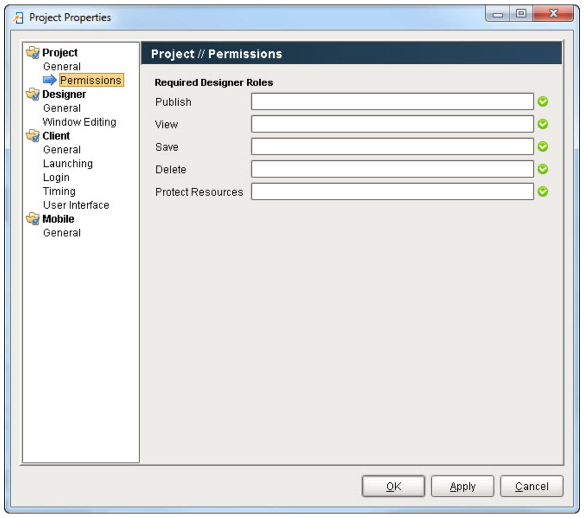
Controlling who Logs into a Project

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.

To access the Required Designer Roles property

1. In the Designer, choose **Project > Properties**.
   The **Project Properties** window is displayed.

2. Go to **Project / Permissions** page.
   In the text boxes on this page, enter a comma-separated list of role names that are required to access the project. If a user doesn't have the required role for any actions such as View, Save, or Delete, they cannot perform those actions in the project.



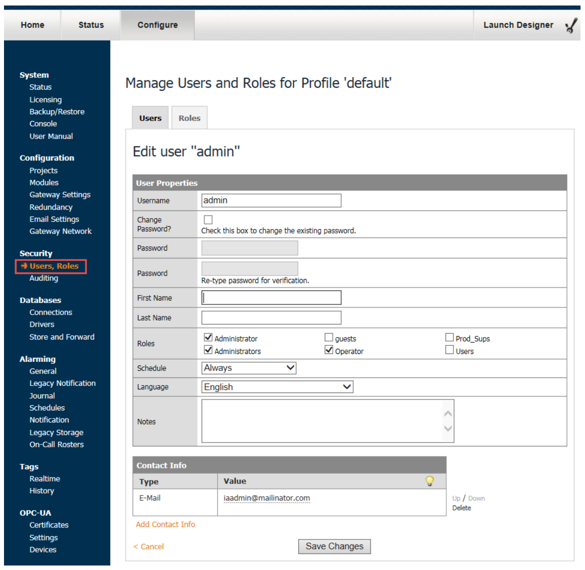
Ignition and Security

Ignition uses what's known as role-based security. Logging in to the Gateway, logging in to a project, access to windows in a project, any type of security configuration relies on users and their associated roles. Users and all their associated roles are stored in user sources that you configure in the Ignition Gateway. See the Security in Ignition section for a quick explanation of how security in Ignition works.

After finishing the initial installation of Ignition a default user source is set up automatically for you to use. It is an internal profile that cannot be deleted but can be modified to include more users and roles.

The default user source

You can manage the default user source by navigating to the Configure > Security > Users, Roles section of the Gateway. The manage users link will allow you to add new users, modify roles and passwords for existing users, remove users, and add/remove roles from the user source. Choosing to edit a user will bring you to the following page allowing you to make any necessary changes to that user.



Manage Users and Roles for Profile 'default'

Edit user "admin"

User Properties	
Username	admin
Change Password?	☐ Check this box to change the existing password.
Password	
Password	Re-type password for verification.
First Name	
Last Name	
Roles	☑ Administrator ☐ guests ☐ Prod_Sups ☑ Administrators ☑ Operator ☐ Users
Schedule	Always
Language	English
Notes	

Contact Info

Type	Value
E-Mail	iaadmin@mailinator.com

Types of user sources

Internal user source - This is a simple to setup, internally managed user source. All information in this type of profile is stored in the internal database used by Ignition. These types of profiles can only be managed from the Ignition Gateway, so they are not ideal for situations where you wish to create an user source that is modifiable at project runtime.

Database user source - All roles, users, and passwords are stored in a database that you specify. Managing users is done via direct interaction with the database so this kind of profile is best suited for managing users and roles during your project at runtime.

Active Directory user source - Roles and users are managed by Active Directory.

AD/Internal user source - Users managed by Active Directory and roles stored internally.

AD/Database user source - Users managed by Active Directory and roles stored in an external

In this section ...

Project Permissions

Permissions are set on a per-project basis. This pertains to not only if a user can open and log into a project, but also what a person can do to the project that exists in the Gateway.

To set the permissions and the Designer roles

1. In the Designer, select **Project > Properties** to access the project's properties.
2. In the **Project Properties** window, go to the **Project > Permissions** section.
3. Enter the the different options of the **Required Designer Roles**.
   There are five options and each accepts a list of roles separated by a comma.
   The following table describes each of these five options:

Option	Affect
Publish	User must have at least one of these roles to publish the project.
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.

INDUCTIVE UNIVERSITY

Project Permissions

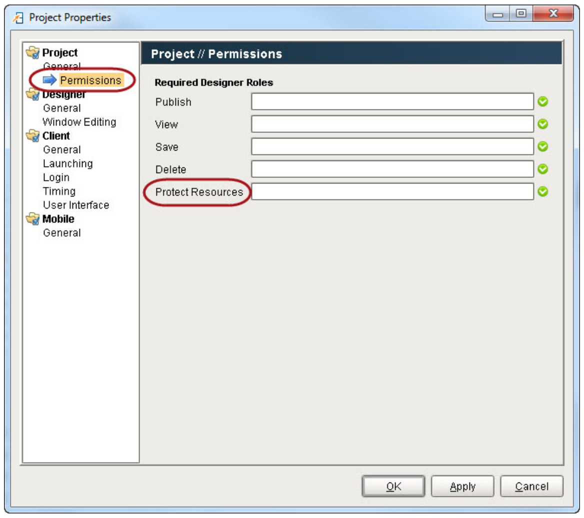
Watch the Video

Next...

- Locking Project Resources

Locking Project Resources

You can lock individual project-level resources in the Ignition Designer using **Project > Properties** and the **Project Properties** window and adding a role to the **Protect Resources** project setting.



Project // Permissions

Required Designer Roles

Publish

View

Save

Delete

Protect Resources

Role-based access

Security is configured using roles. This simple concept just means that instead of granting or revoking privilege based on user, you do so based upon the more abstract concept of a role, and then you assign users to belong to one or more roles.

The maintenance ramifications of this separation are fairly obvious - you define your security based upon the process, not the people. Ideally, the process remains constant even if the cast of characters changes. As people are hired, transferred, promoted, fired, etc, the security management simply becomes the re-assigning of roles, not the re-designing of your project.

Project Required Roles

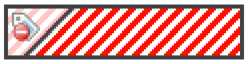
The coarsest level of security for a Vision project is the project's **Required Roles** property. This is a list of roles that the user must have all of in order to log into the Client.

See also: Project/General Properties, 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 all windows in the project, as opposed to configuring component security on each component that displays or controls that tag.

If a user opens a window that has components that are bound to a tag that the user doesn't have clearance to read or write to, the component will get a forbidden overlay.



See also: Quality Overlays, Permission Properties

Component Security

Each window and component can define its own security settings. These settings determine who can see and/or use the component. To define security for a component, right click on it and choose **Component Security**. Here you can choose to implement a security policy different than that of your parent.

In the Client, if the user does not match the role filter that you define, the component will be disabled or hidden and disabled. If a user with higher privileges logs in, the component will be useable again.

INDUCTIVE UNIVERSITY

Locking Project Resources

Watch the Video

If you choose to disable a component, make sure that it is a component that actually does something different when it is disabled. For example, buttons and input boxes can't be used when they are disabled, but disabling a label has no effect.

Securing event handlers

Event handlers often execute logic that must be secured. The various script builders all have special security qualifiers that can be enabled. These qualifiers get translated into the generated script by accessing the user's current roles via scripting.

Example

```
if 'Administrator' in system.security.getRoles():
productCode = event.source.productCode
qty = event.source.parent.getComponent("QuantityBox").intValue
query = "UPDATE my_secure_table SET quantity=? WHERE product=?"
system.db.runPrepUpdate(query, [qty, productCode])
else:
system.gui.errorBox('Insufficient security privileges.')
```

See also: Script Builders, system.security.getRoles

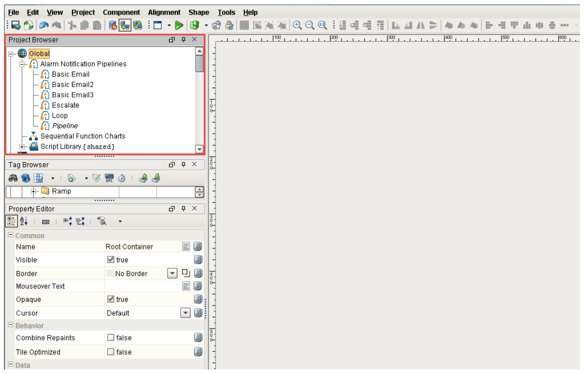
Next...

- Locking Global Resources

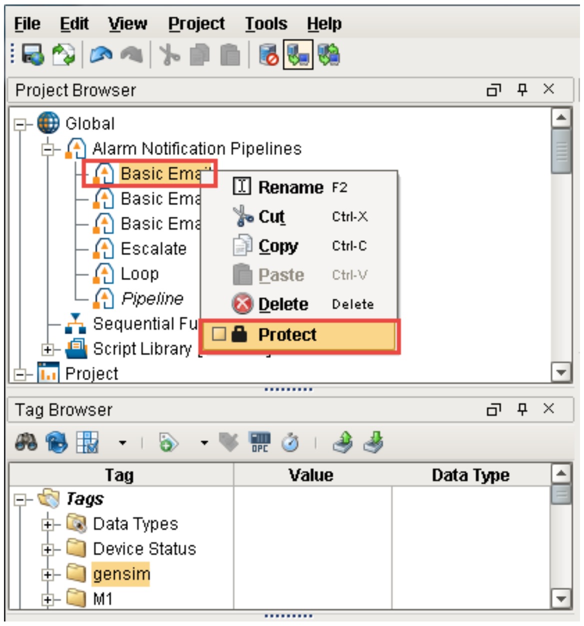
Locking Global Resources

You can lock individual global resources in Ignition using the protection option. Global Resources can be locked in 2 different locations: in the Designer and on the Gateway webpage using the Global Resource Protection setting.

To lock Global Resources in the Designer, go to the Global section of your Project Browser.



Right click on one of the objects under the Global section and select **Protect.** This example uses the Basic Email Pipeline object, but any object can be protected. To remove the protection, simply right click the object and select the **Protect** option again to unprotect it. Once an object is protected, you will see that you can still modify it.

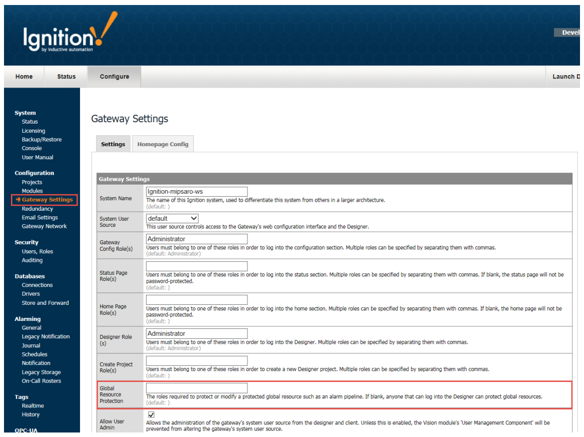


To block certain users, a role requirement must be added on the Gateway Webpage under the Configure section. On the left side under Configuration, click Gateway Settings and locate the Global Resource Protection option.

INDUCTIVE UNIVERSITY

Locking Global Resources

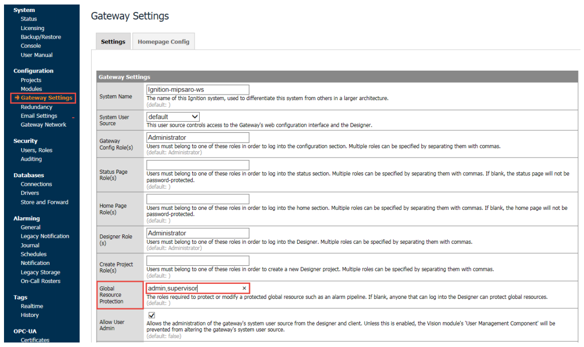
Watch the Video



Gateway Settings

Gateway Settings	
System Name	Ignition-mipsaro-ws The name of this Ignition system, used to differentiate this system from others in a larger architecture. (default:)
System User Source	default This user source controls access to the Gateway's web configuration interface and the Designer.
Gateway Config Role(s)	Administrator Users must belong to one of these roles in order to log into the configuration section. Multiple roles can be specified by separating them with commas. (default: Administrator)
Status Page Role(s)	Users must belong to one of these roles in order to log into the status section. Multiple roles can be specified by separating them with commas. If blank, the status page will not be password-protected. (default:)
Home Page Role(s)	Users must belong to one of these roles in order to log into the home section. Multiple roles can be specified by separating them with commas. If blank, the home page will not be password-protected. (default:)
Designer Role (s)	Administrator Users must belong to one of these roles in order to log into the Designer. Multiple roles can be specified by separating them with commas. (default: Administrator)
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. (default:)
Global Resource Protection	The roles required to protect or modify a protected global resource such as an alarm pipeline. If blank, anyone that can log into the Designer can protect global resources. (default:)
Allow User Admin	✓ Allows the administration of the gateway's system user source from the designer and client. Unless this is enabled, the Vision module's 'User Management Component' will be prevented from altering the gateway's system user source.

You can add or modify the roles you want in the Global Resource Protection field to protect the global resources. In this example, you can enter multiple role names separating them with commas. Once you enter the required roles, anyone without one of these roles will not be able to modify protected objects.



Gateway Settings

Gateway Settings	
System Name	Ignition-mipsaro-ws The name of this Ignition system, used to differentiate this system from others in a larger architecture. (default:)
System User Source	default This user source controls access to the Gateway's web configuration interface and the Designer.
Gateway Config Role(s)	Administrator Users must belong to one of these roles in order to log into the configuration section. Multiple roles can be specified by separating them with commas. (default: Administrator)
Status Page Role(s)	Users must belong to one of these roles in order to log into the status section. Multiple roles can be specified by separating them with commas. If blank, the status page will not be password-protected. (default:)
Home Page Role(s)	Users must belong to one of these roles in order to log into the home section. Multiple roles can be specified by separating them with commas. If blank, the home page will not be password-protected. (default:)
Designer Role (s)	Administrator Users must belong to one of these roles in order to log into the Designer. Multiple roles can be specified by separating them with commas. (default: Administrator)
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. (default:)
Global Resource Protection	admin,supervisor The roles required to protect or modify a protected global resource such as an alarm pipeline. If blank, anyone that can log into the Designer can protect global resources. (default:)
Allow User Admin	✓ Allows the administration of the gateway's system user source from the designer and client. Unless this is enabled, the Vision module's 'User Management Component' will be prevented from altering the gateway's system user source. (default: false)

Before assigning a role, typically an Administrator or the person adding Global Resource Protection needs to identify what role is required and who has that role so the proper protection is placed on global resource objects.

Next...

- Tag Provider Security

Tag Provider Security

Sometimes only certain users with certain roles can alter tags in the Designer. Tag provider security allows you to reserve the right to alter tags in the Designer to only a selected number of roles.

INDUCTIVE UNIVERSITY

Tag Provider Security

Watch the Video

To configure tag provider security

1. Go to the **Configure** section of the Gateway.
2. Choose **Tags > Realtime** from the menu on the left.
   The **Realtime Tag Providers** page is displayed.
3. Click **edit** to modify the real time Tag Provider that you want to apply the roles to.
4. In **Tag Editor Roles**, enter the roles that you want to limit the ability to edit the tags while the user is in the Designer.
5. Click on **Save Changes**.

Next...

Project Creation and Publishing

Publish Selected Resources

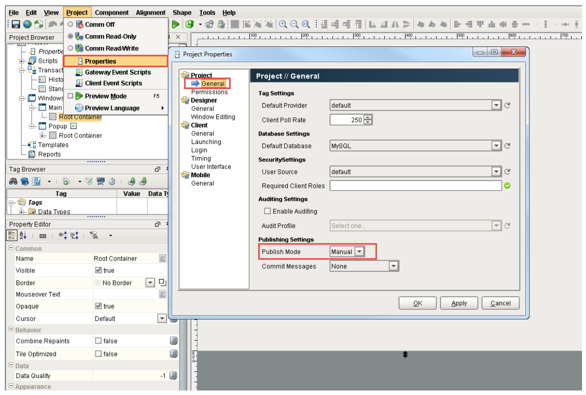
It's possible to publish individual resources on a project as they are finished instead of publishing the entire project. This is helpful if you want to get resource updates to users or into production quicker rather than waiting for the entire project to be published.

INDUCTIVE UNIVERSITY

Project Creation and Publishing

Watch the Video

1. It's important to note that the Publish Selected option is only available in the Publish Mode when set to Manual. If it is set to Auto, you will not be able to publish your project resources individually. To check, go to **Project > Properties**. Under **Project > General**, set the **Publish Mode** to **Manual,** and press **OK.**



Project // General

Tag Settings
Default Provider: default
Client Poll Rate: 250

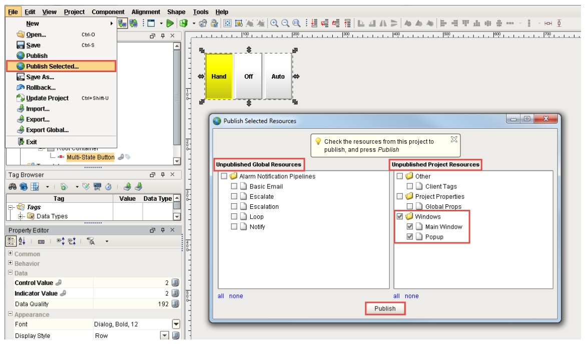
Database Settings
Default Database: MySQL

SecuritySettings
User Source: default
Required Client Roles:

Auditing Settings
☐ Enable Auditing
Audit Profile: Select one...

Publishing Settings
Publish Mode: Manual
Commit Messages: None

2. Go to your **Designer** and open a window. To publish selected resources, go to **File > Published Selected**. Here you can specify which Global and Project resources you would like to publish. Select the resources to publish, and press **Publish**.

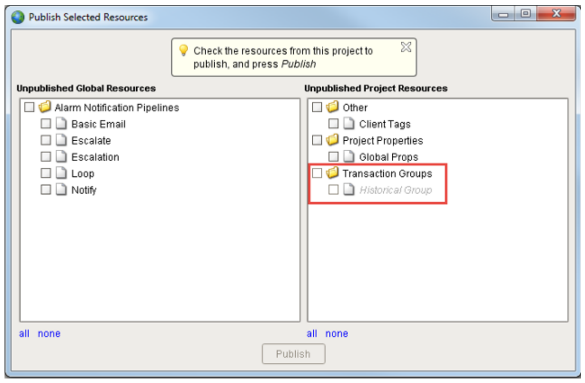


Check the resources from this project to publish, and press *Publish*

Unpublished Global Resources
☐ Alarm Notification Pipelines
☐ Basic Email
☐ Escalate
☐ Escalation
☐ Loop
☐ Notify

Unpublished Project Resources
☐ Other
☐ Client Tags
☐ Project Properties
☐ Global Props
☑ Windows
☑ Main Window
☑ Popup

3. In this example, you will notice that the Historical Group was created as a new resource in the project, but is grayed out. This is because the project was not saved. You can only publish items that were saved in your project. So keep in mind, when you see that an item is grayed out, go back and save your project. Then you can select your item and publish it.



Check the resources from this project to publish, and press *Publish*

Unpublished Global Resources
☐ Alarm Notification Pipelines
☐ Basic Email
☐ Escalate
☐ Escalation
☐ Loop
☐ Notify

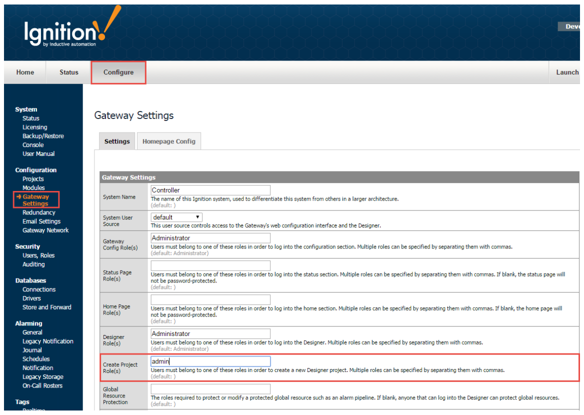
Unpublished Project Resources
☐ Other
☐ Client Tags
☐ Project Properties
☐ Global Props
☐ Transaction Groups
☐ *Historical Group*

As you can see, using the Publish Selected option gives you another way of publishing your project resource updates, and can even get them to your users more rapidly.

Restrict Project Creation

There is a **Create Project Role(s)** setting that allows users with the specified role to create new projects on the Gateway. This is ideal for production systems where you don't want other users creating new projects.

1. Go to **Configure > Gateway Settings**, and scroll down to **Create Project Role(s)**.   Enter the role which permits users to create new projects in Designer. This example uses an Admin role. Click **Save Changes**.



Gateway Settings

Gateway Settings	
System Name	Controller The name of this Ignition system, used to differentiate this system from others in a larger architecture. (default:)
System User Source	default This user source controls access to the Gateway's web configuration interface and the Designer.
Gateway Config Role(s)	Administrator Users must belong to one of these roles in order to log into the configuration section. Multiple roles can be specified by separating them with commas. (default: Administrator)
Status Page Role(s)	Users must belong to one of these roles in order to log into the status section. Multiple roles can be specified by separating them with commas. If blank, the status page will not be password-protected. (default:)
Home Page Role(s)	Users must belong to one of these roles in order to log into the home section. Multiple roles can be specified by separating them with commas. If blank, the home page will not be password-protected. (default:)
Designer Role(s)	Administrator Users must belong to one of these roles in order to log into the Designer. Multiple roles can be specified by separating them with commas. (default: Administrator)
Create Project Role(s)	admin 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. (default:)
Global Resource Protection	The roles required to protect or modify a protected global resource such as an alarm pipeline. If blank, anyone that can log into the Designer can protect global resources. (default:)

Next...

Using Audit Profiles

Auditing allows you to store any audit event to a database automatically. You can see who logs in to a client, unites, and tags, who writes to the database, and more.

In this section ...

Project Auditing

Ignition supports audit logs for your project. An audit log will record to a database the interactions that your users have with tags and databases. An Ignition Gateway can have more than one audit log but a project can only have one of these audit logs. Multiple projects can feed into a single audit log. When an audit log is active on a project data is written into a database. Ignition configures the database table for you. The following table describes the audit table as it exist in the database:

Column Name	Description
AUDIT_EVENTS_ID	The id of the row.
ACTION	Brief description of the action.
ACTION_TARGET	The target of the action.
ACTION_VALUE	The value acted upon the action target.
ACTOR	The logged in user when the action occurred or a description of the system that generated the action.
ACTOR_HOST	The host computer where the action occurred.
EVENT_TIMESTAMP	The time when the action occurred.
ORIGINATING_CONTEXT	A numerical description of the origin of the originating system. **Scope** / **Value** Gateway / 1 Designer / 2 Client / 4
ORIGINATING_SYSTEM	The name of the project or system where this action occurred.
STATUS_CODE	A 32 bit integer that represents a bitmask of severity (good, bad, uncertain). The bitmask represents the status of the action recorded. For example, a tag write failing would result in a "bad" with an appropriate sub code.

Project Auditing

Watch the Video

To create an audit log

1. Go to the **Configure** 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.
4. Select **Database** and click **Next**.
5. Enter the **Name** of the audit log and the **Retention** time.
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**.

To use an audit log

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 the **Audit Profile** from the drop-down menu. If the new audit profile does not show up, click Refresh.

3. Click **OK**.

4. **Save** your Project.

Next...

- Alarm Notification Auditing

Alarm Notification Auditing

Alarm Notification profiles can be set to store information in an Audit Profile.

To store alarm notifications in an audit profile

1. Go to the **Configure** 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.
   The Edit Alarm Notification Profile page is displayed.
4. Scroll down to the **Auditing** section, and select the **Audit Profile** from the drop-down menu.
5. Click **Save Changes**.

Alarm Notification Auditing

Watch the Video

Next...

- View Audit Information

View Audit Information

Audit profiles store information into a database, you can then display that information and see what information is stored. 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 a table functions.

INDUCTIVE UNIVERSITY

View Audit Information

Watch the Video

To access the audit log with table functions

1. Drag a **Table** component on a window.
2. In the **Property Editor**, click on the binding icon from the tables **Data** property.
   The Property Binding window is displayed.
3. Select **Functions** from the various binding options.
4. For **Binding Function**, select the Audit Log from the drop-down menu.
5. Select the appropriate options from the resulting menu as well as the **Polling Mode**.
6. Click **OK**.

Next...

Localization

Localization allows you to translate text to multiple languages. The user can choose which language their Client displays. Text contained in components, that have their properties correctly mapped with localization, will change to reflect the language the user has selected. The **Language Selector** component is a dropdown list that displays the language options for these translations.

A central term database and support is built into 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.

Translation Database and Term Lookup

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. Terms are referenced in the translation database using direct string comparison. The term **Keys** belong to the "base language", which by default is English. The English key will return if no match is found for the requested language. It is possible to define an alternate "translation" for the base language. The base term may be a user-readable string, such as "Start", or a special code, such as "START_COMMAND", which would have an "alternate translation" defined for "Start."

The full translation database can be viewed and edited in the **Translation Manager** panel located in the Designer. The translation manager allows you to import and export a translation database. It also has the ability to define new languages.

Global and Component Scopes

The translation database and term lookup rely on two different scopes for term lookup: global scope and component scope.

Global Scope

A single language selector on a window has the potential to trigger translations on all windows. This way a single language selector component can exist on a navigation window and provide language translations for all components on all windows.

Component Scope

The component scoped translation terms can display special translations that may be more relevant to the given context of the translation. For example, if the English language has one word that shares meaning between context while the alternate language has two words that differentiate between the two context then the component level translation may be appropriate. If the component level translation is not configured on the component then the translation term is automatically taken from the global scope. Component level translation will reflect on all components that occur on a single window. Component level translations does not encompass terms beyond the window on which they are configured.

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.

System Considerations During Translation

Inherent operating system and Java configurations may affect Ignition's ability to provide a complete translation in certain circumstances.

For example, suppose a user has 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.

Previewing Translations

You can preview the translations in the Designer with the preview mode in several ways. Using the Language Selector Component or scripting will temporarily change the current language, but the language will revert back to the base language (which is normally English) when preview mode is disabled. To change the preferred language for preview mode, use the Preview Language menu item under the Project menu in the Designer. This selection will persist even after preview mode is disabled and will persist when the Designer is restarted. Currently, you can not define translations while displaying in a different language.

In this section ...

Introduction to Project Translation

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.

Introduction to Project Translation

Watch the Video

To choose the language for your project - Global Translation

1. Go to **Tools > Translation Manager**.
2. Add a language, for example **Spanish**.
3. Add terms and phrases to be translated.
   The translation database shares terms across all projects.
   You can use codes for the Key field of your Terms like '#stop' or '#welcome_screen_info' (the # is just for convention, it is not a special character), to help manage longer instruction sets.
4. Add a Language Selector Component to your client.
   Now your users can move easily between languages.
   There is no binding involved because the languages are compared directly against the Translation Manager database.

Component Translation

Alternatively, you can right-click on the Root Container of a window and select **Translations**. The **Translatable Terms** panel will provide all the terms on the window. This can provide a central place to manage all the translations associated with the window. The difference between the Global and the Component translations is that the Component translations are stored on the component instead of in the translation manager's database.

Next...

- Import and Export translations

Import and Export translations

Often projects are build with a single language in mind. However Ignition's Language Selector components will allow the user to translate the text of a window or a project to the language of their choice. To ensure all the terms are correctly translated in anticipation of a user using the Language Selector, it is important to leverage the import and export translation features associated with Ignition's Translation Manager.

INDUCTIVE UNIVERSITY

Import and Export Translations

Watch the Video

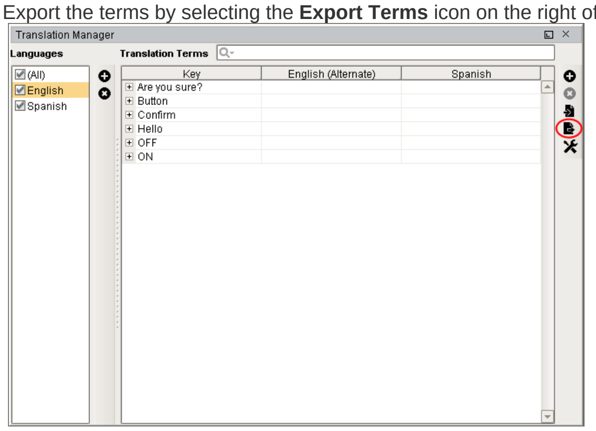
To export all the terms from a window

1. Right-click on an existing window's **Root Container** and select **Translations**.
   The **Translatable Terms** panel is displayed on top of the Property Editor.

2. Select all the available Component Terms and mark the terms for translation by clicking the Earth icon with the green plus sign. **Note:** the list of available terms are based on the components in the Root Container. If you have no available terms, try adding a label or button to the window.

3. Select **Tools > Translation Manager**.

4. Export the terms by selecting the **Export Terms** icon on the right of the **Translation Manager**.



5. Select the **Folder** location, **Base file name**, and the appropriate **Languages**.

6. In the **Format** drop-down select either **PROPERTIES** or **XML**, they both 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 the XML file can be imported back into Ignition's Translation Manager.

To import terms into Ignition's Translation Manager

1. Go to **Tools > Translation Manager**, and select the **Import Terms** icon from the right of the Translation Manager.

2. Choose the file from the **Open** window, and Click **Open**.

3. In the **Translation Manager**, choose the correct Language and choose to import all or the selected terms.
   The terms are imported into the Translation Manager.

Next...

- Switching the Current Language

Switching the Current Language

The currently displayed language can be changed in a number of ways:

- On the **project Login** page.
  The user can select their desired language when they login to the project.
- Using a **Language Selector** component in the project.
  This component displays the languages that have translations available, and changes the current language based on selection.
- With the `system.util.setLocale()` function.
  This function takes an ISO 639-1 two-letter code for the language to use (for example, "en", "fr", "de", "es", and so on).

INDUCTIVE UNIVERSITY

Switching the Current Language

Watch the Video

Next...

Tags

Tags provide a consistent data model throughout Ignition, and offer the easiest way to get up and running creating realtime status and control systems. The phrase "realtime status and control system" is sort of an industry buzz word.

Despite their low 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 in a central SQL database means that you can build widely distributed SCADA systems more easily than ever before with a high level of performance and relatively easy configuration.

Tag Configuration in the Gateway

While the goal of Tags is to create an easy yet powerful tag model, the variety of options and terminology can sometimes make configuration confusing. The goal of this section is to provide a clear overview of the Tag landscape, and provide a clear guide to the configuration of various architectures.

Note that you will use both the Gateway and the Designer for tag configuration.

- In the **Gateway**, as described in this section, you perform the tag configuration by specifying such things as the realtime/historian providers and internal/external databases. The tag configurations in the Gateway apply globally to all your projects.

- In the **Designer**, when working on your project design, you will create the different types of tags such as an OPC tags, memory tags, and expression tags. You will then set the tag properties and other functionalities such as alarming. You will also gain a working knowledge of what tags are and how they are executed.

Tag Providers and Drivers

At the highest level of 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 provider holds tags, but not every provider is a Tag driver, or a tag executor. Some tag providers simply make tags available to use, and the tag execution is performed by a different driving provider elsewhere. For example, the Database Provider is a Tag provider that simply watches a tag database stored in an external database. It does not execute tags, it only monitors the values of the tags present. Somewhere else there is a tag driver such as a Driving Datasource Provider or a legacy FactorySQL that is executing the tags and storing the values to the database.

Realtime vs. Historian Providers

As discussed above, all Tags reside in a tag provider and provide 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 Providers - Internal vs. External

The Tags "tag database", or how Tags tag configuration is stored, can take two forms. For internal tags, the configuration is stored in the Ignition Gateway. In the external form, tags are stored in a SQL database, outside of the Ignition Gateway. The two different storage mechanisms have different pro and cons, and so it is a good idea to acquaint yourself with each of them.

Internal Tags Provider

The internal Tags provider manages the tags inside the Ignition Gateway. This is the most common method and is optimized for the best scaling and performance.

It is possible to create multiple internal providers per Gateway. The tags can be exposed to other Gateways on the network through the Gateway's OPC-UA settings.

External Tags Providers

There are two types of external Tags providers in Ignition:

- Database Provider
- Database Driving Provider (provided by the SQL Bridge module)

The driving provider, as mentioned above, will execute tags as well as make available tags driven by other external drivers looking at the same database, such as other Ignition Gateways using the driving provider, or legacy FactorySQL installations.

The primary benefit of external providers is that the data is stored in a central database, and is therefore accessible to other data consuming systems that do not have the capabilities to connect to Ignition's OPC-UA server as an OPC-UA client yet still require the need to consume realtime tag information.

The negative side to external providers is that all tag values must be written to the database and then polled for change, which is a less efficient method than internal provider's method. In other words the inherent architecture of the external model does not scale as well as the internal provider model.

Configuring Realtime Tags

Realtime Tags providers are configured in the Gateway's **Configure** 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.

There are three types of **Realtime Tag Provider** that you can choose from:

- Standard Tag Provider
- Tag Consuming Provider
- Tag Driving Provider

The settings for each of these tag providers are shown next.

Standard Tag Provider

Tags are stored inside of Ignition and executed by the system.

Name	The name of the provider.
Description	The description of the provider.
Tag Editing Role (s)	Users must belong to one of these roles in order to edit this provider's tags in the Designer. Multiple roles can be specified by separating them with commas. If blank, tag editing for this provider will not be restricted in the Designer.
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 Consuming Provider

Reads tags stored in an external database and driven by a different instance. Only consumes tags provided by other systems, does not write tag values back to the database.

Name	The name of the provider.
Description	The description of the provider.
Tag Editing Role (s)	Users must belong to one of these roles in order to edit this provider's tags in the Designer. Multiple roles can be specified by separating them with commas. If blank, tag editing for this provider will not be restricted in the Designer.
Database	The database connection where the tag database resides.
Poll rate	The rate in milliseconds at which value and configuration changes will be polled.
Poll overlap	The amount of time, in milliseconds, to overlap the polling time span.

Tag Driving Provider

Stores and executes tags in an external database.

Name	The name of the provider.
Description	The description of the provider.
Tag Editing Role (s)	Users must belong to one of these roles in order to edit this provider's tags in the Designer. Multiple roles can be specified by separating them with commas. If blank, tag editing for this provider will not be restricted in the Designer.
Database	The default database connection where the tag database resides.

Driver name	The unique name of this driver. Since the tags are stored in a central database, there may be multiple providers and drivers operating on them. This name will be used to identify this driver instance, and the tags that it executes. While the driving provider will read all of the tags stored in the database, it will only execute those tags that are assigned to it.
Poll rate	The rate in milliseconds at which value and configuration changes will be polled.
Poll overlap	The amount of time to overlap polls by. If set to 0, the config scan will look for changes only since the last execution. However, on databases that do not support millisecond resolution or are performing less-than-optimally, this could result in missed changes. This setting will expand the window in order to avoid missing these changes.
Enable browsing	Allows remote browsing of the OPC servers available to this driver over TCP/IP. This allows other Gateways to remotely browse and add tags assigned to this driver into the central database.
Browse port	The port to listen on for remote connections. This port must not be in use by any other entity on the machine. Also, each driving provider that wishes to support browsing must have its own port.
Browse address	The IP address/network name that remote Gateways will use when browsing. Therefore, care must be taken that the address is available from the Gateways that will try to connect. Also, since it is used for access by remote systems, it should not be the loopback address (localhost or 127.0.0.1). The browse address and port will be stored in the Tags database so that other Gateways can easily look them up. After the settings are configured, you should immediately see the driver name in the OPC browse list for the external provider on other systems looking at the same database. **Note:** When using remote browsing, make sure that the local firewall has an exception for the port that is used to listen. Otherwise, remote machines will not be allowed to connect.

External Provider Reference

Important

The information provided here requires an understanding of Tags and how they work. It is an advanced reference to how the tables of external Tags providers are structured and an overview of the concepts of tag execution. If you are a new user, it is suggested that you read the Understanding Tags section first.

Basic Concepts and Data Flow

Tags operate through tables created in the database.

Tag Configuration Table Names and Descriptions	
sqlt_core	The core tag information table, has one entry per tag. Defines fundamental properties like data type, as well as the current value of the tag. Is monitored by the provider to determine value and configuration changes.
sqlt_meta	Provides additional properties for tags. Only consulted when tag configuration has changed.
sqlt_as	Provides alert state configuration for tags which utilize alerting.
sqlt_perm	Provides custom permission settings for tags set to use them.
Operations Table Names and Descriptions	
sqlt_sc	Contains the definitions of scan classes, which dictate how tags are executed.
sqlt_sci	Contains an entry for each scan class from sqlt_sc, for each driver currently driving tags. Used to verify that drivers are properly executing.
sqlt_drv	Contains an entry for each Tags driver. Only really used for browsing tags.
sqlt_err	Contains errors that have occurred executing tags.
sqlt_wq	The "write queue". All write requests are entered into this table, where the driver will detect and execute them. The result will be written back by the driver, and will be noticed by the provider.

Tag Execution Concepts

Polling – Many operations require polling of the database by either the driver or the provider. To ensure efficiency, all polling operations utilize indexed timestamp fields. This allows the database to do very little work when nothing has changed.

Tag Configuration – Tags are configured by inserting or modifying the appropriate entries in the configuration tables above. Configuration change is signaled to the provider by updating the **configchange** of **sqlt_core** to be the current time. Deleting a tag works by setting its **deleted** column and then **touching** config change. This will inform all drivers and providers to remove the tag from memory. At some point later, a daemon will delete the tag information from the database.

Tag Execution, drivers – Each tag has a **drivername** property that indicates which driver is responsible for executing it. Other drivers and providers with different names will treat the tag as an **external** tag – a tag driven by a different entity – and will only monitor its value.

Tag Execution, scan classes – Each tag is assigned to a scan class. The idea is that scan classes will define how often the tag should execute, as well as provide more advanced options like leased and driven execution. In reality, the tag driver is free to execute tags as it desires, but it is important to understand how the scan classes and the **sqlt_sci** table are expected to work, as that is how the provider will verify that the tags are being executed.

Tag Monitoring – Both providers and drivers generally monitor tag value and configuration changes. In general, the entities will monitor tags whose **dr ivername** isn't equal to their own, which for providers means all tags, since providers don't have a driver name. Monitoring occurs by selecting the tag values (or any information desired) from the appropriate table where one of the indexed timestamp columns is greater than the last checked time. The provider/driver will then store that time in memory as the last check, and will use it in the next poll.

Table Reference

The following is a reference list for the table structures of all the tables listed above. In general, all integer time values are in milliseconds.

sqlt_core

Column	Data Type	Notes
id	integer	Auto-incrementing, unique id for the tag
name	string	Name of tag
path	string	Folder path, in form of `path /to/`
drivername	string	Name of driver responsible for executing tags
tagtype	integer / TagType enum	The type of tag - that is, OPC, DB, and so on
datatype	integer / DataType enum	The type of data provided by the tag
enabled	integer (0 or 1)	Whether the tag is enabled for execution
accessrights	integer / AccessRightsenum	Access permissions for the tag
scanclass	integer	ID of the scan class for the tag
intvalue	integer	Value column used if tag has integer data
floatvalue	double	Value column for float/real data
stringvalue	string	Value column for string data
datevalue	datetime	Value column for date data
dataintegrity	integer / DataQualityenum	Current quality of the value
deleted	integer (0 or 1)	Whether the tag is deleted or not
valuechange	datetime	The last time that the value changed
configchange	datetime	The last time that the tag's config changed

sqlt_sc

Column	Data Type	Notes
id	integer	Auto-incrementing unique id
name	string	Name of the scan class
lorate	integer	The slower rate to run at, in milliseconds. Only rate used if scan class mode is direct
hirate	integer	Higher rate, in ms. Only used if scan class is driver or leased
drivingtagp ath	string	Path to tag to watch if mode is driven
comparison	integer / Comparison enum	Operation to apply to driving tag in driven mode
compareval ue	double	Value to compare driving tag to for driven mode
mode	integer / Scan class mode enum	The mode of the scan class
staletimeout	integer	Time, in milliseconds, before scan class is determined to not be running
leaseexpire	datetime	The time that the lease should expire, if using leased mode
configchange	datetime	The last time that the scan class has been modified
deleted	integer (0 or 1)	Whether the scan class has been deleted

sqlt_sci

Column	Data Type	Notes

sqlt_meta

Column	Data Type	Notes
tagid	Integer	ID of tag that the property belongs to
name	string	The well-known property name
intval	integer	Value, if property has integer type
floatval	double	Value if property has float type
stringval	string	Value, if property has string type

sqlt_as

Column	Data Type	Notes
id	integer	Unique id of alert state
statename	string	Name of alert state
severity	integer /Severity enum	
low	double	Low setpoint
high	double	High setpoint
flags	integer / Alert Flags	Flags that dictate how the state acts
lotagpath	string	Path to tag that provides low setpoint, if low driven flag is set
hitagpath	string	Path to tag that Provides high setpoint, if high driven flag is set
timedeadband	double	Time deadband value
timedbunits	integer / TimeUnits enum	Time deadband units

sqlt_perm

Column	Data Type	Notes
tagid	integer	ID of tag that the permission belongs to
rolename	string	Name of the role that this permission is applied to
accessrights	integer / AccessRights enum	Access rights for the given role on the given tag

sqlt_drv

Column	Data Type	Notes
name	string	Name of the tag drive
ipaddr	string	Address of browser server, blank or null if browsing isn't available
port	integer	Port of browse server

sc_id	integer	The id of the scan class represented
drivername	string	The driver executing this instance
lastexec	datetime	Last time that the scan class executed
lastexecrate	integer	The rate of the scan class at last execution
lastecexduration	integer	Time, in ms, that the scan class took to execute
lastexecopcwrite	integer	Writes to OPC performed during last execution
lastexecopcreads	integer	Value updates from OPC processed in last execution
lastexecdbwrites	integer	Writes to DB performed during last execution
lastexecdbreads	integer	Value updates from the database processed during the last execution
lastecexdelay	integer	The delay between when the scan class should have ran and when it actually ran for the last execution
avgexecduration	integer	The average duration time of the scan class, in ms
execcount	integer	The number of times the scan class has executed
nextexec	datetime	The next time that the scan class should execute

sqlt_wq

Column	Data Type	Notes
id	integer	Auto-incrementing unique id for the write operation
tagid	integer	ID of the tag to write to
intvalue	integer	Value, if tag has integer data type
floatvalue	double	Value, if tag has float or real data type
stringvalue	string	Value, if tag has string data type
datevalue	datetime	Value, if tag has date data type
responsecode	integer / Write Responseenum	The state of the write request. When created, the response code should be set to 2 - Pending
responsemsg	string	Write error if operation failed
t_stamp	datetime	The time that the write request was created
objectid	integer	ID of the object with the error
objectype	integer / Object Type enum	The type of object. Used with objectid to identify the item that caused the message
lifecycleid	integer/ Lifecycle enum	When the message was generated
msgtype	integer / Message Type enum	
errormsg	string	The primary message

stack	string	Additional error information
t_stamp	datetime	When the message was generated

Enum Reference

There are known values for many of the objects in the Ignition Tags. Almost every list in Ignition has an enumeration that starts at 0 and counts down the list. Below are a few:

Basic Tag fields

Tag Type	
0	OPC Tag
1	DB Tag
2	Client Tag
6	Folder Tag

Data Types	
0	Byte
1	Short
2	Integer
3	Long
4	Float
5	Double
6	Boolean
7	String
8	Date
9	Integer Array
10	Long Array
11	Double Array
12	Boolean Array
13	String Array
14	DataSet

Access Rights	
0	Read Only
1	Read/Write
2	Custom

Scan Class Modes	
0	Direct
1	Driven
2	Leased

Write Response	

Quality

Data Quality	
0	Bad Data from OPC
4	CONFIG_ERROR
8	NOT_CONNECTED
12	DEVICE_FAILURE
16	SENSOR_FAILURE
20	Bad, showing last value
24	COMM_FAIL
28	OUT_OF_SERVICE
32	WAITING
64	UNCERTAIN
68	UNCERTAIN showing last value
80	SENSOR_BAD
84	LIMIT_EXCEEDED
88	SUB_NORMAL
28	SERVER_DOWN
192	Good Data
216	Good, with local override
256	OPC_UNKNOWN
300	Config Error
301	Comm Error
310	Expr Eval Error
330	Tag exec error (fsql)
340	Type conversion error
403	Access Denied
404	Not Found
410	Disabled
500	Stale
600	Unknown (loading)
700	Write Pending

Alarms

Comparison Mode	
0	Equal
1	Not Equal
2	Less Than
3	Less Than Equal
4	Greater Than
5	Greater Than Equal

Alert Flags	
0x01	Low Exclusive
0x02	Low Infinite
0x04	High Exclusive
0x08	High Infinite
0x10	Any Change
0x20	Low Driven
0x40	High Driven

0	Failure
1	Success
2	Pending

In this section ...

Understanding Tags

What is a Tag?

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, in transaction groups, and more.

Tags are stored in tag providers. By default, a fresh Ignition installation will have an internal tag provider - this can be thought of as a standard internal tag database. Additionally, it is possible to create external Database-based tag providers, thus turning your SQL database into the tag database. This ability opens up some very flexible architectures.

Despite their low 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 in a central SQL database means that you can build widely distributed SCADA systems more easily than ever before, with a high level of performance and relatively easy configuration. 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 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.

Main Benefits of Tags

Tags work naturally and easily with Ignition to offer the following features:

- **Drag and Drop Screen Design**
  Using the Vision module, you can drag and drop tags to a window to automatically create new bound components. Drag tags to existing components or properties to quickly bind them to the data. Creating powerful status and control screens is literally just clicks away!

- **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.

- **Performance and Scalability**
  Tags offer a great performance on both the Gateway and Client. On the Gateway, the system can support thousands of value changes per second and hundreds of thousands of tags. On Client, Tags improve efficiency with their lightweight subscription architecture. Adding additional Clients creates a nearly negligible effect on the database and the Gateway performance.

- **Integrated Component Feedback**
  Tags offer a quality and overlay system for Vision module 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.

- **Historical Logging**
  Tags Historian™ makes it easier that 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 Easy Chart to create trends or onto a table to display historical values. Tags Historian's robust querying provides you great flexibility in how you retrieve the data.

- **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. You can attach associated data to alarms to turn the alarm journal into a forensic tool to help determine the state of your process when issues occurred.

Creating Tags

The easiest way to create a tag is from the OPC Browser. But you can also create tags manually using the menu items, or by converting a tag type to a new type.

- **Creating From OPC Tags**
  The easiest and most common way to create Tags is to drag tags into the Tag Browser from the OPC Browser. After browsing OPC and finding the tags that you want, simply drag and drop them onto the correct tag provider, and the system will create OPC Tag for each.

- **Creating Tags Manually**
  The above method only works for OPC tags, and then only for browsable tags. For other types of tags, as well as OPC tags that cannot be obtained through browsing, you can, in the **Tag Browser**, click on the **New Tag** icon or right-click on the provider node and select **New Tag**.

- **Converting Tag Types**
  Once created, it is possible to convert a tag to a different type – such as from OPC to Expression. To do this, right-click on the tag, and select **Convert Tag Type**.

Re-naming Tags

Tags can be named anything (inside the rules of allowed characters). In other words, 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 whose underlying data storage changes, or for system with many repeat tag structures. By providing tags with meaningful names and arranging them in hierarchical folders, indirect binding can be used to create robust screens that can be used for multiple systems.

In this section ...

Tags Intro

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, in transaction groups, and more. Additionally, tags offer a core set of features above and beyond simple values, such as the scaling, alarming, meta information, and history facilities.

Depending on the specific type of tag, even more options are available. In general, Tags provide a common interface for tying together many types of data in Ignition.

Historical Tag data is stored in the database connection associated with the selected history tag provider. Ignition creates and manages several tables within the Tag database that the historian uses to not only store your data but also make it easily available within your Ignition projects.

There are several types of Tags that are similar in their configurations, but differ in how the value is generated. The different types of Tags are: OPC Tags, Memory Tags, Query Tags, Complex Tags, System Tags, and Client Tags.

On this page ...

INDUCTIVE UNIVERSITY

Tags in Ignition

Watch the Video

Tag Configuration in the Designer

To configure a tag, right-click the tag, then from the popup menu go to **New Tag** and select the type of tag you want to configure. The **Tag Editor** window is displayed and you can now set the properties for that tag. The **Tag Editor** window has the following sections: General, Numeric, Metadata, Permissions, History, Alarming, SQL Query, Expression, and Tag Event. The description for the properties in each section is described next.

General Properties

Properties common to most tags are as follows:

Property	Binding	Name Description
Name	`Name`	How the tag will be presented and referenced in the system. The tag path will be the provider, the folder structure, and this name.
Value	`Value`	The value of the tag. Can only be modified if the tag allows value writing and the user has sufficient privileges.
Data Type	`Datatype`	The type of the value. 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.
Enabled	`Enabled`	Whether the tag will be evaluated by the scan class. If false, the tag will still be present, but will have a bad quality.
Access Rights	`AccessRights`	Specifies the access level allowed to the tag: **Read/Write**, **Read only**, or **Custom**. If Custom, the tag will use the permission settings.
Scan Class	`ScanClass`	The scan class that will execute the tag. The scan class dictates the rate and conditions on which the tag will be evaluated.
Additional Properties - OPC Tags		
OPC Server	`OPCServer`	The server against which to subscribe the data point.
OPC Item Path	`OPCItemPath`	The path to subscribe to on the server. The point will be subscribed at the rate dictated by the scan class.
Additional Properties - Tag in External Providers		
Driver	`DriverName`	The name of the Ignition Gateway that will be responsible for the execution of the tag. All other Gateways will monitor the value.

Numeric Properties

The numerical properties are available to OPC, DB, and Client tags whose data types are numeric.

Property	Binding	Name Description
Scale mode	`ScaleMode`	If and how the tag value will be scaled between the source, and what is reported for the tag.
Raw Lo	`RawLow`	Start of the "raw" value range.
Raw Hi	`RawHigh`	End of the "raw" value range.
Scaled Lo	`ScaledLow`	Start of "scaled" value range. Raw low will map to Scaled low for the tag.
Scaled Hi	`ScaledHigh`	End of "scaled" value range. Raw high will map to Scaled high for the tag.
Clamp mode	`ClampMode`	How values that fall outside of the ranges will be treated. "Clamped" values will be adjusted to the low/high scaled value as appropriate.
Scale Factor	`ScaleFactor`	For single parameter modes (currently only Exponential Filter), the factor parameter for the equation.
Deadband	`Deadband`	A floating point value used to prevent unnecessary updates for tags whose values "float" by small amounts.
Deadband Mode	`DeadbandMode`	There are two modes to choose from: Absolute or Percentage.

Metadata Properties

The metadata properties provide informational properties for a tag. The values of these fields can be read and modified through scripting, or bound to properties such as range, tooltips, and so on.

Property	Binding	Name Description
Format String	`FormatString`	How the value should be formatted when converted to a string (only applies to numerical data types).
Eng Unit	`EngUnit`	The engineering units of the value.
Engineering Limits Low	`EngLow`	The lowest expected value of the tag.
Engineering Limits High	`EngHigh`	The highest expected value of the tag.
Engineering Limit Enforcement	`EngLimitMode`	Dictates how the engineering range should be enforced on the tag. If not "none", the tag will change to bad quality ("limit exceeded"), when the value exceeds the specified range.
Tooltip	`Tooltip`	The tooltip provides a hint to visual components as to what should be displayed when the user hovers their mouse cursor over the component that is being driven by the value of this tag.
Documentation	`Documentation`	A freeform text property for information about the tag.

Permission Properties

Note: Make sure your tag's **Access Rights** (in the General section of the Tag Editor window) 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.

By default, a tag's **Access Rights** property is set to **Read/Write**, which means that any user can read the value of the tag and write to the tag. **Read-only** mode makes the tag non-writeable for all
users. **Custom** mode allows the tag to assign read/write or read-only privileges to individual roles. Any roles not explicitly granted a right by using the custom permissions editor will not be able to read the tag's value or write to the tag.

History Properties

The properties on the **History** tab of the **Tag Editor** window, specify if and how the tag's history will be stored in the Tags Historian system.

Property	Binding Name	Description
Store History	HistoryEnabled	Whether the tag will report its history to the Tags Historian system.
History Provider	PrimaryHistoryProvider	Which Tag Historian data store the tag will target. A particular tag can only target one history store.

Historical Scanclass	HistoricalScanclass	The scan class to use to evaluate tag history. This allows the tag's history to be stored at a slower rate than the status is updated at.
Historical Deadband	HistoricalDeadband	A deadband that applies only to historical evaluation.
Max time between records	HistoryMaxAgeMode / HistoryMaxAge	The maximum amount of time that can pass before a new record is logged for the tag. See Max Time Between Records below.
Timestamp Source	HistoryTimestampSource	Which timestamp is used for the value of the tag. See Timestamp Source below.
Value Mode	InterpolationMode	How interpolation will be handled for the tag in querying. See Value Mode below.

Value Mode

The value mode, **analog** or **discrete**, dictates the type of value that the tag represents, and will affect how the deadband is applied to values, and how interpolation should be performed when querying. Interpolation is the method in which the Tag Historian query system generates values for a tag when the desired time does not fall directly on a sample timestamp.

Discrete

- **Storage** - The deadband will be applied directly to the value. That is, a new value (V_1) will only be stored when: $|V_1-V_0| >=$ Deadband.
- **Interpolation** - The value will not be interpolated. The value returned will be the previous known value, up until the point at which the next value was recorded.

Analog

- **Storage** - The deadband is used to form a corridor along the trajectory of the value. A new value is only stored when it falls outside the previous corridor. When this occurs, the trajectory is recalculated, and a new corridor formed. See below for an example.
- **Interpolation** - The value will be interpolated linearly between the last value and the next value. For example, if the value at $Time_0$ was 1, and the value at $Time_2$ is 3, selecting $Time_1$ will return 2.

Max Time Between Records

Normally Tag Historian only stores records when values change. By default, an "unlimited" amount of time can pass between records – if the value doesn't change, a new row is never inserted in the database. By modifying this setting, it is possible to specify the maximum number of scan class execution cycles that can occur before a value is recorded. Setting the value to 1, for example, would cause the tag value to be inserted each execution, even if it has not changed. Given the amount of extra data in the database that this would lead to, it's important to only change this property when necessary.

Timestamp Source

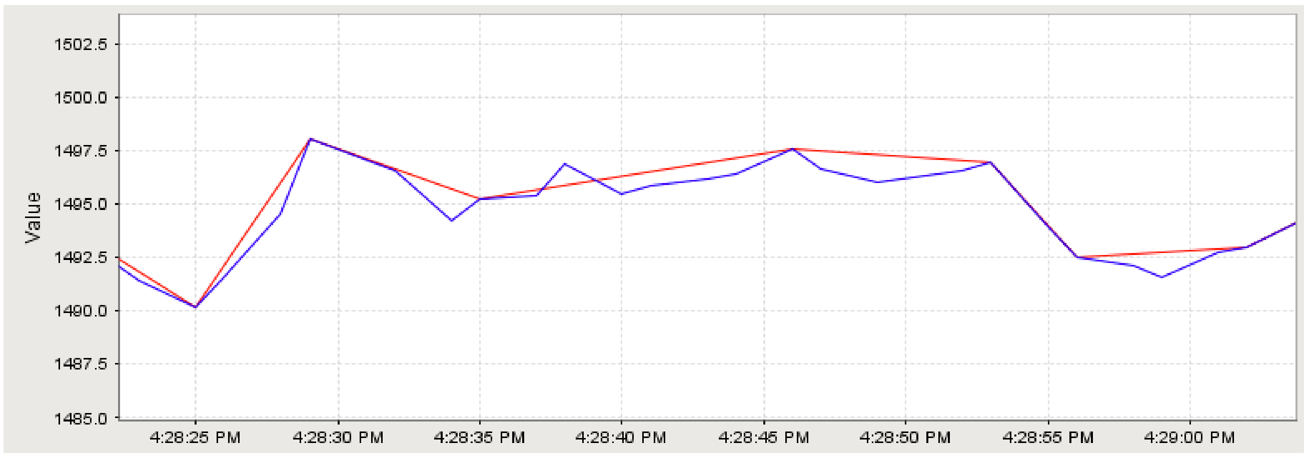
When a Tag executes, there are two possible timestamps that can be observed: the time associated with the data, and the time that the tag was evaluated. The first case is generally only interesting when the value is provided by an OPC server. In most cases, the time provided by OPC, which in Ignition is referred to as the "Value" time, will be very close to the system time. Some servers, however, either due to their location or how they function (history playback, for example), will provide times that are very different than the current time.

It is generally desirable to store the System time, as it is the time that the value was actually observed by the system, and it creates a uniform timeframe for all realtime data. However, in the later case, it is necessary to store the time provided by the OPC server. Using the Value timestamp source has several consequences: the system is no longer able to validate the tag quality against the scan class' execution, and tag value interpolation will behave differently.

The validation of the scan class execution is generally not a concern when recording historical playback data. Interpolation only occurs when the value mode is Analog, and when there is not a value for every time window. Using System time, the value is only interpolated during the last "scan class execution window", that is, one scan class timeframe before the next value. Using Value time, however, the value is interpolated for the entire time between two data points.

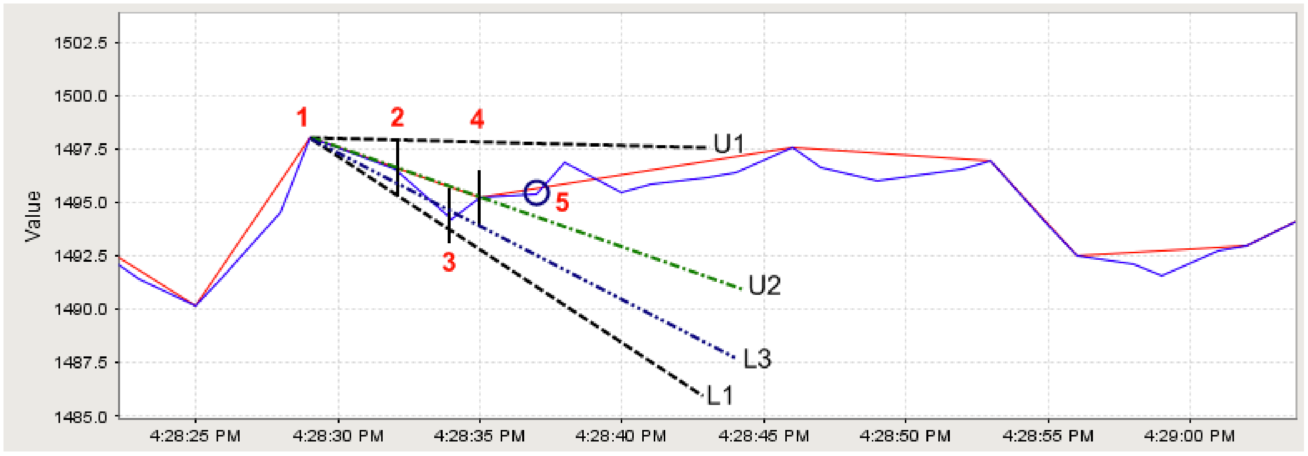
The Deadband, and Analog Compression

As described above, the deadband value is used differently depending on whether the tag is configured as a Discrete tag, or an Analog tag. Its use with discrete values is straight forward, registered a change any time the value moves +/- the specified amount from the last stored value. With Analog tags, however, the deadband value is used more as a compression threshold, in an algorithm similar to that employed in other Historian packages. It is a modified version of the 'Sliding Window' algorithm. Its behavior may not be immediately clear, so the following images show the process in action, comparing a raw value trend to a "compressed" trend.



In this image, an analog value has been stored. The graph has been zoomed in to show detail; the value changes often and ranges over time +/- 10 points from around 1490.0. The compressed value was stored using a deadband value of 1.0, which is only about .06% of the raw value, or about 5% of the effective range. The raw value was stored using the Analog tag mode, but with a deadband of 0.0. While not exactly pertinent to the explanation of the algorithm, it is worth noting that the data size of the compressed value, in this instance, was 54% less than that of the raw value.

By looking at one specific sequence, we can see how the algorithm works:



The sequence starts with the second stored compressed value on the chart.

1. A value is stored. No further action is taken.
2. The next value arrives. A line is made through the value, with the size of the specified deadband value. A line is projected from the last stored value to the upper (line U1), and lower (line L1), bounds of this new value line. This establishes the initial corridor.
3. A new value arrives. The same procedure is taken, and new lines are created. However, only lines that are more restrictive than the previous are used. In this case, that means only line U2, the new upper line.
4. Another value arrives, causing a new lower line (L3) to be used.
5. Finally, a value arrives that falls outside of our corridor. The last received value (value 4) is stored, and a the process is started again from that point.

Alarming Properties

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 clear.

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. Or, you might bind the setpoint of an alarm to a tag that operators can manipulate, thereby letting the setpoint be changed at runtime.

To bind a tag, simply click on the binding icon ( ), and the binding UI will slide in from the right. From here you can select the binding type (No Binding, Tag, Expression, or UDT Parameter, if applicable). Note that the expression can reference many useful values such as the tag's value and other settings of the alarm. When you've configured the binding to your liking, click on the **Back** button.

Associated 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.

Main Alarm Settings

Alarm Name

Each alarm has it's own name. For example, if the tag is representing a level, the alarm name might be **High Level**.

Enabled

This boolean determines whether or not the alarm will be evaluated. A disabled alarm's condition will not be evaluated, and thus will not generate any alarm events.

Priority

An alarm's priority can affect how is appears in an alarm status table, or can affect how it is escalated through a pipeline. The priorities, which can be referenced by their integer equivalent in scripts and expressions, are: Diagnostic [0], Low [1], Medium [2], High [3], Critical [4]

Timestamp Source

Chooses where the timestamp for the alarm event should come from: the system time of when the event was generated, or the timestamp of the value that tripped the event.

Display Path

This is a string value that will be used to display the alarm to operators. If this is blank, the operator will see the path to the tag instead. Please use the forward-slash character to separate hierarchy levels in this path, for example: East Area/Boilers/Boiler5

Ack Mode

Dictates how acknowledgement works for the alarm.

- **Unused** - Acknowledgement will not be used for this tag, and any alarm 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 for the user to manually acknowledge alarms.

Notes

A place for any free-form documentation about the alarm that can be displayed to operators.

Ack Notes Required

If this setting is true, the operators will be unable to acknowledge this alarm without entering some notes.

Shelving Allowed

If this setting is true, the shelving feature will be unavailable for this alarm.

Alarm Mode Settings

Mode

This setting controls what condition this alarm is evaluating. The available modes are as follows:

Equal - Active when the tag's value equals the alarm's setpoint.

Not Equal - Active when the tag's value does not equal the alarm's setpoint.

Above Setpoint - Active when the tag's value is above the alarm's setpoint.

Below Setpoint - Active when the tag's value is below the alarm's setpoint.

Between Setpoints - 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.

Outside Setpoints - 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.

Out of range - The same as Outside Setpoints, but uses the tag's Engineering High and Engineering Low as the high and low setpoints.

Bad Quality - Active if the tag value becomes a bad quality, for example, on comm loss.

Any Change - 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.

Bit State - 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.

On Condition - 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.

Setpoint / Low Setpoint / High Setpoint

The setpoint properties are used for many alarm modes to specify in what range the alarm becomes active.

Inclusive / Low Inclusive / High Inclusive

These settings correspond to a setpoint. If true, the range will be active if the value is exactly equal to the setpoint, not only above or below it.

Deadbands and Time Delays

Deadband Mode

Absolute - The deadband setting is considered to be an absolute value.

Percent - The actual deadband is calculated as a percent of the tag's engineering unit span.

Deadband

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 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, 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.

Active Delay

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".

Clear Delay

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".

Notification Settings

Active Pipeline

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.

Clear Pipeline

The name of an alarm notification pipeline to put this alarm into when it becomes clear in order to send out clear messages.

Email Notification Settings

Custom Subject

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.

Custom Message

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.

Extended Configuration Settings

Modules may add additional properties to alarms. They will appear in the list as standard properties, unless the module is no longer available, in which case they'll appear as Associated Data.

Expression/SQL Properties

DBTags have the ability to use an expression or a SQL query as their value instead of an OPC item path. This can be used to select information from the database or create your own formulas to manipulate other tag values.

Expression

In expression mode, the tag can use all of the features available in the expression language. It can refer to other tags, and use operators and functions to calculate a value for the tag.

SQL Query

In this mode, the tag's value will be the result of the specified SQL query. The query can be any valid query, but should result in only one value. Note that insert and update queries can be used, and will often result in an integer value, so the tag's data type should be set accordingly.

Like SQL Query bindings in the Vision module, the queries for tags can refer to other tag values. The values of referenced tags will inserted as literal text in the query before being sent to the database.

Next ...

- Types of Tags

Types of Tags

Ignition supports many different types of tags. There are Gateway, System, and Client tags and the tags behave differently depending on which of the three scopes they belong to.

On this page ...

INDUCTIVE UNIVERSITY

Types of Tags

Watch the Video

Tag Scope

The three major scopes are Gateway, System, and Client. Every tag belongs to a scope and the scopes play a role in how the tags behave.

- Gateway Executed Tags (includes: OPC Tags, Memory Tags, Expression Tags, Query Tags, and complex Tags (UDTs)
- Gateway System Tags
- Client Tags

While in discussing "Tags", we commonly mean Gateway executed tags, but System and Client tags can play an important role in the overall design of a project.

Gateway Executed Tags

Tags executed in the Gateway support all of the primary features of Tags: scaling, alarming, history, and role-based permissions. They are identical in their configurations, apart from defining how the value is generated.

OPC Tags

The OPC tags specify an OPC server and address which drives their values. The OPC address will be subscribed at the rate of the tag's scan class.

Memory Tags

The Memory tags are simply values. The value is specified during configuration, and is stored when written (if the tag allows writing).

Expression Tags

The 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, logic operations and more.

SQL Query Tags

The SQL 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.

Complex Tags (UDTs)

The Complex tags are created out of standard tag types, but offer a variety of additional features. In simple terms, 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.

Gateway System Tags

The Gateway System tags provide status about the system, such as memory usage, performance metrics, and so on. They exist for the Client and the Gateway. You can modify the Gateway System tags to use alarming, history, and scaling but additional System tags cannot be added to the System folder. The tag names and folder structure is managed by the Ignition Gateway.

Client Tags

The Client tags, as the name implies, are only available for use in Clients. They are scoped at the client level unlike the Gateway executed tags. All clients will have the same client tags however the actual values the client tags represent will be independent of each other. In other words even though they are created in the Designer, each client will create their own instances. This makes them very useful as in-project variables, for passing information between screens, and between other parts of the clients, such as scripting.

Client tags do not have a scan class. When set to run as an expression or query, a poll rate is specified dictating how often the value should be calculated.

Client tags support datasets in addition to all the other datatypes such as integers and date-times.

Next ...

- Keeping Tags Organized

Keeping Tags Organized

It is important to give Tags a meaningful structure (you can also rename the Tags independent of their Item Path) and arrange them in hierarchical tag folders so that they are easy to understand and identify for all developers.

Keeping Tags Organized

Watch the Video

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 scripting.

A tag path looks something like this: `[Source]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 `[Source]` portion surrounded by square braces can have the following options:

Source Option	Meaning	Applicability
[Tag Provider Name]	The name of the tag provider that hosts the tag.	OPC and Expression tags
[] or not specified	The default tag provider for the current project.	OPC, Expression tags
[.]	Relative to the folder of the tag that is being bound.	Expression, Client tags
[~]	Relative to the tag provider of the tag that is being bound (root node).	Expression, Client tags
[Client]	Refers to a client tag.	Client
[System]	Refers to a system tag.	System

Using Relative Paths

Paths that begin with [.] or [~] are known as *relative paths*. They are used inside Tags that bind to other tags, and 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 . . 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 with Ignition's tag paths being nothing more than 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.

```
tankNumber = 2
tagPath = "[default]Tanks/Tank %i/Level" % tankNumber
```

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.

Next ...

- Tag Quality and Overlays

Tag Quality and Overlays

Data Quality is the measure of how reliable a particular Tag's data is. If a tag's quality is not *Good*, the value generally should not be trusted and an overlay is shown to make the bad quality visible. There are a wide variety of causes of bad data, from network disconnections, to software failure to invalid tag configuration.

The quality is a property of the tag called "Quality", and you can see it in the Tag Browser once you expand (click on the + icon) a tag. Additionally, bad tag qualities are reflected in components bound to tags through the *quality overlay* system.

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:

Tag Quality and Overlays

Watch the Video

Quality	Value	Meaning
Good	192	The data has met all criteria for being considered reliable.
OPC_Bad_Data	0	The data is not reliable, further data isn't available.
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.
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.
Expression_Eval_Error	310	The expression in the tag generated an error during execution. The error log should provide more information on the error.
Tag_Exec_Error	330	There was an error when evaluating the tag.
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.
Driver_Demo_Timeout	900	The system driving the tag is operating in demo mode and has timed out.
Access_Denied	403	The tag permission settings do not allow the current user to view the tag.
Not_Found	404	The tag, or a tag referenced from inside of it, could not be found (incorrect reference path).
Disabled	410	The tag's "enabled" property has been set to false.
Stale	500	The tag has not been evaluated within the expected time frame. There is likely a deeper problem with the tag provider.
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.

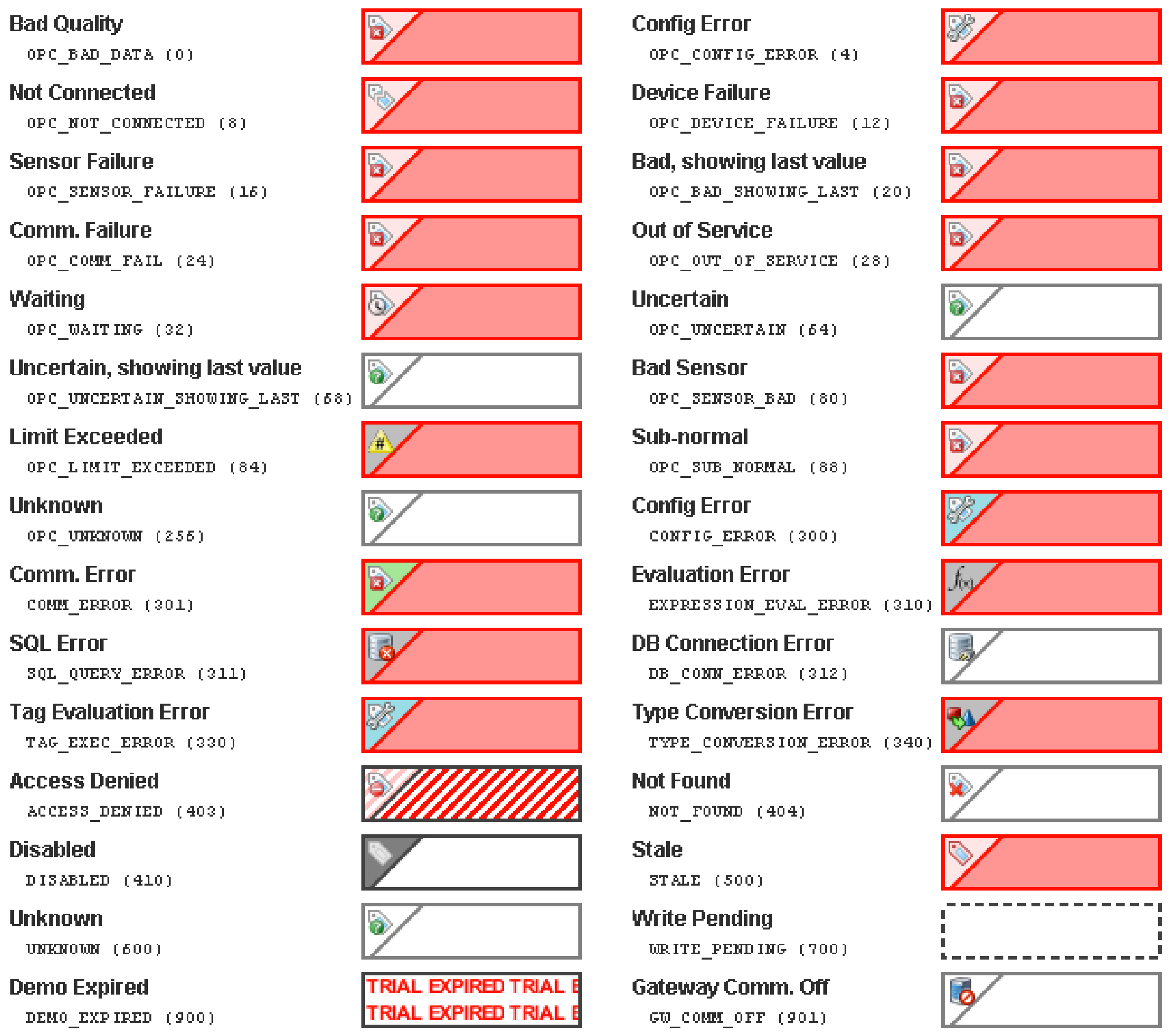
Tag Quality and Referenced Tags

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".

Quality Overlays

It is especially important for operators 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, components will get visual overlays for various reasons to indicate that the data they are displaying is not good. Each data binding that drives a component is evaluated for quality. If any of these qualities becomes poor, the 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.



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)	Gateway Comm. Off	GW_COMM_OFF (901)

Next ...

- Tag Metadata Properties

Tag Metadata Properties

The metadata properties provide informational properties for a Tag. The values of these fields can be read and modified through scripting, or bound to properties such as Format, Engineering Units, Engineering Low, Engineering High, Engineering Limits, Tooltip, and Documentation.

Property	Binding	Name Description
Format String	`FormatString`	How the value should be formatted when converted to a string (only applies to numerical data types).
Eng Unit	`EngUnit`	The engineering units of the value.
Engineering Limits Low	`EngLow`	The lowest expected value of the tag.
Engineering Limits High	`EngHigh`	The highest expected value of the tag.
Engineering LimitEnforcement	`EngLimitMode`	Dictates how the engineering range should be enforced on the tag. If not "none", the tag will change to bad quality ("limit exceeded"), when the value exceeds the specified range.
Tooltip	`Tooltip`	The tooltip provides a hint to visual components as to what should be displayed when the user hovers their mouse cursor over the component that is being driven by the value of this tag.
Documentation	`Documentation`	A freeform text property for information about the tag.

INDUCTIVE UNIVERSITY

Tag Metadata Properties

Watch the Video

Next ...

- Tag Scaling Properties

Tag Scaling Properties

Configuring a tag's scaling properties will condition the data for use within the Ignition Designer. Scaling will take the raw value driving a Tag, and scale the value. For example, if the capacity of a tank is 5250 Gallons but the tanks 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.

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, DB, and Client tags whose data types are numeric.

Property	Binding	Name Description
Scale mode	`ScaleMode`	If and how the tag value will be scaled between the source, and what is reported for the tag.
Raw Lo	`RawLow`	Start of the **raw** value range.
Raw Hi	`RawHigh`	End of the **raw** value range.
Scaled Lo	`ScaledLow`	Start of **scaled** value range. Raw low will map to Scaled low for the tag.
Scaled Hi	`ScaledHigh`	End of **scaled** value range. Raw high will map to Scaled high for the tag.
Clamp mode	`ClampMode`	How values that fall outside of the ranges will be treated. "Clamped" values will be adjusted to the low/high scaled value as appropriate.
Scale Factor	`ScaleFactor`	For single parameter modes (currently only for Exponential Filter), the factor parameter for the equation.
Deadband	`Deadband`	A floating point value used to prevent unnecessary updates for tags whose values **float** by small amounts.
Deadband Mode	`DeadbandMode`	There are two modes to choose from: Absolute or Percentage.

Tag Scaling Properties

Watch the Video

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
```

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.

Next ...

- Importing and Exporting Tags

Importing and Exporting Tags

Ignition can export and import tag configurations to and from **XML** (Extensible Markup Language) or **CSV** (Comma Separated Value) file formats.

To import and export tags

1. Go to the **Designer.** Select the desired folder from the **Tag Browser** that contains the tags that you want to export.
2. On the **Tag Browser** toolbar, click the **Export** icon to export, or **Import** icon to load a previously exported file.
   The Save (to export) or the Open (to import) window is displayed.
3. Choose the **.xml** or **.csv** to export or import your tags.
   Note: Importing tags from XML will overwrite tags if exact match is found.

INDUCTIVE UNIVERSITY

Importing and Exporting Tags

Watch the Video

XML Format

Extensible Markup Language (XML) is a markup language that defines a set of rules to encode documents. XML is easy for software to read and export therefore exporting tags to an XML format results in simple and reliable transport of all tag configuration settings.The XML format is best for cases where you are interested in transferring tag definitions and configurations from one Gateway to another. However it is not optimized for directly editing its contents. Modern variations of Microsoft Excel can open and display the contents of an XML file.

CSV Format

The CSV format is often used to export tags with the intent of making mass edits to the tag definitions and configurations in a spreadsheet program such as Microsoft Excel. The tags can be edited and imported back into the Gateway. Alarm properties are not included in CSV export format but are included in XML export format.

There are many configuration settings for tags than what is displayed in a CSV 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.

Property Values in CSV Export Files

When viewing a CSV file which has been exported from Ignition, you will see many columns each showing a configuration property value. The property consists of its name and its value, in some cases there might be additional attributes available.

The following table shows the configuration property names and values you will see in an exported CSV file.

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	
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		
ExpressionType	Int	0 - None 1 - Expression 2 - SQL Query	
DriverName	String		Used for external tags
ScanClass	String		
HistoryEnables	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.

Alarm Property Values included in XML Export Files

Property Name	Type	Values (if applicable)	Description

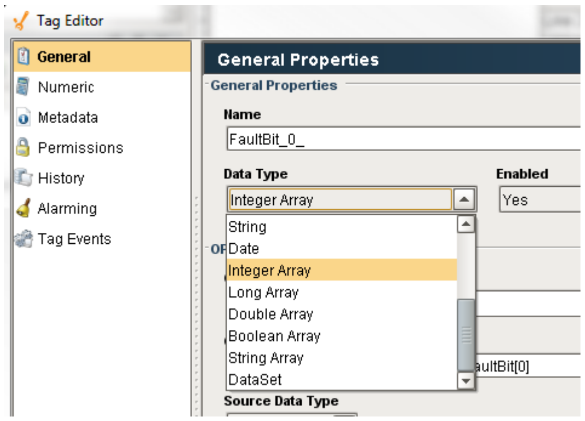
Enabled	Boolean	true/false	
Priority	String	Diagnostic, Low, Medium, High, Critical	May also be numeric, 0-4.
DisplayPath	String		
ActivePipeline	String		
ClearPipeline			
Deadband	Float		
DeadbandEvalMode	Integer	0 - Absolute 1 - Percentage	
TimeOnDelaySeconds	Float		
TimeOffDelaySeconds	Float		
TimestampSource	Int	0 - System 1 - Value	
AckMode	Int	0 - Unused 1 - Auto 2 - Manual	
Notes	String		
AckNotesReqd	Boolean	true/false	
ShelvingAllowed	Boolean	true/false	
Mode	String	Equality Inequality AboveValue BelowValue BetweenValues OutsideValues OutOfEngRange BadQuality AnyChange Bit OnCondition	
SetpointA			The setpoint, or the **low** setpoint for dual value modes.
SetpointB			The **high** setpoint for dual setpoint modes.
InclusiveA	Boolean		
InclusiveB	Boolean		
BitOnZero	Boolean		Used by the **Bit** condition, to indicate that 0 is **active**.
BitPosition			Bit to use in **Bit** condition.
ActiveCondition	Boolean		The property that drives the **OnCondition** mode.
AnyChange	Boolean		Whether or not to apply the **any change** behavior in applicable modes.

Next ...

- add link here

Array and Dataset Tags

Array and Dataset data types are now available on tags. Both data types 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.



FaultBit_0_

Data Type: Integer Array

Enabled: Yes

String
Date
Integer Array
Long Array
Double Array
Boolean Array
String Array
DataSet

INDUCTIVE UNIVERSITY

Array and Dataset Tags

Watch the Video

Array Tags

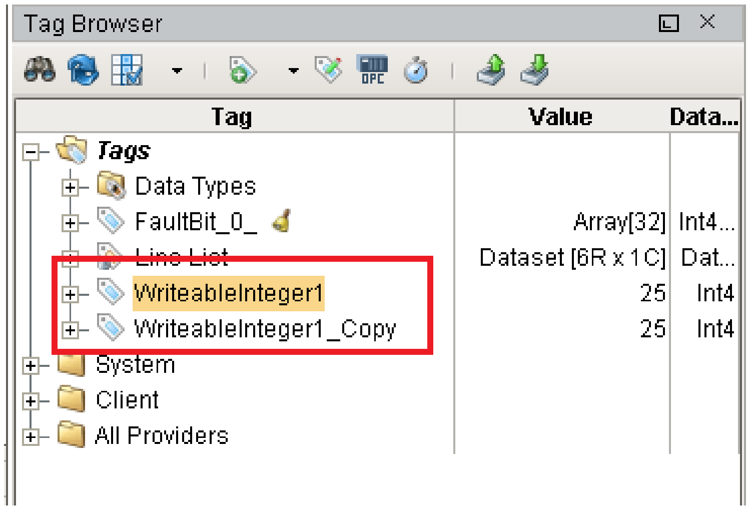
Many OPC servers and drivers already support array type tags, and now each element in the array can easily be represented with the array data types in Ignition. Additionally, array data types can be used with devices that do not support array types, and will instead expose each bit in the value.

*Note: **Individual members of Array Tags are read only**, and cannot be used with bi-directional bindings. You can set the value of the whole array in scripting (ie: writing 5 to an Integer Array will set both the 0 and 2 positions to 1 and everything else to 0) or additional (non-array type) tags can be created to write to those addresses.*

Array Alarm Example

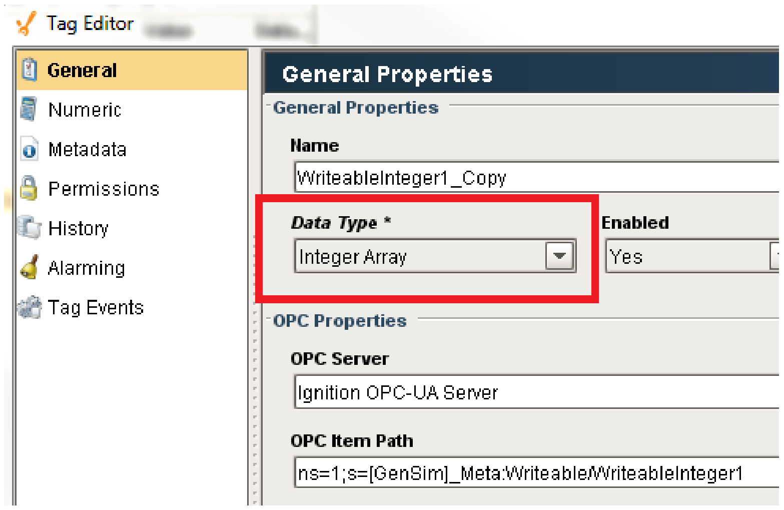
Because the core data type of each element in the array is the same, it is possible to apply Tag History, Alarming, or Scaling configurations onto the array, and these configurations will be inherited by each element.

1) The WriteableInterger1 and WriteableInteger1_Copy tags below are using the same OPC Item Path, and both are currently configured as Integer tags.



Tag	Value	Data...
Tags		
Data Types		
FaultBit_0_	Array[32]	Int4...
Line List	Dataset [6R x 1C]	Dat...
WriteableInteger1	25	Int4
WriteableInteger1_Copy	25	Int4
System		
Client		
All Providers		

2) Edit the WriteableInteger1_Copy tag, and change it's data type to an Integer Array, and click OK


Tag Editor

General
Numeric
Metadata
Permissions
History
Alarming
Tag Events

General Properties

General Properties

Name: WriteableInteger1_Copy

Data Type *: Integer Array

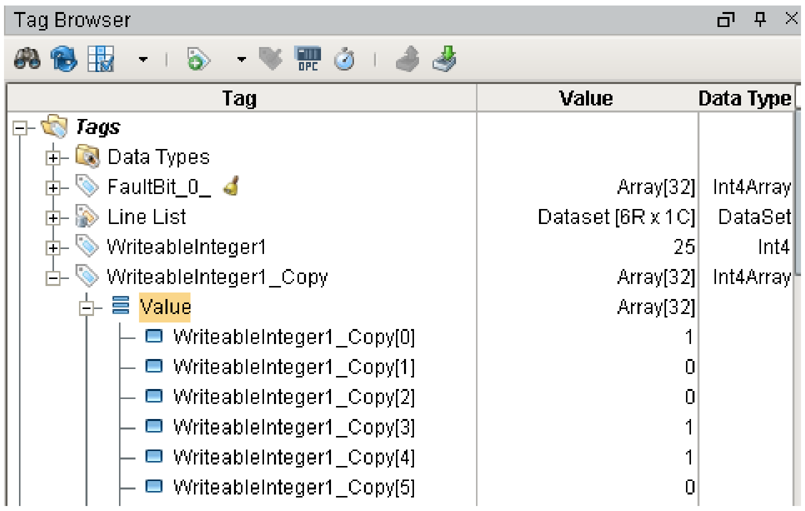
Enabled: Yes

OPC Properties

OPC Server: Ignition OPC-UA Server

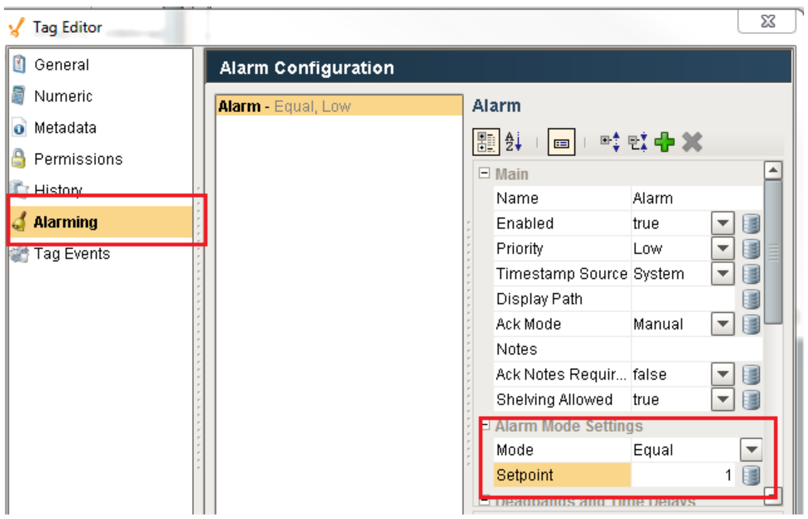
OPC Item Path: ns=1;s=[GenSim]_Meta:Writeable/WriteableInteger1

3) Once the tag finishes re-subscribing, the value for WriteableInteger1_Copy will now show "Array[32]" instead of the previous value. Expanding the tag reveals a new Value section, and expanding this section will show the value of each bit.


Tag Browser

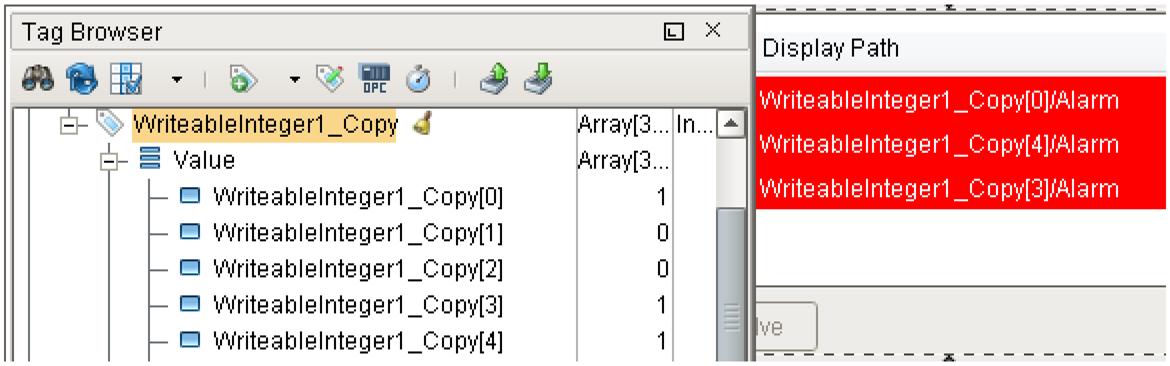
Tag	Value	Data Type
Tags		
Data Types		
FaultBit_0_	Array[32]	Int4Array
Line List	Dataset [6R x 1C]	DataSet
WriteableInteger1	25	Int4
WriteableInteger1_Copy	Array[32]	Int4Array
Value	Array[32]	
WriteableInteger1_Copy[0]	1	
WriteableInteger1_Copy[1]	0	
WriteableInteger1_Copy[2]	0	
WriteableInteger1_Copy[3]	1	
WriteableInteger1_Copy[4]	1	
WriteableInteger1_Copy[5]	0	

4) Next, edit the WriteableInteger1_Copy tag again, and head to the Alarming section. Create a new alarm with Mode set to Equal, but change the Setpoint to 1, and then click OK. This way an alarm will be active for each element that is set to 1.



Main	
Name	Alarm
Enabled	true
Priority	Low
Timestamp Source	System
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Requir...	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Equal
Setpoint	1

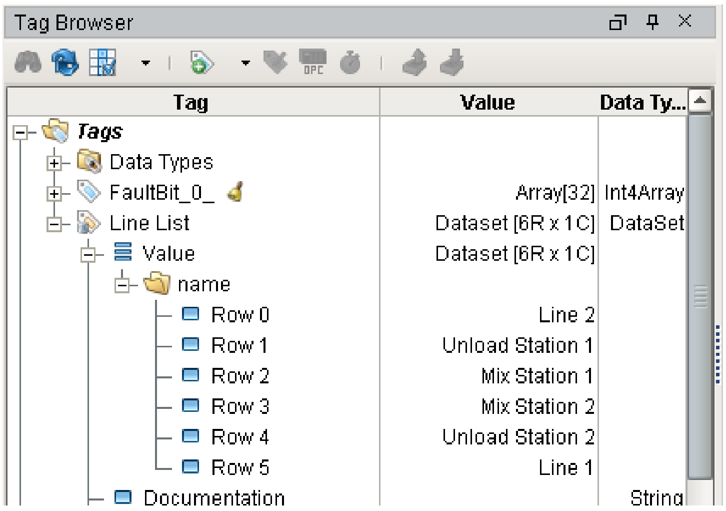
5) Create a new Alarm Status Table report, and you should see an alarm for each true element.



Tag	Value	
WriteableInteger1_Copy	Array[3...	In...
Value	Array[3...	
WriteableInteger1_Copy[0]	1	
WriteableInteger1_Copy[1]	0	
WriteableInteger1_Copy[2]	0	
WriteableInteger1_Copy[3]	1	
WriteableInteger1_Copy[4]	1	

Display Path
WriteableInteger1_Copy[0]/Alarm
WriteableInteger1_Copy[4]/Alarm
WriteableInteger1_Copy[3]/Alarm

Dataset Tags



Tag	Value	Data Ty...
Tags		
Data Types		
FaultBit_0_	Array[32]	Int4Array
Line List	Dataset [6R x 1C]	DataSet
Value	Dataset [6R x 1C]	
name		
Row 0	Line 2	
Row 1	Unload Station 1	
Row 2	Mix Station 1	
Row 3	Mix Station 2	
Row 4	Unload Station 2	
Row 5	Line 1	
Documentation		String

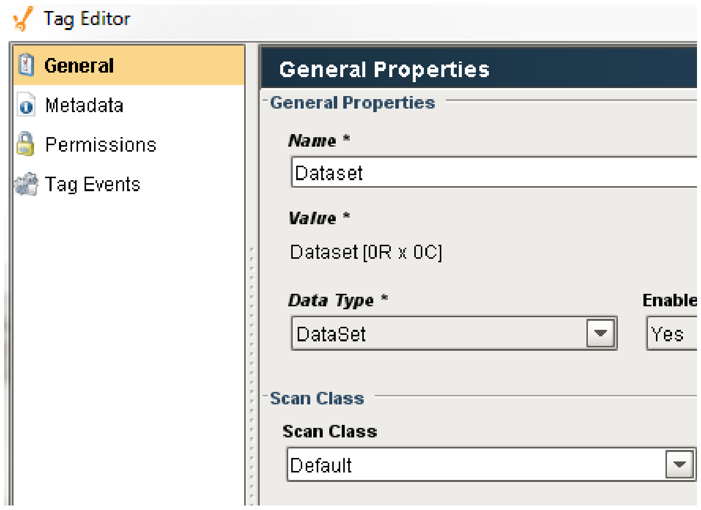
Dataset Tags allow multiple rows and columns worth of data to be stored in a tag. Each column is exposed as a separate folder in the tag (i.e., the "name" folder in the above image). Dataset Tags can be driven by a query, so it's possible to query for multiple columns on a row in a single tag. This is much more efficient than using multiple query tags (and thus multiple queries) to retrieve the same data.

Memory Tags that use the Dataset type support bi-directional bindings in tables.

Dataset Tag Write Example

The following example will write the contents of a Table component to a dataset tag.

1) Create a new Memory tag. Name it Dataset, and change the data type to Dataset. The Dataset will be empty by default.


Tag Editor

General
Metadata
Permissions
Tag Events

General Properties

General Properties

Name *
Dataset

Value *
Dataset [0R x 0C]

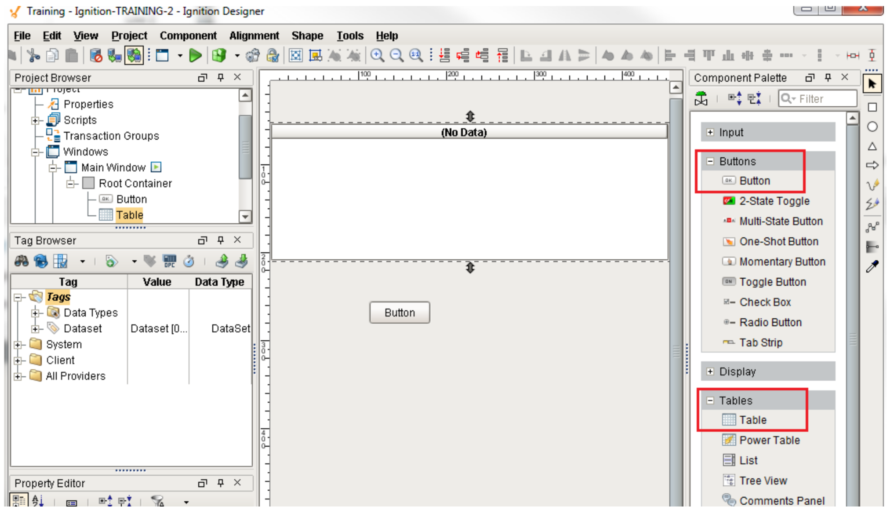
Data Type *
DataSet

Enable
Yes

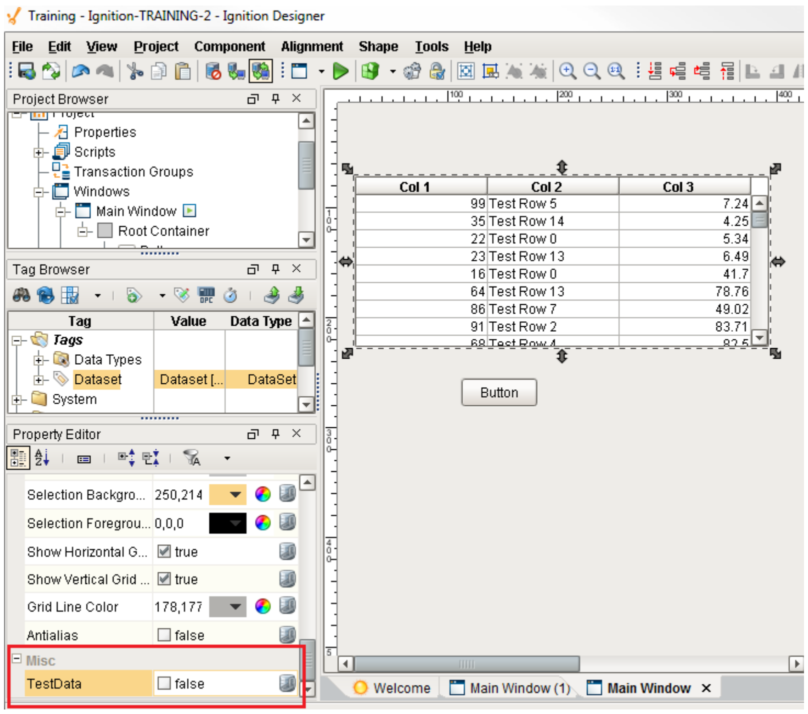
Scan Class

Scan Class
Default

2) On a new window, add a button component and table component.



3) Generate some test data on the Table by toggling the TestData property.



Tag	Value	Data Type
Tags		
Data Types		
Dataset	Dataset [...	DataSet
System		

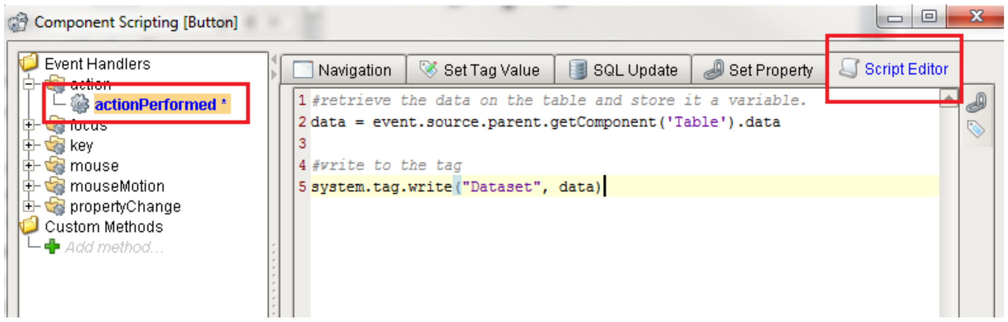
Col 1	Col 2	Col 3
99	Test Row 5	7.24
35	Test Row 14	4.25
22	Test Row 0	5.34
23	Test Row 13	6.49
16	Test Row 0	41.7
64	Test Row 13	78.76
86	Test Row 7	49.02
91	Test Row 2	83.71
68	Test Row 4	82.5

Selection Backgro...	250,214
Selection Foregrou...	0,0,0
Show Horizontal G...	✓ true
Show Vertical Grid ...	✓ true
Grid Line Color	178,177
Antialias	false
Misc	
TestData	false

4) Next, the button will need a script to handle writing to the tag. Right click on the button and select Scripting. Choose the actionPerformed Event Handler, and click on the Script Editor tab. Use the following code:

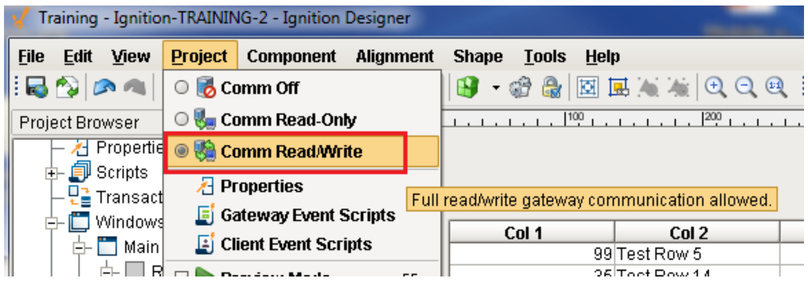
Code Snippet

```
#retrieve the data on the table and store it a variable.
data = event.source.parent.getComponent('Table').data

#write to the tag
system.tag.write("Dataset", data)
```



```
#retrieve the data on the table and store it a variable.
data = event.source.parent.getComponent('Table').data

#write to the tag
system.tag.write("Dataset", data)
```

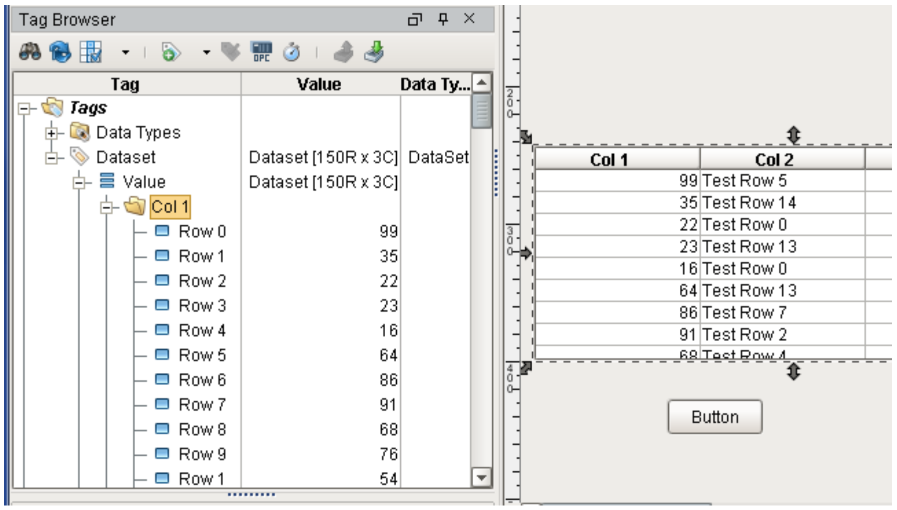
5) Make sure the Designer is set to Read/Write mode: Project menu > Comm Read/Write



Full read/write gateway communication allowed.

Col 1	Col 2
99	Test Row 5

6) Put the Designer into preview mode, and click on the button. The data in the table has been passed to the tag. Expanding the tag in the Tag Browser will show the new values on the tag.



Tag	Value	Data Ty...
Tags		
Data Types		
Dataset	Dataset [150R x 3C]	DataSet
Value	Dataset [150R x 3C]	
Col 1		
Row 0	99	
Row 1	35	
Row 2	22	
Row 3	23	
Row 4	16	
Row 5	64	
Row 6	86	
Row 7	91	
Row 8	68	
Row 9	76	
Row 1	54	

Col 1	Col 2
99	Test Row 5
35	Test Row 14
22	Test Row 0
23	Test Row 13
16	Test Row 0
64	Test Row 13
86	Test Row 7
91	Test Row 2
68	Test Row 4

Next ...

Browsing and Creating OPC Tags

There are two ways to create OPC tags.

- **From OPC Browser**
  Using the OPC Browser to browse and find the tag and then drag it to the Tag Browser to create the tag, see Browsing for OPC Tags. 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, from the Tag Editor window, see Creating OPC Tags Manually.

In this section ...

Browsing for OPC Tags

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.

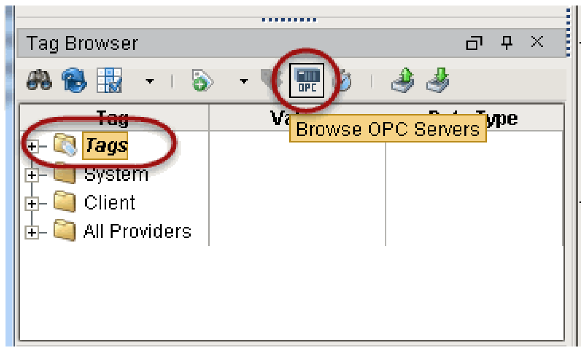
Creating Tags

Now that you are in the Designer, a good first step is to create some Tags. You'll use these tags for realtime status and control and to store history with the Tags Historian. Tags are all configured in the Tag Browser panel.

To create tags by dragging them from OPC

If you have created a device in the earlier step, the easiest way to create some Tags is by dragging the tags from the OPC Browser.

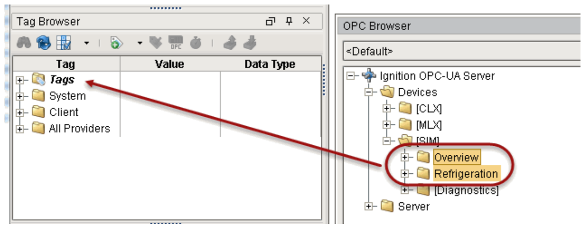
1. In the **Tag Browser**, select the **Tags** folder and click the **Browse OPC Servers** icon.



   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 the devices in it you created earlier. Browse the devices and find some tags that you're interested in.

2. From the **OPC Browser**, highlight the tags (for example, as in the picture below, both folders inside the **SIM** folder) and drag them to 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. As soon as you drag the tags into Tag Browser, you can see their values being updated in the Tag Browser,

   That's it! You now have some Tags. You can see their values come in and start updating automatically. By default, they update at a rate of 1 per second.

Next ...

- Creating OPC Tags Manually

INDUCTIVE UNIVERSITY

Browsing for OPC Tags

Watch the Video

To Continue on the step-by-step simple workflow...

After you **Create Tags**, as described on this page, click the link below for Step 6 to continue.

Step 1. **Launch the Gateway**

Step 2. **Connect to a Device**

Step 3. **Connect to a Database**

Step 4. **Launch the Designer**

Step 5. **Create Tags**

Step 6. **Launch Clients**

Step 7. **Add History to Tags**

Step 8. **Create a Graph and View Graphs in Clients**

Creating OPC Tags Manually

You can create OPC Tags manually when you can't find them by using the OPC Browser.

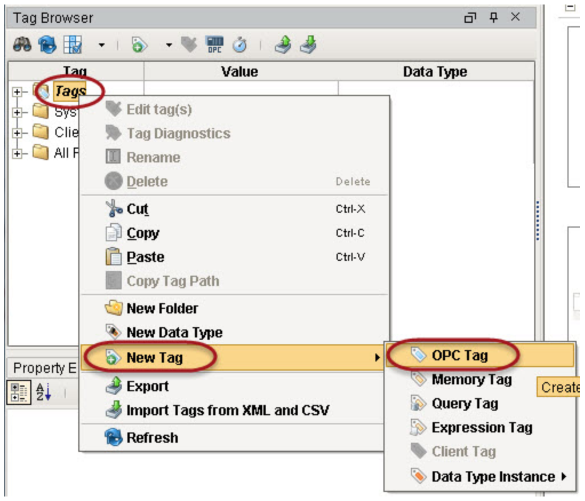
INDUCTIVE UNIVERSITY

Creating OPC Tags Manually

Watch the Video

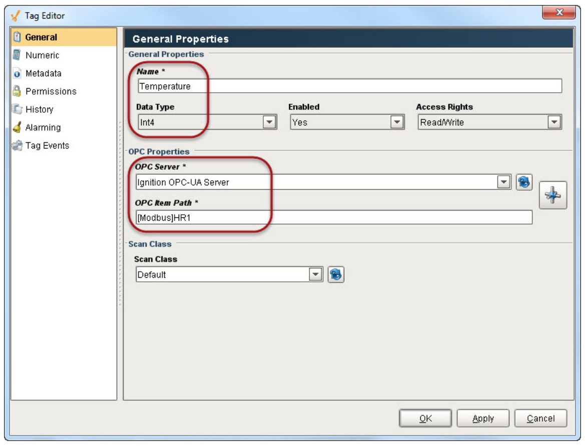
To create OPC Tags Manually

1. From **Tag Browser**, right-click on **Tags** or a folder of your choice and select **New Tag > OPC Tag**.



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

   Name: **Temperature**
   Data Type: **Int4**
   OPC Server: choose **Ignition OPC-UA Server** from the dropdown
   OPC Item Path: **[Modbus]HR1**, the **Modbus** device name goes in the square brackets then you give the address to PLC which in this case is the **HR** designator plus **1** as the Modbus address. The Modbus Specific Addressing section, explains how your can construct these addresses.



Tag Editor

General
Numeric
Metadata
Permissions
History
Alarming
Tag Events

General Properties

General Properties

Name *: Temperature

Data Type: Int4
Enabled: Yes
Access Rights: Read/Write

OPC Properties

OPC Server *: Ignition OPC-UA Server

OPC Item Path *: [Modbus]HR1

Scan Class

Scan Class: Default

OK Apply Cancel

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

Next ...

- Addressing Bits

Addressing Bits

1000000000000

1000000000000

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	0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0	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

Addressing Bits

Watch the Video

To address an individual bit

Suppose a MicroLogix PLC is connected to a Gateway.

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** with a value of 1 or True
   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

Creating Memory, Expression, and Query Tags

Memory tags are simply values you use as setpoints or for storing information. Expression tags let you perform calculations based on other tags. Query tags execute a SQL query, whose result provides the tag value, and they can reference other tags to build dynamic queries.

In this section ...

Creating Memory Tags

Memory tags are simply values. 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 when written.

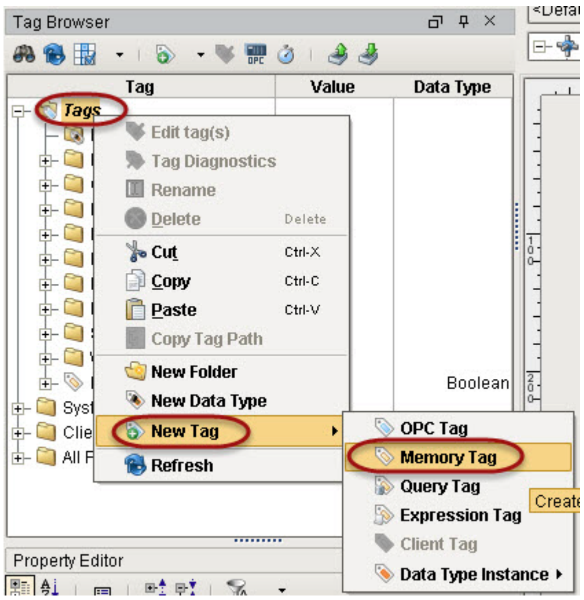
INDUCTIVE UNIVERSIT

Creating Memory Tags

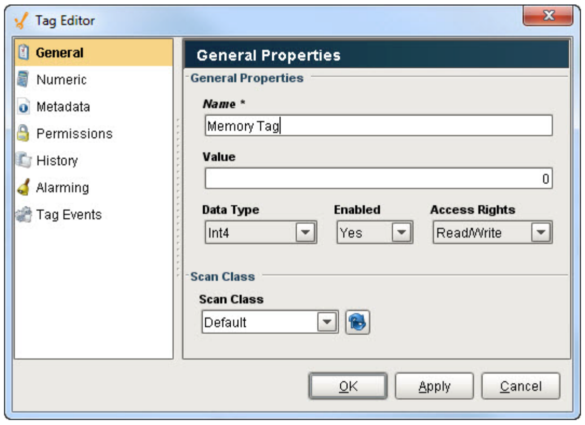
Watch the Video

To create memory tags

1. From **Tag Browser**, right-click on **Tags** or a folder of your choice and select **New Tag > Memory Tag**.



2. In the **Tag Editor**, type a **Name** (for example, Memory Tag), initial **Value**, and **Data Type**.



General Properties

Name * Memory Tag

Value 0

Data Type Int4

Enabled Yes

Access Rights Read/Write

Scan Class

Scan Class Default

3. Click **OK**.

   That's it! 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.

Next ...

- Creating Expression Tags
- Creating Query Tags

Creating Expression 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, logic operations, and more. Expression Tags cannot be written to.

With expression tags you can do a special calculation based on other tags. In this way, you can do the calculation one time and view it on any window in your project.

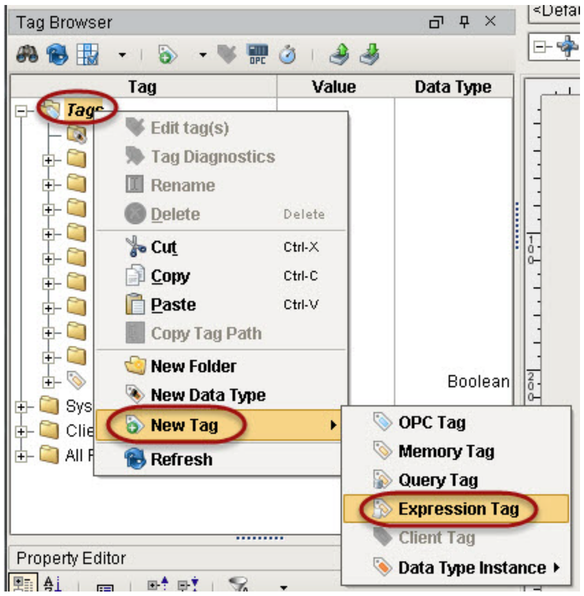
INDUCTIVE UNIVERSITY

Creating Expression Tags

Watch the Video

To create an expression tag

1. In the **Tag Browser**, right-click on **Tags** or a folder of your choice and select **New Tag > Expression Tag**.



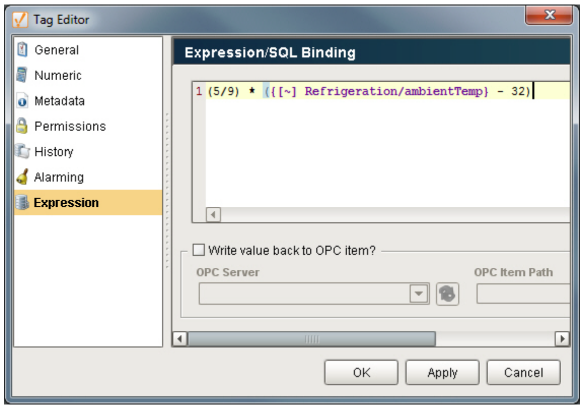
2. In the **Tag Editor**, type **FtoC Expression** for **Name** and choose **Float4** for **Data Type**.

3. Click on the **Expression** tab on the left side of **Tag Editor**.

4. 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:

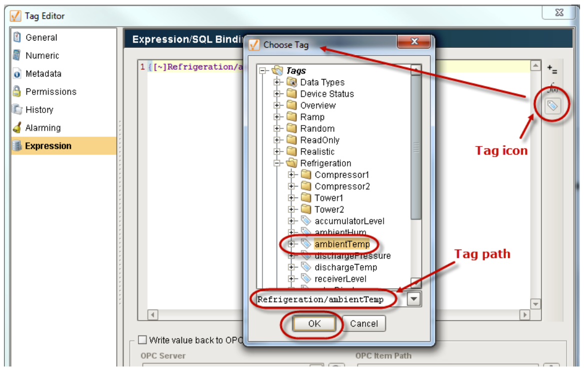
```
(5/9) * ({[~]Refrigeration/ambientTemp} - 32)
```



```
(5/9) * {[~] Refrigeration/ambientTemp} - 32)
```

If you want to use a different tag, just replace the `{[~]Refrigeration/ambientTemp}` tag path with any Tag of your choice.

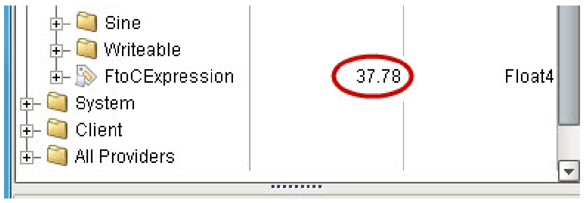
Note that instead of typing the referenced tag path (in this example `Refrigeration/ambientTemp`), you can use the Tag icon on the right side of the **Tag Editor > Expression** window to select the tag path.



```
1 {[~]Refrigeration/a
```

5. Click **OK**.

   You will see the value from your calculation on the Tag. Note that the Expression tags are evaluated at the rate the Scan Class is set to.



With an expression tag, you can also choose to write the value back to another OPC item. In the **Tag Editor > Expressions**, simply click the check box next to **Write value back to OPC item?** and select the tag you want to
write to. Every time the value is calculated, the value gets written to the selected tag.

Next...

- Creating Query Tags

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 SQL Query tag comes from the database through a SQL query.

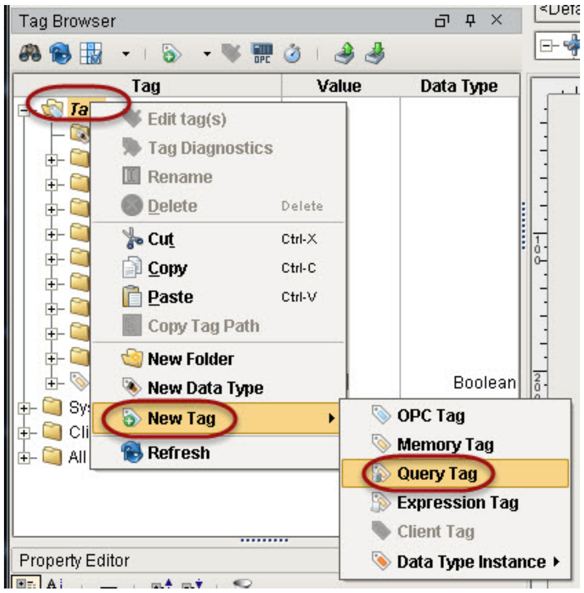
INDUCTIVE UNIVERSITY

Creating Query Tags

Watch the Video

To create a SQL query tag

1. In **Tag Bowser**, right-click on **Tags** or a folder of your choice and select **New Tag > Query Tag**.



2. In the **Tag Editor** window, type **Current Time** for **Name** and choose **DateTime** for **Data Type**.

3. Click on **SQL Query** tab on the left side of **Tag Editor**.

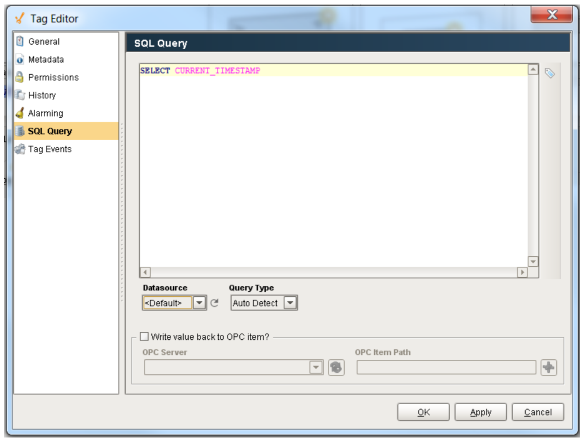
4. Type in any SQL query you want.

   Note: The query must return a scalar result because the data type is not a dataset.

5. In the drop-down list, select the database to run the query through, for example **MySQL**.

6. Copy in the following query:

```
SELECT CURRENT_TIMESTAMP
```



```
SELECT CURRENT_TIMESTAMP
```

Datasource: <Default>
Query Type: Auto Detect
☐ Write value back to OPC item?
OPC Server:
OPC Item Path:

Query Type

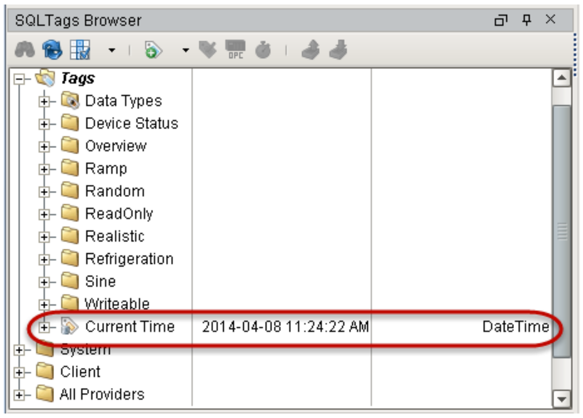
In certain circumstances you may want to update the database instead of querying it for data. The query tag will accept an update query and update the target data source. The value attribute of the tag becomes the number of rows affected by the update query.

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.

7. Click **OK**.

   You will see the value from your SQL query on the **Current Time** Tag.



Current Time 2014-04-08 11:24:22 AM DateTime

Next ...

- add link here

Creating User Defined Types (UDTs)

Complex Tags, also referred to as UDTs (user defined types), offer the ability to leverage object-oriented data design principles in Ignition. Using complex tags, 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.

On this page ...

Primary UDT Features

The primary features of UDTs are as follows:

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 tag, 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 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 for dealing with irregular data and corner cases.

UDT Terminology

Many terms are frequently used when discussing complex tags:

UDT or **UDDT**
User Defined Type or User Defined Data 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 3 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 1 setting.

Members
Members are the tags inside of a data type or instance.

Defining Data Types

Creating a new data type is very similar to creating standard tags.

To create a new data type

1. Simply select the **Tags** folder in the **Tag Browser**.

2. Click the **Tag** icon on the **Tag Browser** toolbar.
   The tag creation menu is displayed.

3. Select **New Data Type** from the menu.
   The **Tag Editor** is displayed.
   When editing complex tags, the **Tag Editor** window appears a bit differently. The member tree structure is presented in the upper left. By selecting a member, the tag property categories are displayed below the member area, and the editor for the selected category appears to the right.

To extend other Types

Data types can extend other data types, to add additional members, or override default values. The parent data type can be modified by selecting the data type itself in the tag member tree. Note that the data type can only be selected when the tag is first created. After that, it is not possible to modify the parent tag type.

To add members to a data type

Simply click the **green +** icon at the toolbar above the member tree. Data types can contain standard tags like OPC and DB, as well as folders and instance of other complex types.

To add 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.

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 sub types or instances. Furthermore, unlike normal tags, in the context of a data type many properties (general the string based properties) can reference the custom attributes of the type in order to build parameterized types.

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}`, or use **Ctrl-Space** to display a list of available parameters to choose from.

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.

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`.

Properties that can be parameterized

The following tag properties can reference parameters:

- Value (for string data type only)
- OPC Server
- OPC Item Path
- Tooltip
- Documentation
- Expression/SQL Query
- Bindable Alarm Properties (Note: you can bind a property directly to the parameter, or use parameters in the binding expression, or directly in string property values)

Overriding Properties

Sub types 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.

Creating Data Types from Existing Tags or OPC

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.

Converting from existing Tags

To create a data type from existing Tags, simply select the tags or folders you wish to include, right-click and select **Create Data Type from Selected**.

The Tag Editor window is displayed with the selected tags pre-populated as members. From here you can modify the tags, add parameters, and so on. The original tags will not be affected.

Tip: If you select a single folder as the root to create the type from, its sub-members will be added, and its name will be the basis for the type (that is, the folder itself won't be included in the structure).

Creating from OPC

If you have data types defined in your PLC (or OPC server), you can short-cut the conversion step outlined above by simply 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.

Creating Instances of Complex Types

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.

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 right-click a tag in the **Tag Browser**, then select **New Tag > Data Type Instance > Multi-instance Wizard** from the menu. The **Instance Creation Wizard** window is displayed. Once you select a data type, you will see its parameters come into the table, where you can specify values for them. The size of the parameter patterns dictate how many tags are created, or if only single values are specified, you can choose to create multiple copies of the same configuration.

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:

1-10,20-30,30-40	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.

In this section ...

Understanding Complex Tags (UDTs)

Complex Tags are created out of any of the standard tag types, but offer a variety of additional features. In simple terms, you can think of them as a way to create "data templates", where a particular structure of tags is defined, and then many instances can be created as if each were a single Tag.

Use UDTs (user-defined data types) to design re-usable, parameterized, and extendable data types. Then new instance tags can be created and configured in seconds, potentially saving a tremendous amount of time over traditional tag systems.

Understanding Complex Tags (UDTs)

Watch the Video

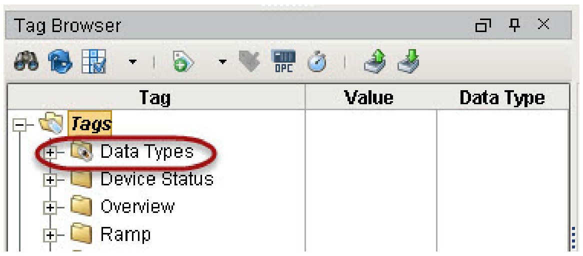
Next ...

- Creating UDT Definition

Creating UDT Definition

So far we have only dealt with simple tags. Additionally, In Ignition, you can 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 **Tags Browser** called **Data Types**.



INDUCTIVE UNIVERSITY

Creating UDT Definition

Watch the Video

UDTs (User Defined Types) can be created in the following different ways:

- from browsing OPC servers via OPC Browser
- by converting them from existing tags
- by manually creating them

If your PLC supports UDTs, the easiest way to create a type is from OPC.

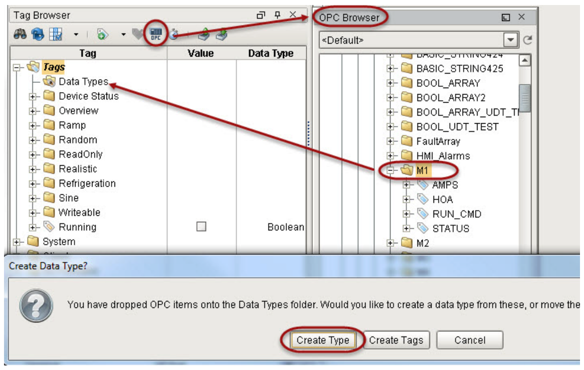
To create the UDT from OPC Browser

In the example here, we will use a ControlLogix which supports UDTs. We will use the second method from above that is, creating the UDT from OPC. In our ControlLogix PLC, lets say we already have a motor UDT setup and now we will create the UDT in Ignition.

1. From **Tags Browser**, click the **OPC Browse** icon.

2. Under the **CLX** folder, go to the **Global** folder and find the **M1** folder.
   M1 represents a motor which has four tags: AMPS, HOA, RUN_CMD, and STATUS.

3. Drag the **M1** folder to the Data Types folder in Tags Browser.
   A window prompts you asking if you want to create a type or tags.

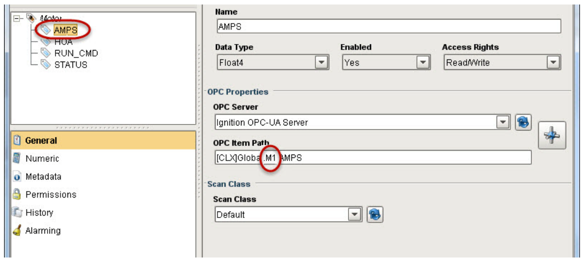


You have dropped OPC items onto the Data Types folder. Would you like to create a data type from these, or move the

4. Click on **Create Type**.
   The **Tag Editor** window is displayed. Notice all four tags AMPS, HOA, RUN_CMD, and STATUS are automatically part of the UDT.

5. In the Tag Editor, change the name from **M1** to **Motor** to make the type name more of a generic name.

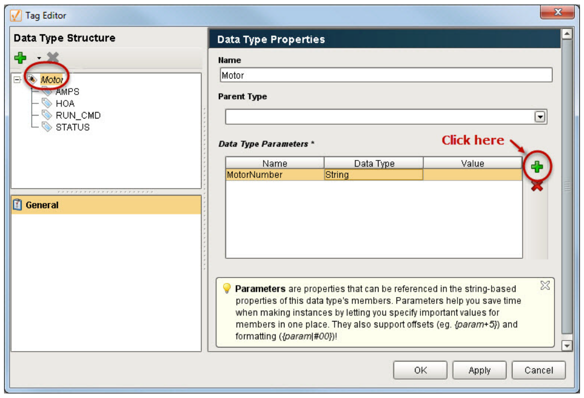
6. Right now each tag is pointing to a specific address in the PLC, select the **AMPS** tag to see that it is pointing to **M1** in the PLC.



Name: AMPS
Data Type: Float4
Enabled: Yes
Access Rights: Read/Write
OPC Properties
OPC Server: Ignition OPC-UA Server
OPC Item Path: [CLX]Global.M1.AMPS
Scan Class
Scan Class: Default

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 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.

7. In the **Tag Editor**, select the Motor data type and add a new parameter by clicking the **green +** button and setting the following:
Name: **MotorNumber**
Data Type: **String**



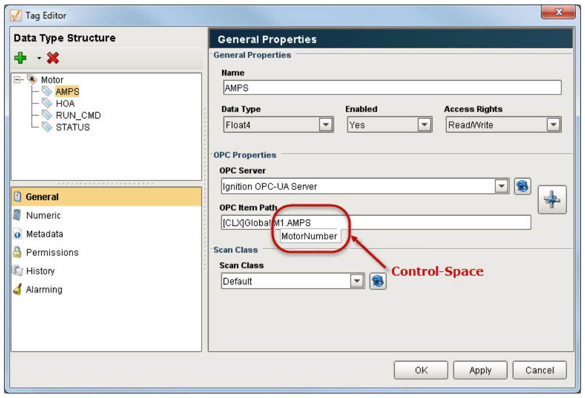
Name: Motor
Parent Type:
Data Type Parameters *

Name	Data Type	Value
MotorNumber	String	

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param\|#00}*)!

8. Click **OK**.
Now for each tag, we can substitute the motor number with the **MotorNumber** parameter.

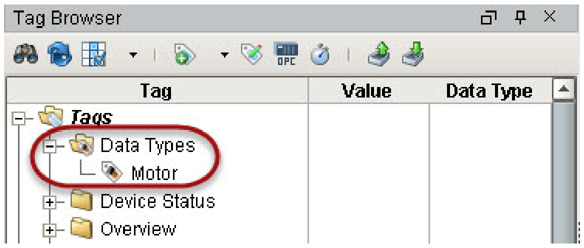
9. Open the Tag Editor again, select the **AMPS** tag. In the **OPC Item Path**, click on **M1** and press **Ctrl-Space** to bring up the list of parameters, click **MotorNumber** to replace **1** with **{MotorNumb er}**.



General Properties
Name: AMPS
Data Type: Float4
Enabled: Yes
Access Rights: Read/Write
OPC Properties
OPC Server: Ignition OPC-UA Server
OPC Item Path: [CLX]Global.M1.AMPS
MotorNumber
Scan Class
Scan Class: Default

10. Do the same for the HOA tag.
The **OPC Item Path** will then show:
[CLX]Global.M{MotorNumber}.HOA

11. Click **OK** to save the UDT.
You will see the **Motor** UDT in the **Data Types** folder.



Next ...

- Creating UDT Instances

Creating 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.

INDUCTIVE UNIVERSITY

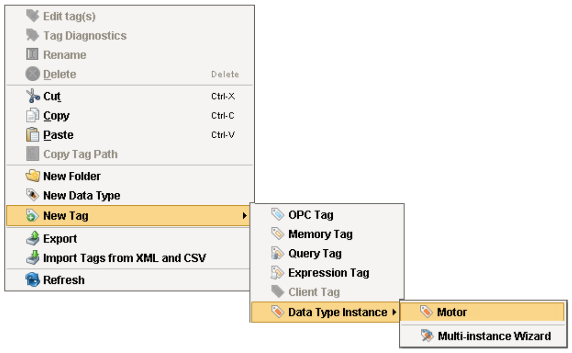
Creating UDT Instances

Watch the Video

To Create a UDT Instance

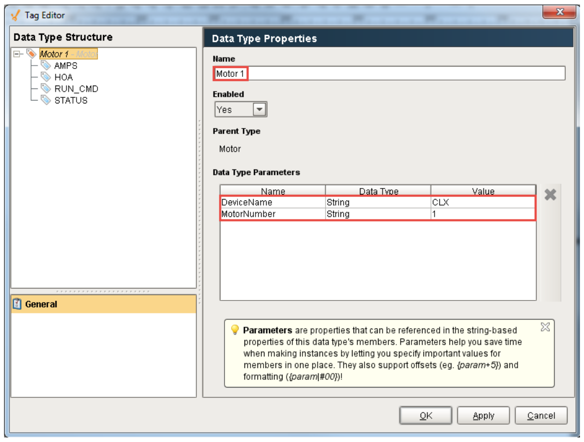
Once a UDT Definition is created, you can create an instance of the UDT as an actual tag in Ignition. Using the Motor UDT Definition from the previous section, lets create a UDT Instance.

1. In the **Tag Browser**, right-click on **Tags** and select **New Tag > Data Type Instance > Motor** to create a new instance.



2. Assign the new instance a name, **Motor 1** and enter a value for each parameter that is used in the configuration of the UDT Motor.

   Assign the **DeviceName** parameter a value of **CLX** for ControlLogixs.
   Assign the **MotorNumber** parameter a value of **1** for Motor 1.
   Click **OK** and the Motor 1 instance is created in Ignition.



Name: Motor 1

Enabled: Yes

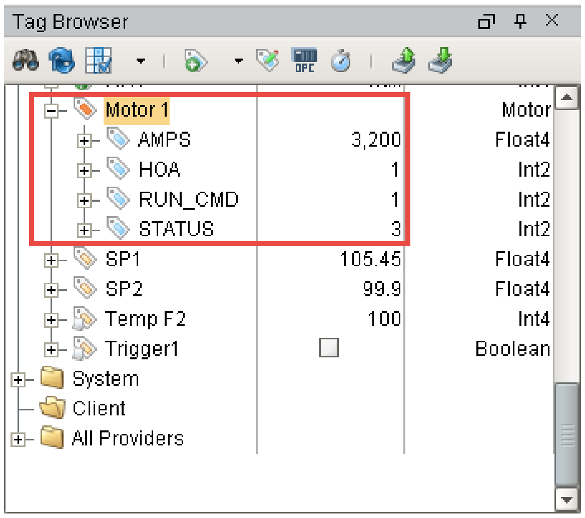
Parent Type: Motor

Data Type Parameters

Name	Data Type	Value
DeviceName	String	CLX
MotorNumber	String	1

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param|#00}*)!

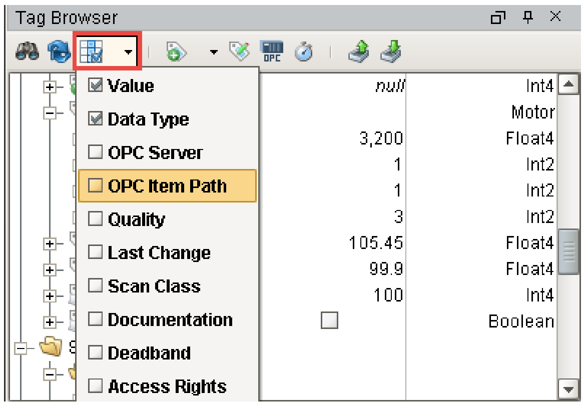
3. From the **Tag Browser**, expand **Motor 1** to verify that the tags are working correctly.



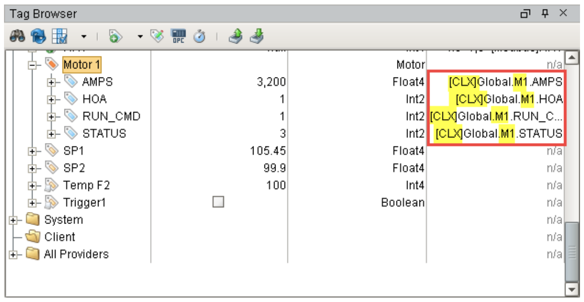
Tag Browser

Motor 1		Motor
AMPS	3,200	Float4
HOA	1	Int2
RUN_CMD	1	Int2
STATUS	3	Int2
SP1	105.45	Float4
SP2	99.9	Float4
Temp F2	100	Int4
Trigger1	☐	Boolean
System		
Client		
All Providers		

4. To see how Ignition translated the path for each parameter and verify that they are correct and of good quality, check the OPC Item Path in the Tag Browser.
   From the **Tag Browser**, select **Motor 1**, click **Column Selector** and check **OPC Item Path** from the dropdown.



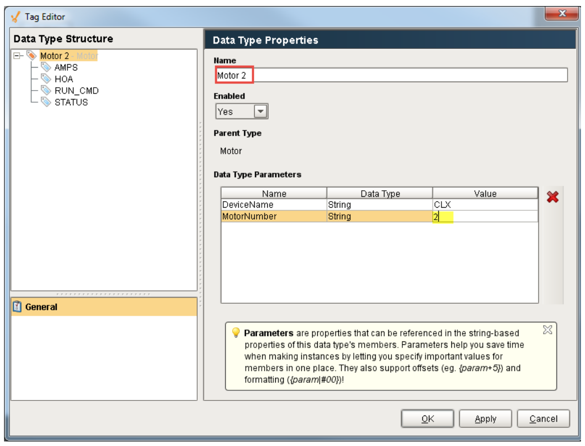
   You can see that Ignition displays both the **DeviceName (CLX)** and **MotorNumber (M1)** correctly.



Tag Browser

Motor 1		Motor	n/a
AMPS	3,200	Float4	[CLX]Global.M1.AMPS
HOA	1	Int2	[CLX]Global.M1.HOA
RUN_CMD	1	Int2	[CLX]Global.M1.RUN_C...
STATUS	3	Int2	[CLX]Global.M1.STATUS
SP1	105.45	Float4	n/a
SP2	99.9	Float4	n/a
Temp F2	100	Int4	n/a
Trigger1	☐	Boolean	n/a
System			n/a
Client			n/a
All Providers			n/a

5. To create another instance of Motor 1, **copy** and **paste Motor 1** in the Tag folder to create **Moto r 2**.
   Open the **Tag Editor** and modify the **MotorNumber** value from **1** to **2** to point to a different Motor, in this case, Motor 2.
   Click **OK**.



Name: Motor 2

Enabled: Yes

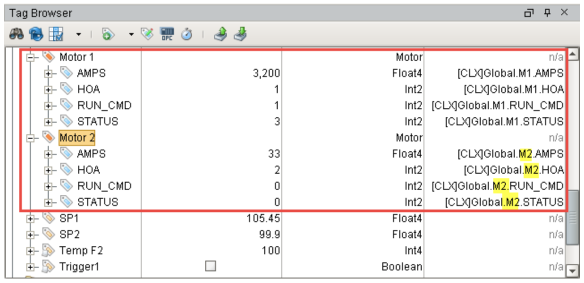
Parent Type: Motor

Data Type Parameters

Name	Data Type	Value
DeviceName	String	CLX
MotorNumber	String	2

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param\|#00}*)!

6. To confirm Motor 2 is working correctly, open the **Tag Browser** and expand **Motor 2**.



Motor 1		Motor	n/a
AMPS	3,200	Float4	[CLX]Global.M1.AMPS
HOA	1	Int2	[CLX]Global.M1.HOA
RUN_CMD	1	Int2	[CLX]Global.M1.RUN_CMD
STATUS	3	Int2	[CLX]Global.M1.STATUS
Motor 2		Motor	n/a
AMPS	33	Float4	[CLX]Global.M2.AMPS
HOA	2	Int2	[CLX]Global.M2.HOA
RUN_CMD	0	Int2	[CLX]Global.M2.RUN_CMD
STATUS	0	Int2	[CLX]Global.M2.STATUS
SP1	105.45	Float4	n/a
SP2	99.9	Float4	n/a
Temp F2	100	Int4	n/a
Trigger1	☐	Boolean	n/a

Creating UDT Instances is really simple once you have a UDT Definition in place.

Next ...

- UDT Multi-Instance Wizard

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.

INDUCTIVE UNIVERSITY

UDT Multi-Instance Wizard

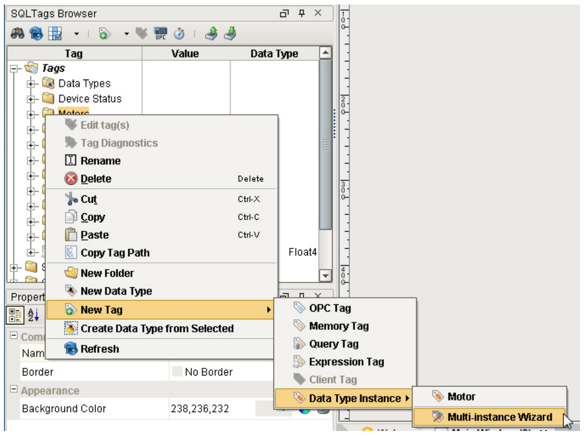
Watch the Video

To make instances of the UDT

Now that we have the Motor UDT (see Creating Complex Tags (UDTs)) we can make instances of it. Instances are running copies of a data type with concrete data for all members.

1. In **Tags Browser**, right-click on **Tags** and select **New Folder** to create a new folder and call it **Motors**.
   You can create a single instance of the motor or use the multi-instance wizard to rapidly create all four at the same time. Here we will use the multi-instance wizard.

2. Right-click on the **Motors** folder and select **New Tag> Data Type Instance> Multi-Instance Wizard**.

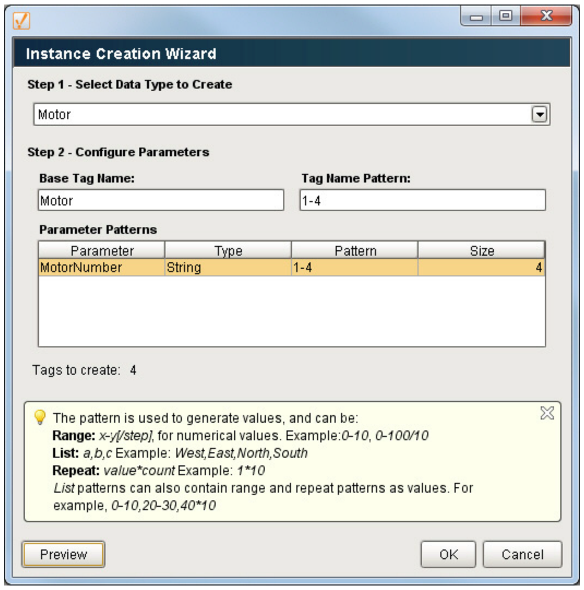


   The Instance Creation Wizard is displayed.

3. From the **Step1 - Select Data Type to Create** drop-down, select **Motor**.

4. Set the following:

   Base Tag Name: **Motor**
   Tag Name Pattern: **1-4** this creates four tags **Motor 1**, **Motor 2**, and so on
   Parameter: **MotorNumber** which is what we used to reference specific tags in the PLC
   Pattern: **1-4** so that Motor 1 tag will have a parameter of 1 and Motor 2 will have a parameter of 2 and so on



Instance Creation Wizard

Step 1 - Select Data Type to Create

Motor

Step 2 - Configure Parameters

Base Tag Name: Motor

Tag Name Pattern: 1-4

Parameter Patterns

Parameter	Type	Pattern	Size
MotorNumber	String	1-4	4

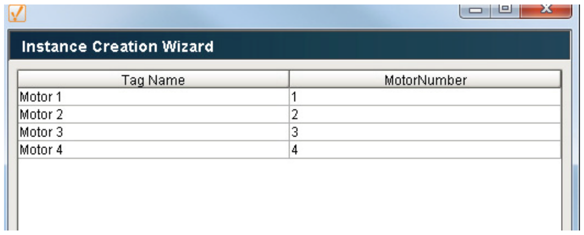
Tags to create: 4

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*

Preview OK Cancel

Basically, you have the tag name and the parameter value to set.

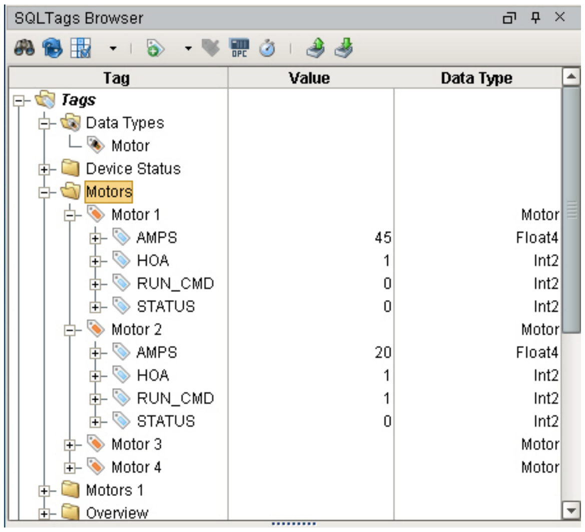
5. Click **Preview** to see what Ignition will create for the tag name and the parameter value.



Instance Creation Wizard

Tag Name	MotorNumber
Motor 1	1
Motor 2	2
Motor 3	3
Motor 4	4

6. Click **OK** to see all four tags created.
If you expand each tag, you can see all the members of the UDT and their respective values.



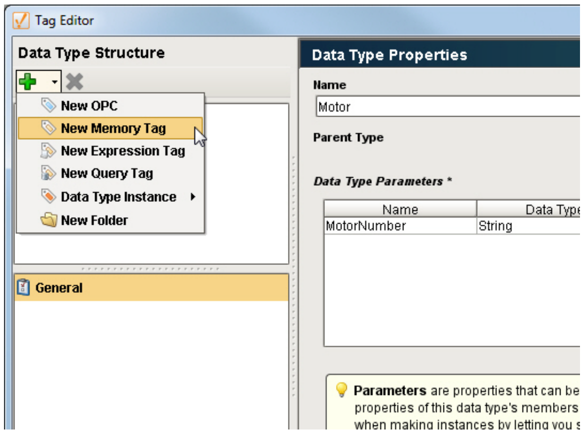
SQLTags Browser

Tag	Value	Data Type
Tags		
Data Types		
Motor		
Device Status		
Motors		
Motor 1		Motor
AMPS	45	Float4
HOA	1	Int2
RUN_CMD	0	Int2
STATUS	0	Int2
Motor 2		Motor
AMPS	20	Float4
HOA	1	Int2
RUN_CMD	1	Int2
STATUS	0	Int2
Motor 3		Motor
Motor 4		Motor
Motors 1		
Overview		

Now if you make a change to the UDT, all four instances are automatically updated because they are of that type.

7. Edit your Motor UDT by double-clicking on it in the Tags Browser.
The Tag Editor is displayed.

8. Click on the **green +** button on the top-left of window and select **New Memory Tag**.



Name: Motor

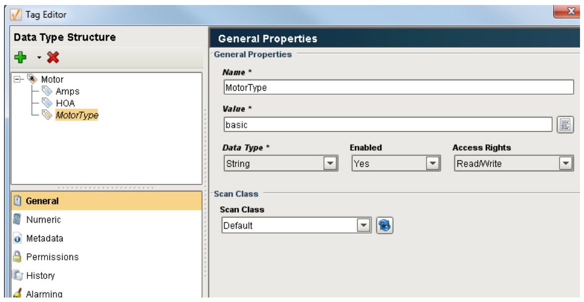
Parent Type

Data Type Parameters *

Name	Data Type
MotorNumber	String

Parameters are properties that can be
properties of this data type's members
when making instances by letting you s

9. Set the following:
Name: **MotorType**
Value: **basic**
Data Type: **String**


Tag Editor

Data Type Structure

- Motor
  - Amps
  - HOA
  - MotorType

General | Numeric | Metadata | Permissions | History | Alarming

General Properties

General Properties

Name *: MotorType

Value *: basic

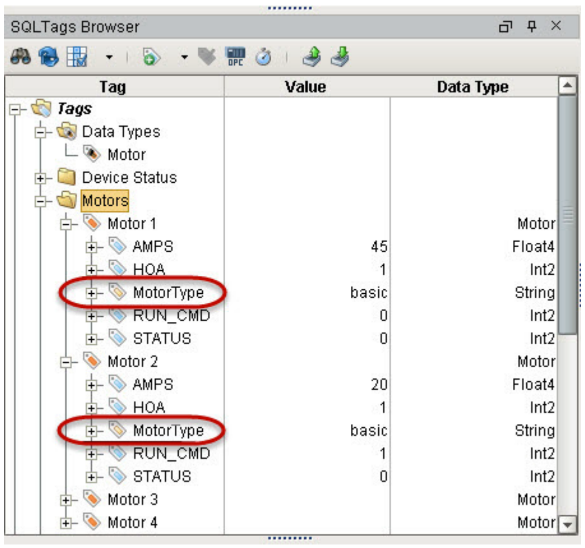
Data Type *: String | Enabled: Yes | Access Rights: Read/Write

Scan Class

Scan Class: Default

10. Click **OK** to save.

    Now each instance automatically gets the new tag.


SQLTags Browser

Tag	Value	Data Type
Tags		
Data Types		
Motor		
Device Status		
Motors		
Motor 1		Motor
AMPS	45	Float4
HOA	1	Int2
MotorType	basic	String
RUN_CMD	0	Int2
STATUS	0	Int2
Motor 2		Motor
AMPS	20	Float4
HOA	1	Int2
MotorType	basic	String
RUN_CMD	1	Int2
STATUS	0	Int2
Motor 3		Motor
Motor 4		Motor

Next ...

- Overriding Properties in UDT Instances

Overriding 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. For example, you may have to create an instance of a UDT that uses a completely different OPC address scheme from the definition.

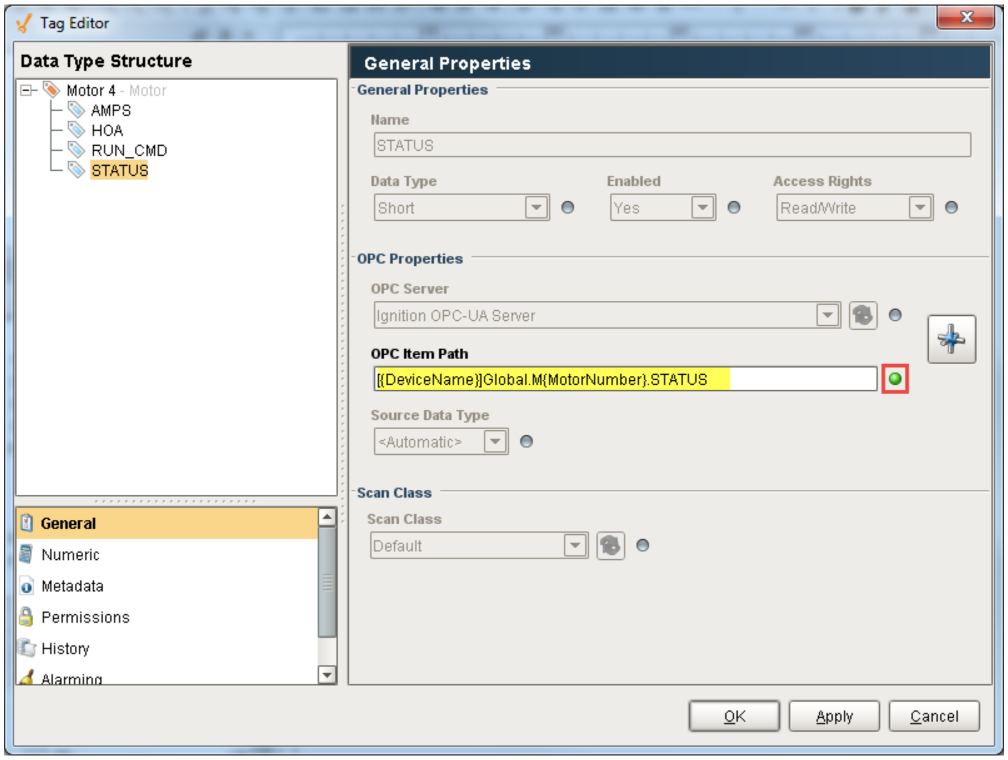
Overriding Properties in UDT Instances

Watch the Video

To override the properties

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. Lets say we want to override the **OPC Item Path** property of the **STATUS** tag of the **Motor 4** instance.

1. From the **Tag Browser**, you would go to **Tags > Data Types**, right-click on **Motor 4**, then select **Edit tag(s)**.
   The **Tag Editor** is displayed.
2. Under **Data Type Structure**, select the **STATUS** tag.
   You can see that none of the fields are highlight for which you can make any changes to them.
3. Look for the circle to the right of the **OPC Item Path** box and click it so you can highlight this box.
   The circle changes its color to green allowing you to edit the box.
4. Edit the **OPC Item Path** and click **OK**.
5. Go to **Tag Browser**, click on the **Column selection** icon on the toolbar, and select the box for **O PC Item Path**.
   See your changes in the column that displays the value of the OPC Item Path.
6. To go back to the original configuration, you can unclick the green circle next to the **OPC Item Path**.



General Properties

General Properties

Name: STATUS

Data Type: Short Enabled: Yes Access Rights: Read/Write

OPC Properties

OPC Server: Ignition OPC-UA Server

OPC Item Path: [{DeviceName}]Global.M{MotorNumber}.STATUS

Source Data Type: <Automatic>

Scan Class

Scan Class: Default

Next ...

- UDT Inheritance

UDT Inheritance

It is possible to setup UDT inheritance where data types extends to other data types, to add additional members or override default values.

INDUCTIVE UNIVERSITY

UDT Inheritance

Watch the Video

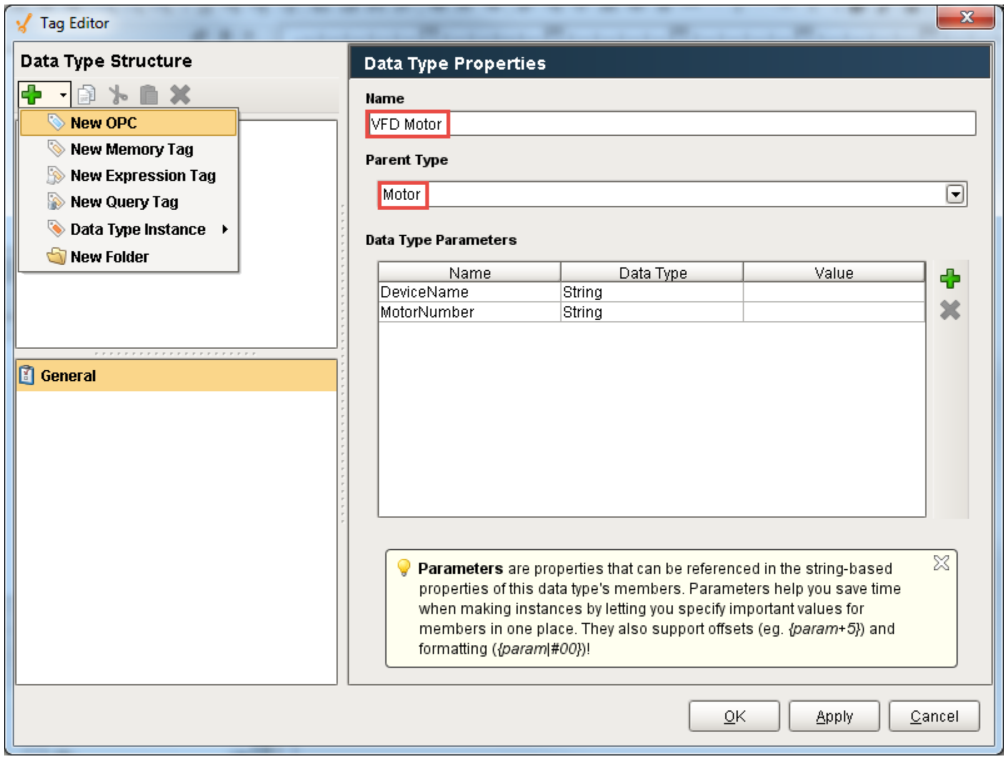
To inherit property values from an existing UDT

1. From **Tag Browser**, right-click on **Data Type**, and choose **New Data Type**.
   The Tag Editor window is displayed.

2. In the **Name** box, enter a new name, and from the dropdown of the **Parent Type** box, choose an existing UDT.
   This new UDT will automatically inherit all the properties of the parent UDT.

3. You can add new properties to this new UDT by clicking the **green +** icon on the top-left of the **T ag Editor** window.

4. Click **OK**.
   The new UDT appears in the **Tag Browser**.



Name: VFD Motor

Parent Type: Motor

Data Type Parameters

Name	Data Type	Value
DeviceName	String	
MotorNumber	String	

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param|#00}*)!

Next ...

- UDT Composition

UDT Composition

You can setup UDT composition in Ignition where you use one UDT as a parent for another, or insert one or more UDTs inside another UDT.

In this example, a Motor data type and Tank data type were already created. We want to create a third UDT called Area. The Area UDT will contain the Motor and Tank data types inside of it.

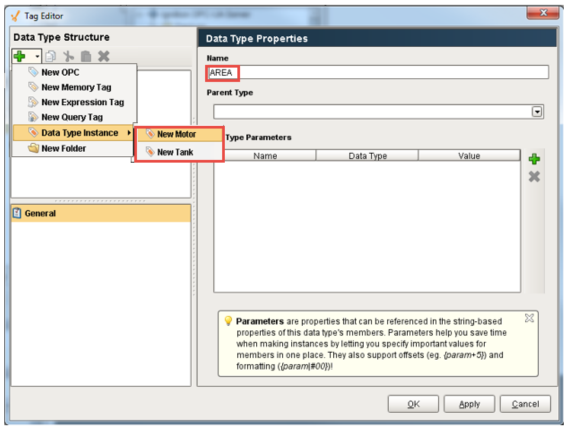
INDUCTIVE UNIVERSITY

UDT Composition

Watch the Video

1. In the **Tag Browser**, right click on the **Data Types folder** and select **New Data Type**.
   In the **Name box**, enter **Area**.

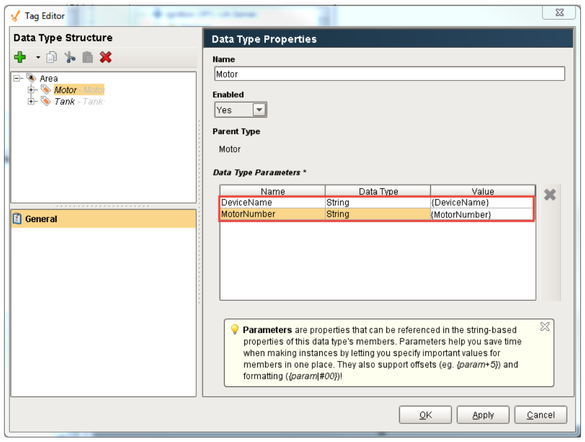
2. In the **Tag Editor**, add 2 data type instances, one for Motor and the other for Tank.
   Click on the **green + icon**, select **Data Type Instance > New Motor**.
   Click on the **green + icon**, select **Data Type Instance > New Tank**.
   Rename both new data types to Motor and Tank respectively.



Name: AREA

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param|#00}*)!

3. Every UDT you add inside another UDT has paramters that need to be specified and passed in from the parent UDT. Motor has 2 parameters: DeviceName and MotorNumber. Tank has 2 parameters: DeviceName and TankNumber.



Name: Motor

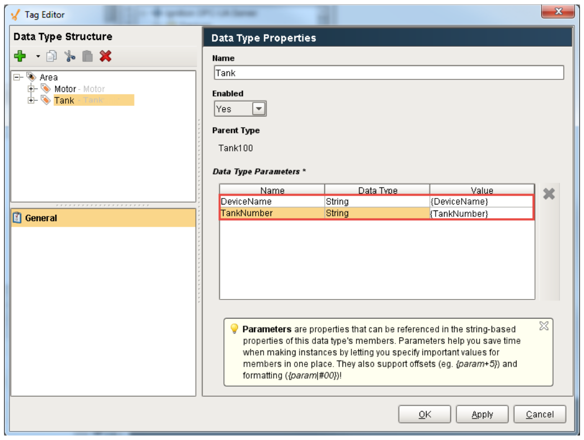
Enabled: Yes

Parent Type: Motor

Data Type Parameters *

Name	Data Type	Value
DeviceName	String	{DeviceName}
MotorNumber	String	{MotorNumber}

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param|#00}*)!



Data Type Properties

Name: Tank

Enabled: Yes

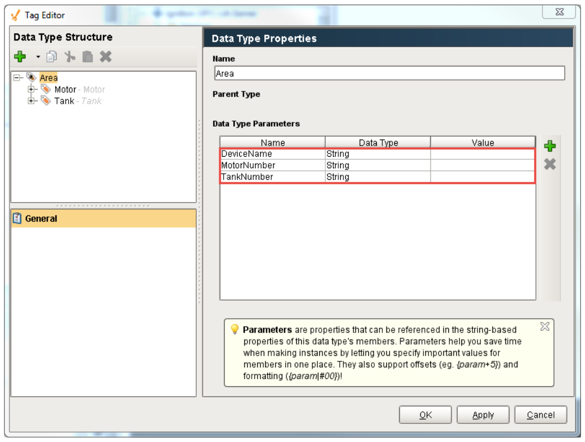
Parent Type: Tank100

Data Type Parameters *

Name	Data Type	Value
DeviceName	String	{DeviceName}
TankNumber	String	{TankNumber}

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param|#00}*)!

4. If both data types use the same parameter as in this example with the DeviceName parameter, you only need to enter it once in the new UDT.
   In the **Tag Editor**, click **Area** and enter the parameter from the Motor and Tank data types: **Devi ceName**, **MotorNumber** and **TankNumber**.
   Click **OK**.



Data Type Properties

Name: Area

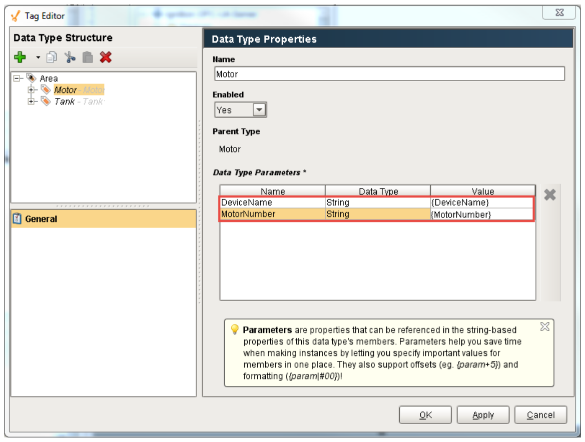
Parent Type

Data Type Parameters

Name	Data Type	Value
DeviceName	String	
MotorNumber	String	
TankNumber	String	

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param|#00}*)!

5. The values in the Area UDT need to pass to the Motor and Tank data types so references need to be added.
   In the **Tag Editor**, click on **Motor** and enter the references.
   For the parameter DeviceName, enter the value **{DeviceName}.**
   For the parameter MotorNumber, enter the value **{MotorNumber}.**
   Click **OK**.



Data Type Properties

Name: Motor

Enabled: Yes

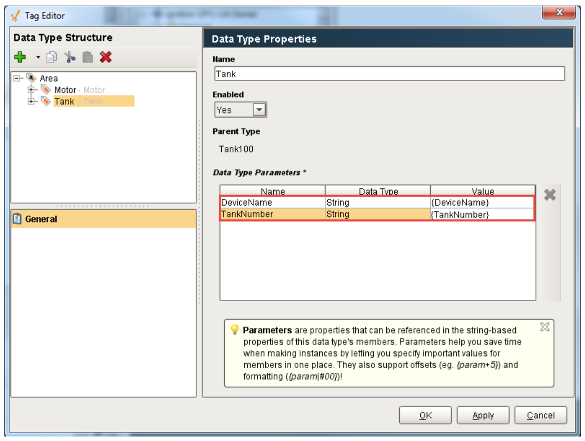
Parent Type: Motor

Data Type Parameters *

Name	Data Type	Value
DeviceName	String	{DeviceName}
MotorNumber	String	{MotorNumber}

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param|#00}*)!

In the **Tag Editior**, click on **Tank** and enter the references:
For the DeviceName enter the value **{DeviceName}.**
For the parameter TankNumber enter the value **{TankNumber}.**
Click **OK**.


Tag Editor

Data Type Structure

- Area
  - Motor - Motor
  - Tank - Tank

General

Data Type Properties

Name
Tank

Enabled
Yes

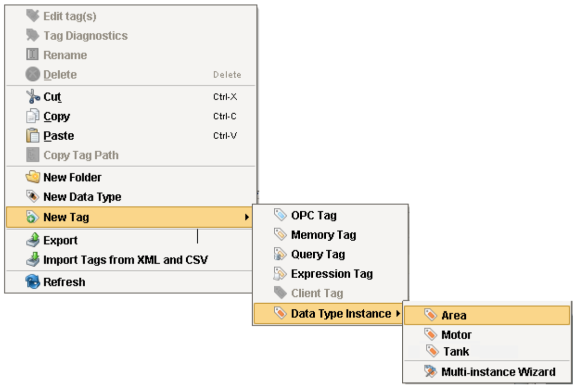
Parent Type
Tank100

Data Type Parameters *

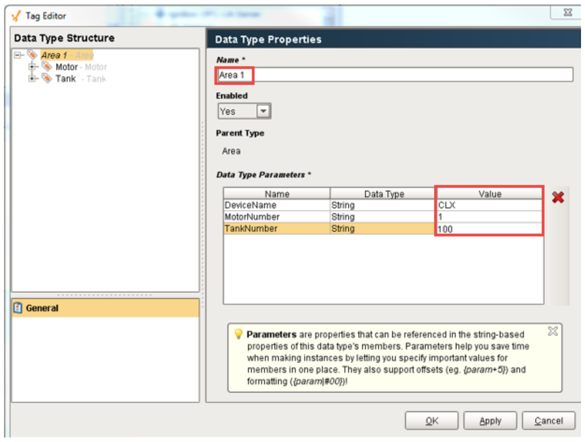
Name	Data Type	Value
DeviceName	String	{DeviceName}
TankNumber	String	{TankNumber}

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param\|#00}*)!

6. Now, create a new instance of Area. In the **Tag Browser**, right click on the **Tags folder**, select **New Tag > New Data Type Instance > Area**.



In the **Name box**, enter the new instance name, **Area 1**. Enter the values for each parameter:
DeviceName, enter **CLX**.
MotorNumber, enter **1**.
TankNumber, enter **100**.
Click **OK**.


Tag Editor

Data Type Structure

- *Area 1* - *Area*
  - Motor - Motor
  - Tank - Tank

General

Data Type Properties

Name *
Area 1

Enabled
Yes

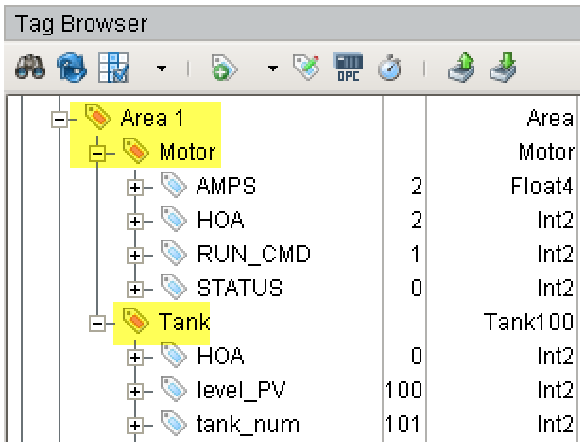
Parent Type
Area

Data Type Parameters *

Name	Data Type	Value
DeviceName	String	CLX
MotorNumber	String	1
TankNumber	String	100

Parameters are properties that can be referenced in the string-based properties of this data type's members. Parameters help you save time when making instances by letting you specify important values for members in one place. They also support offsets (eg. *{param+5}*) and formatting (*{param\|#00}*)!

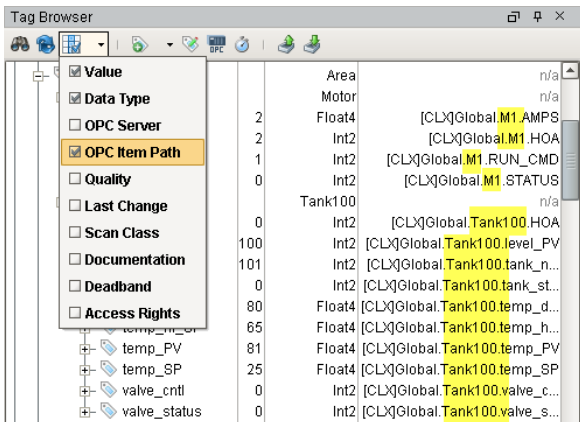
7. From the **Tag Browser**, you can see **Area 1** was created. Expand Motor and Tank data types to validate the values are pointing to the appropriate places in the PLC.



Tag Browser

Area 1		Area
Motor		Motor
AMPS	2	Float4
HOA	2	Int2
RUN_CMD	1	Int2
STATUS	0	Int2
Tank		Tank100
HOA	0	Int2
level_PV	100	Int2
tank_num	101	Int2

8. You can also verify the path in the Tag Tree that Motor is pointing to M1 and Tank is pointing to Tank 100.
   From the **Tag Browser**, click **Area 1** and check the **OPC Item Path** from the dropdown.   .



Tag Browser

- ☑ Value
- ☑ Data Type
- ☐ OPC Server
- ☑ OPC Item Path
- ☐ Quality
- ☐ Last Change
- ☐ Scan Class
- ☐ Documentation
- ☐ Deadband
- ☐ Access Rights

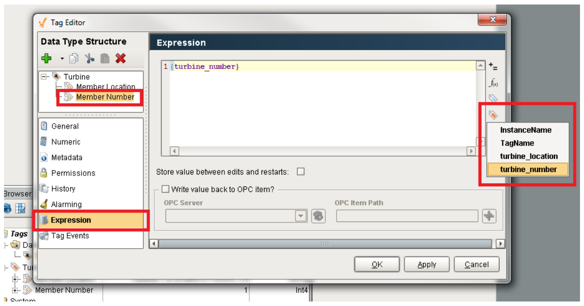
		Area	n/a
		Motor	n/a
	2	Float4	[CLX]Global.M1.AMPS
	2	Int2	[CLX]Global.M1.HOA
	1	Int2	[CLX]Global.M1.RUN_CMD
	0	Int2	[CLX]Global.M1.STATUS
		Tank100	n/a
	0	Int2	[CLX]Global.Tank100.HOA
	100	Int2	[CLX]Global.Tank100.level_PV
	101	Int2	[CLX]Global.Tank100.tank_n...
	0	Int2	[CLX]Global.Tank100.tank_st...
	80	Float4	[CLX]Global.Tank100.temp_d...
	65	Float4	[CLX]Global.Tank100.temp_h...
temp_PV	81	Float4	[CLX]Global.Tank100.temp_PV
temp_SP	25	Float4	[CLX]Global.Tank100.temp_SP
valve_cntl	0	Int2	[CLX]Global.Tank100.valve_c...
valve_status	0	Int2	[CLX]Global.Tank100.valve_s...

Next ...

- add link here

Data Type Parameters in Expressions

It is possible to use the value of data type parameters directly in expression bindings within a UDT.



```
{turbine_number}
```

Parameter references can be quickly inserted into expression by clicking on the UDT Parameters button on the right of the expression area

Syntax

The syntax for data type parameters expressions differs from tag and property references when using string values. **Quotation marks must be used when referencing string parameters**. The reason for this is parameter references are replaced before any expressions are evaluated. This differs from tag and property references seen elsewhere in ignition.

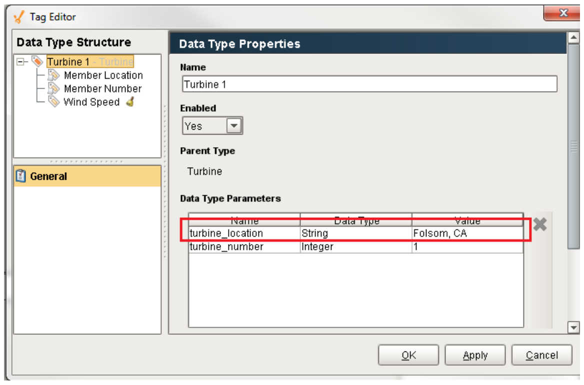
If the data type of the parameter is not a string, then quotation marks are not required, but can be used.

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.

```
"{string parameter}" //This expression will evaluate successfully
'{string parameter}' //This expression will evaluate successfully as well
```

Examples

Below is an instance of a Turbine UDT. It contains a string parameter named "turbine_location".



Name: Turbine 1

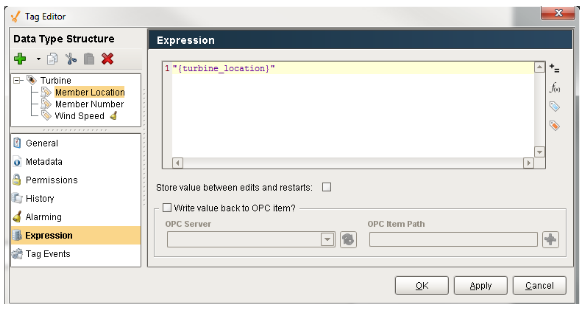
Enabled: Yes

Parent Type: Turbine

Data Type Parameters

Name	Data Type	Value
turbine_location	String	Folsom, CA
turbine_number	Integer	1

Inside the Turbine UDT is an expression tag named Member Location. To show the value of the turbine_location parameter on Member Location, the following expression would be used on the UDT definition

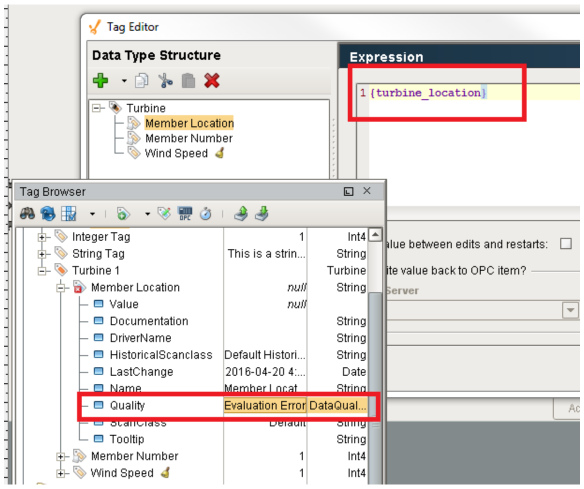


```
"{turbine_location}"
```

Store value between edits and restarts: ☐

☐ Write value back to OPC item?

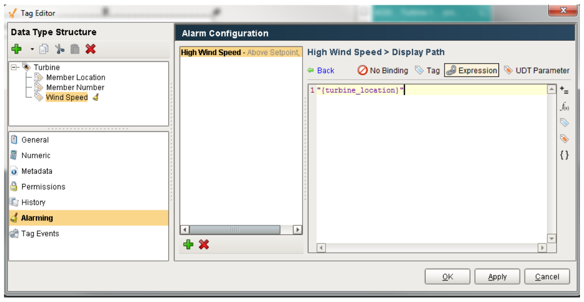
Omitting the quotation marks would result in an evaluation error:



```
{turbine_location}
```

Integer Tag	1	Int4
String Tag	This is a strin...	String
Turbine 1		Turbine
Member Location	null	String
Value	null	
Documentation		String
DriverName		String
HistoricalScanclass	Default Histori...	String
LastChange	2016-04-20 4:...	Date
Name	Member Locat...	String
Quality	Evaluation Error	DataQual...
ScanClass	Default	String
Tooltip		String
Member Number	1	Int4
Wind Speed	1	Int4

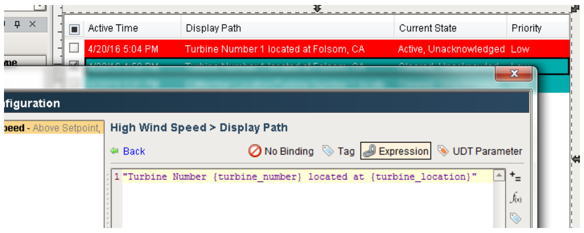
The same syntax should be applied to parameters in bindings on alarm properties. Below is an example using the same UDT, but the expression is instead located on the Display Path property of an alarm.



High Wind Speed > Display Path

```
"{turbine_location}"
```

Additionally, non-string parameters and static values can be included in the same quotation marks.



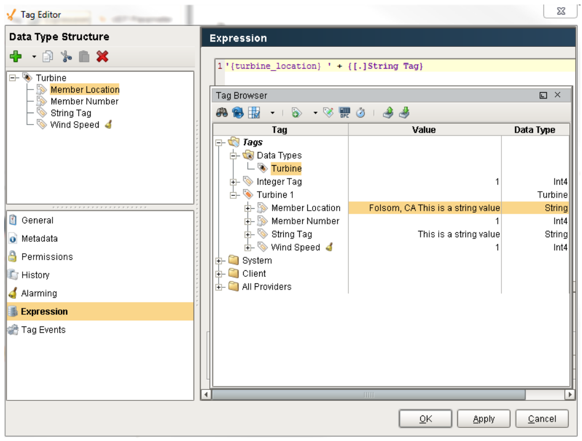
Active Time	Display Path	Current State	Priority
4/20/16 5:04 PM	Turbine Number 1 located at Folsom, CA	Active, Unacknowledged	Low

High Wind Speed > Display Path

```
"Turbine Number {turbine_number} located at {turbine_location}"
```

Combining Parameters and Tag references

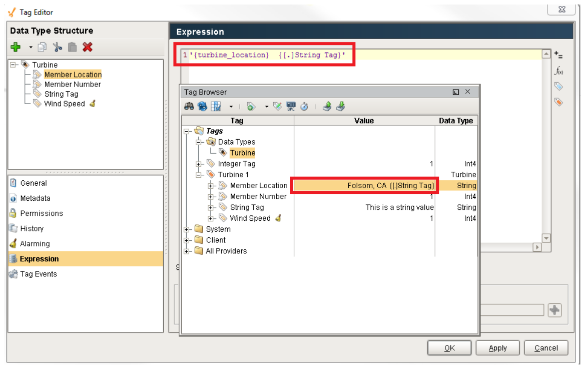
Because parameter and tag references differ in syntax, some consideration must be made when attempting to use both in the same expression. Tag references must not be placed inside of quotes. After adding a string tag to the Turbine UDT, a reference to the tag can be added to Member Location's expression:



```
'{turbine_location} ' + {[.]String Tag}
```

Tag	Value	Data Type
Tags		
Data Types		
Turbine		
Integer Tag	1	Int4
Turbine 1		Turbine
Member Location	Folsom, CA This is a string value	String
Member Number	1	Int4
String Tag	This is a string value	String
Wind Speed	1	Int4
System		
Client		
All Providers		

Adding quotation marks around the tag reference would result in the literal value of the reference appearing in the value of the tag:



```
'{turbine_location}  {[.]String Tag}'
```

Tag	Value	Data Type
Tags		
Data Types		
Turbine		
Integer Tag	1	Int4
Turbine 1		Turbine
Member Location	Folsom, CA {[.]String Tag}	String
Member Number	1	Int4
String Tag	This is a string value	String
Wind Speed	1	Int4
System		
Client		
All Providers		

Quick Reference

Reference Type	Require Quotes?	Expression Example
Static String	Requires Quotes	`"This is a string"`
String UDT Parameter	Requires Quotes	`"{turbine_location}"`
String Tag Reference	No Quotes	`{[.]String Tag}`
String UDT Parameter with Static String	Requires 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}"`

Next ...

- Expression Overview and Syntax

Scan Classes

Scan classes dictate the rate of execution of tags, and therefore play a crucial role in the design of large and high-performance systems. It will often make sense to have more than one scan class as usually not all of your tags will need to be subscribed at the same rate. Some tags you may wish to see updated at 250-500ms, while others may not be so crucial and may only need to be subscribed at 1500ms.

Creating different scan classes allow you to organize your tags into groups that subscribe at different rates. It is good practice to put some forethought and planning into the organization of your Tags and scan classes.

On this page ...

Tag Execution by Scan Class

Tags are executed by scan classes inside of a tag provider. In a typical system, there are a number of scan classes and one or two tag providers: the internal tag provider and possibly an external tag provider.

- **Internal provider** keeps the tag configuration in the project.
- **External provider** stores tag configuration and values in a database.

Tags stored in an external provider are available to all Ignition installations that have access to that database. One of the installations can be specified as the tag's driver. The driving system has a copy of the scan class that it executes, which in turn evaluates the tag. The value is stored to the database and all of the other installations are notified of the new value.

There are several key execution modes to help create efficient projects. The different execution modes are as follows:

- **Direct** executes based on the slow rate setting at a fixed rate.
- **Driven** executes based on a condition.
- **Driven one-shot** executes only once when a condition is true.
- **Leased** executes according to on-demand polling.

Adding and Editing Scan Classes

You can add and edit scan classes by clicking on the **Edit Scan Classes** icon in the **Tag Browser**. The **Scan Class Editor** window is displayed showing a list of already configured Scan Classes on the left, and configuration settings on the right. To add a scan class, simply click the **green +** icon on the lower-left of the **Scan Class Editor** and then specify the name and the properties for it.

Scan Class Editor

Edit Selected Scan Class

Scan Class Name	Unique name of the scan class.
Mode	**Direct** The scan class executes at a fixed rate, defined by the slow rate setting. **Leased** The scan class executes at the fast rate when any of the tags it contains are subscribed and visible in a Client window. If no tags are subscribed, the scan class runs at the slow rate. **Driven** The rate of the scan class 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 scan class 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, the scan class 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 scan class triggering.
Slow Rate	Base update rate, specified in milliseconds, at which tags will be executed. **Note:** If the rate is set to 0, the scan class will not be executed. It is common for leased and driven modes to use 0 as a slow rate in order to achieve an **on/off** effect.
Fa	

st Rate	Used by the Driven and Leased modes, this is the faster rate that the tags will be executed at when those modes are active. **Note:** If the rate is set to 0, the scan class will not be executed.
Stale Timeout	How long to wait before the tags in the scan class are determined to be stale (not running). This is calculated off of the last expected execution time of the scan class, and is particularly important for scan classes executed by other drivers through the external Tags provider. This property is not used by internal providers.
Driven Properties **Used by the driven mode to determine when the scan class should run at the fast rate.**	
Driving Tag	Which tag will determine when the rate of the scan class should be changed. The Driving Tag should not use this same scan class. The Driving Tag should be on a separate scan class that runs at a faster rate than the Driven scan class. This means that changes to the Driving Tag will be recognized by the system faster, and the change in rate on the Driven scan class will occur faster.
Operator	How the 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 Scan Class. Otherwise, the Slow Rate will be used. The **Any Change** operator works differently than the other operators: The scan class will execute immediately whenever the driving tag changes value. Using the Any Change operator means that the scan class no longer uses the Slow Rate or Fast Rate properties.
Value	Used by the Operator property to determine if the Scan Class should execute at the slow or fast rate.
One-shot execution	**One-shot** will execute once when the comparison condition is true, and not again until the condition become false, and subsequently true.
Advanced Properties **Settings that subtly affect how the scan class operates.**	
OPC Data Mode	This mode dictates how OPC values are obtained. The default mode, **Subscription**, is generally recommended. **Subscribed** All OPC tags in the scan class will be subscribed according to the scan class rate. Values will come in asynchronously as they change. **Read** Tags will not be subscribed, but will instead be synchronously read each time the scan class 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.
OPC Read After Write	The following feature is new in Ignition version **7.8.0** Click here to check out the other new features 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 scanclasses as it the tags will show the latest value quicker than the normal execution would allow.

Historical Scan Classes

Historical scan classes are simply standard scan classes used by Tags to store history. By using separate scan classes 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 scan classes, it is recommended that you create separate scan classes for each purpose and name them in a manner that indicates their usage. It is common to modify scan classes 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.

In this section ...

Types of Scan Classes

Scan classes dictate the execution schedule for tags, and therefore play a crucial role in the design of large, high-performance systems. Every tag has a scan class and that dictates how often that value is polled from a OPC location, how often an expression tag calculates its expression, or how often a query tag runs its query.

It is very easy to configure scan classes and there are four different types of scan classes you can use. From the **Tag Browser**, click on the **Edit Scan Class** icon to open the **Scan Class Editor.** You can create new scan classes here and assign one of four types of scan classes.

The four different scan classes that specify the rate of tag execution are as follows: Direct, Driven, Driven one-shot, and Leased. You specify these scan class types from the **Mode** setting on the **Scan Class Editor**.

Types of Scan Classes

Watch the Video

Direct

The Direct scan classe executes based on the slow rate setting at a fixed rate. Every tag that uses the direct scan class will poll the PLC at the slow rate setting 24x7x365 (at all times).

Driven

The Driven scan class executes based on a condition. The rate of the Driven scan class is based on the value of a driving tag which provides the condition. The condition is a simple comparison between a tag value and a number. If the condition is true, the scan class executes at the fast rate. If false, it runs at the slow rate.

There are two exceptions to this: the **Any Change** operator and **one-shot** mode. Using either of these, the scan class does not run at a rate. Instead, it will be triggered by a change in the driving tag's value. Any change will execute each time the 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.

It's useful to 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 scan class triggering.

Every tag that uses the driven scan class, will poll the PLC at the fast rate when the condition is true and the slow rate when the condition is false.

Driven One-shot

The Driven one-shot executes only once when a condition is true.

Leased

The Leased executes according to on-demand polling.

Next ...

- Setting a Tag's Scan Class

Setting a Tag's Scan Class

Each tag in Ignition is assigned a scan class which dictates the polling rate and conditions on which the tag will be evaluated. You can specify both a realtime scan class and a historical scan class for each tag.

The four different scan classes that specify the rate of execution of tags are as follows:

- Direct executes based on the slow polling rate setting at a fixed rate.
- Driven executes based on a condition.
- Driven one-shot executes only once when a condition is true.
- Leased executes according to on-demand polling.

INDUCTIVE UNIVERSITY

Setting a Tag's Scan Class

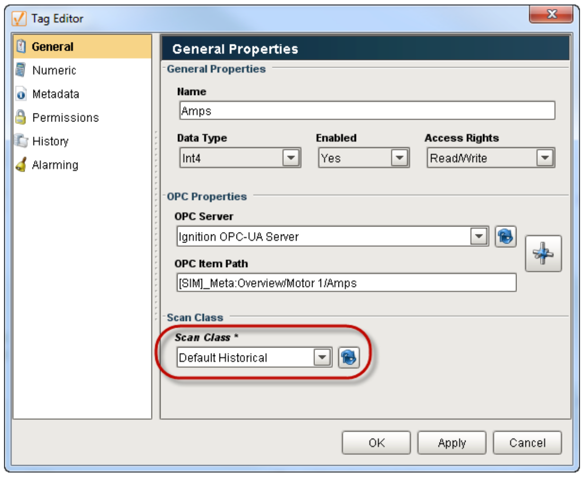
Watch the Video

To set the scan class for a tag

1. Right-click on any tag and select **Edit tag(s)**.
   The **Tag Editor** window is displayed.

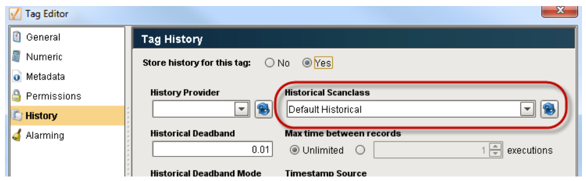
2. On the **General** tab, choose a scan class from the **Scan Class** drop-down for the selected tag (s).
   For example, choose the **Default Historical** scan class.

3. Click **OK** to save it.



Name: Amps
Data Type: Int4
Enabled: Yes
Access Rights: Read/Write
OPC Server: Ignition OPC-UA Server
OPC Item Path: [SIM]_Meta:Overview/Motor 1/Amps
Scan Class *: Default Historical

You will immediately see the tag updating at a different rate. The tag also uses the scan class to determine how fast to log data for the historian. The history scan class doesn't affect how fast the values get polled from the PLC, but affects how fast the data gets logged. You can select the history scan class on the **History** tab.



Store history for this tag: No [✓ Yes]
History Provider:
Historical Scanclass: Default Historical
Historical Deadband: 0.01
Max time between records: [✓ Unlimited] 1 executions
Historical Deadband Mode
Timestamp Source

Next ...

- Direct Scan Class

Direct Scan Class

The direct scan class executes at a fixed rate, which is defined by the slow polling rate setting. Every tag that uses the direct scan class will poll the PLC at the slow rate setting 24x7x365 (at all times).

Ignition comes with two default direct scan classes called **Default** and **Default Historical**. You can add new scan classes.

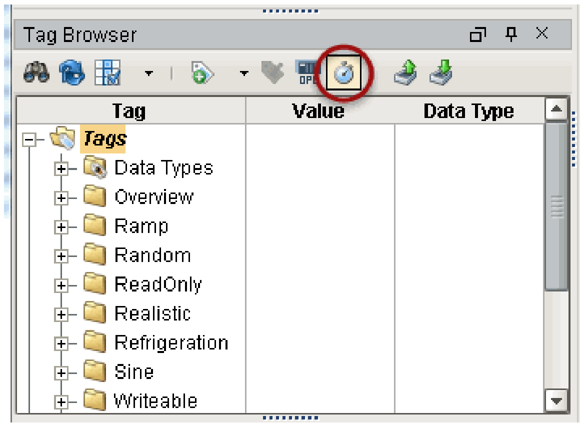
INDUCTIVE UNIVERSITY

Direct Scan Class

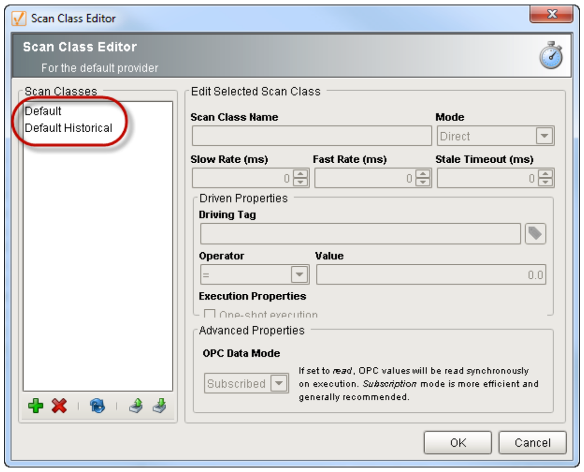
Watch the Video

To add a direct scan class

1. Open the **Designer**, in the **Tag Browser**, click on the **Timer** icon.



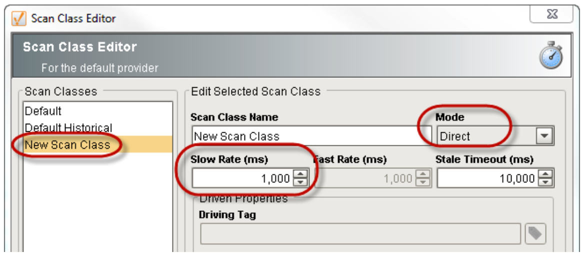
The **Scan Class Editor** is displayed. On the left of the window, you can see the default scan classes: **Default** and **Default Historical**. They are both direct scan classes. The **Default** scan class is a 1 second rate and the **Default Historical** is a 10 second rate scan class.



If set to *read*, OPC values will be read synchronously on execution. *Subscription* mode is more efficient and generally recommended.

2. Click the **green +** icon on the bottom-left of the window to create a new scan class.

3. From the **Mode** drop-down select **Direct** and set **Slow Rate (ms)** to the number of milliseconds you want to poll at.



Scan Class Name: New Scan Class
Mode: Direct
Slow Rate (ms): 1,000
Fast Rate (ms): 1,000
Stale Timeout (ms): 10,000

4. Click **OK**.
   That’s it! You have created a new direct scan class.

Next ...

- Driven Scan Class - Machine State

Driven Scan Class - Machine State

Driven scan classes set the polling rate (Fast Rate and Slow Rate) dynamically for tags based on a given condition, which is actually a logical comparison. Machine states being tied to a Boolean tag, as described below, is one example of a condition where the logic comparison is true when the machine state is ON. In simple cases, when the logic comparison is true, the Fast Rate applies, otherwise the Slow Rate applies.

INDUCTIVE UNIVERSITY

Driven Scan Class - Machine State

Watch the Video

To add a driven scan class based on the machine state

Lets add a driven scan class that updates the tags based on when the machine is ON.

1. In the **Scan Class Editor**, create a new scan class by clicking on the **green +** icon on the lower-left of the window.
2. Name the scan class, for example, you can name it **Driven Machine State**, and set the **Mode** to **Driven**.
   This scan class executes based on the condition you set on the **Driving Tag**.
3. Click on the **tag** icon next to the **Driving Tag** box, and choose the tag you want to set the condition on.
4. Set **Operator** to **=** and the **Value** to **1**.
5. Set the **Slow Rate** to 10,000 (10sec) and the **Fast Rate** to 1000 (1sec).
   When the machine is ON it will use the Fast Rate, when the machine is OFF it will use the Slow Rate.
6. Click **OK**.
   Now in the **Tag Browser** you will see a check box next to the tag for which you have set the machine state of ON and OFF on it.

Next ...

- Driven Scan Class - Manual Trigger

Driven Scan Class - Manual Trigger

Driven scan classes can be used to allow operators to speed up or slow down polling rates on demand based on another trigger value.

To add a driven scan class based on a trigger value

First we need to create the Trigger tag.

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:
   Name: **Trigger**
   Value: **false**
   Data Type: **Boolean**
3. Click **OK** to add the **Trigger** tag to the **Tag Browser**.

Now lets add a driven scan class that updates the tags based on the value of the Trigger tag we just created.

1. In the **Scan Class Editor**, create a new scan class by clicking on the **green +** icon on the lower-left of the window.
2. Name the scan class, for example, you can name it **Driven Manual**, and set the **Mode** to **Driven**.

   This scan class executes based on the condition you set on the **Driving Tag**.
3. Click on the **tag** icon next to the **Driving Tag** box, and select the **Trigger** tag you want to set the condition on.
4. Set **Operator** to **=** and the **Value** to **1**.
5. Set the **Slow Rate** to 4000 (4sec) and the **Fast Rate** to 500 (1/2sec, a very fast rate).
   When the Trigger is equal to 1, it will use the **Fast Rate** and you can see the values changing very fast. When the Trigger is equal to 0, it will use the **Slow Rate**.
6. Click **OK**.
   Now in the Tag Browser you will see a check box next to the Trigger tag for which you have set the on/off condition on it.

INDUCTIVE UNIVERSITY

Driven Scan Class - Manual Trigger

Watch the Video

Next ...

- Driven Scan Class - Time of Day

Driven Scan Class - Time of Day

Driven scan classes 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.

INDUCTIVE UNIVERSITY

Driven Scan Class - Time of Day

Watch the Video

To add a driven scan class

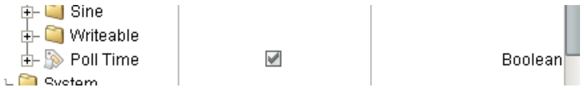
Let's add a driven scan class that updates the tags based on a time of day (using an expression). You can also use other tag values to drive the condition, for example when a machine is on.

First, we need a tag that provides the condition.

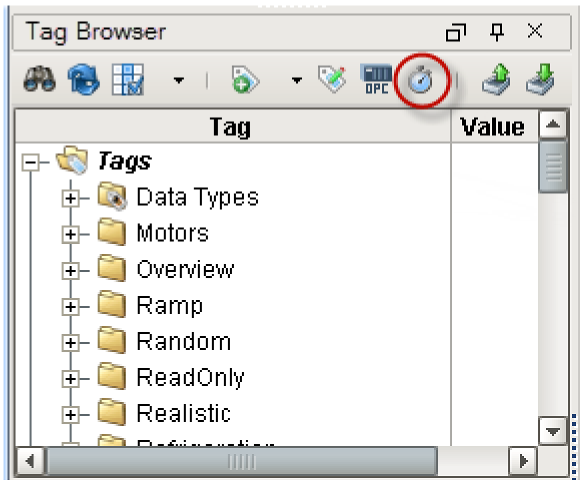
1. Create an **Expression** Tag in the root folder of **Tags** and call it **Poll Time** that is a Boolean with the following expression:

   ```
   dateExtract(now(0), "hour") >= 8 &&
   dateExtract(now(0), "hour") <= 17
   ```

   The value will be true when the time is between 8am and 5pm and false otherwise. So with that tag we can poll the PLC values during that time.

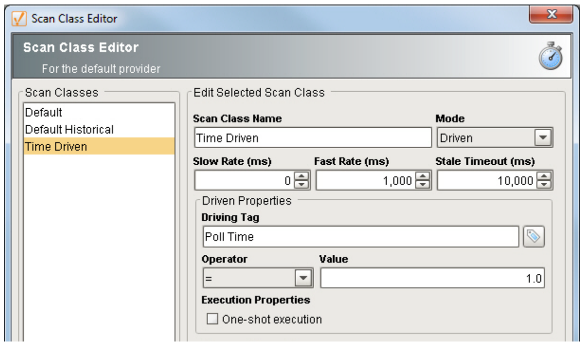
   

2. Click on the **Timer** icon in the **Tag Browser** to open the Scan Class Editor.

3. Click on the **green +** icon to add a new scan class and set the following:

   Scan Class Name: **Time Driven**
   Mode: **Driven**
   Slow Rate (ms): **0** so we don't poll when the condition is false. If you set the value to 10 seconds it simply would just poll slower when the condition is false.
   Fast Rate (ms): **1000**
   Driving Tag: **Poll Time** expression tag and set the condition Operator:**=** and Value: **1.0**.


Scan Class Editor

Scan Class Editor
For the default provider

Scan Classes
- Default
- Default Historical
- Time Driven

Edit Selected Scan Class

Scan Class Name: Time Driven
Mode: Driven
Slow Rate (ms): 0
Fast Rate (ms): 1,000
Stale Timeout (ms): 10,000

Driven Properties
Driving Tag: Poll Time
Operator: =
Value: 1.0

Execution Properties
☐ One-shot execution

4. Click **OK** to save.
   Set the scan class on any tag to this one to update on this condition. If you use this scan class for history, the tag will only store history during those times.

Next ...

- Driven Scan Class - One shot

Driven Scan Class - One shot

Driven scan classes can be configured as a **one shot**. The One-Shot polling rate executes once when the comparison condition is true, and not again until the condition become false, then subsequently true.

The rate of the driven one-shot scan class is based on the value of a driving tag which provides the condition. The condition is a simple comparison between a tag value and a number. If the condition is true, the scan class executes at the fast rate. If false, it runs at the slow rate. There are two exceptions to this: the any change operator and one-shot mode. Using either of these, the scan class will not run at a rate. Instead, it will be triggered by a change in the driving tag's value. One-shot will execute once when the comparison condition is true, and not again until the condition become false and subsequently true.

Driven Scan Class - One shot

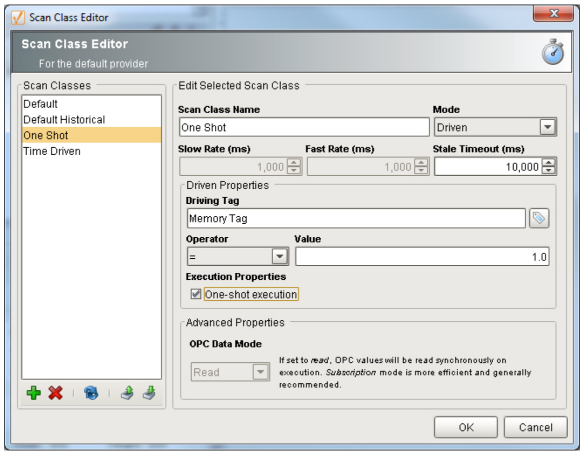
[Watch the Video]

To make a one-shot scan class

Let's make a one-shot driving scan class that only updates once when a memory tag goes to 1. We are using a memory tag so a user can change the value when they want to poll the PLC. Essentially, we can have a refresh button on the screen that allows the operator to poll the PLC once for the new values.

1. In the **Scan Class Editor**, click on the **green +** icon to add a new scan class.
2. Set the following:

   Scan Class Name: **One Shot**
   Mode: **Driven**
   Don't worry about the slow and fast rate since they are not used
   Driving Tag: set to a memory tag (in this example **Memory Tag**) and set the condition Operator: **=** and Value: **1.0**
3. Check the **One-shot execution** box.



Scan Class Editor
Scan Class Editor
For the default provider

Scan Classes
Default
Default Historical
One Shot
Time Driven

Edit Selected Scan Class
Scan Class Name: One Shot
Mode: Driven
Slow Rate (ms): 1,000
Fast Rate (ms): 1,000
Stale Timeout (ms): 10,000
Driven Properties
Driving Tag: Memory Tag
Operator: =
Value: 1.0
Execution Properties
[✓] One-shot execution
Advanced Properties
OPC Data Mode: Read
If set to *read*, OPC values will be read synchronously on execution. *Subscription* mode is more efficient and generally recommended.

4. Click **OK** to save.

5. Set the value of the memory tag to 1 and you will see the values update once. Set the value to 0 and back to 1 to see them update again.

Next ...

- Leased Scan Class

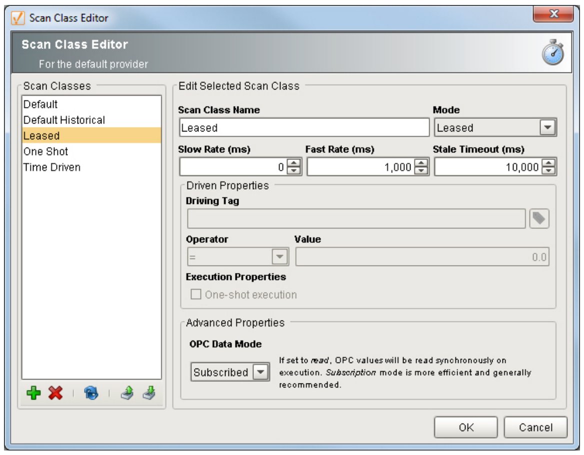
Leased Scan Class

Tags set to the **Leased** scan class execute/poll the PLC at the fast polling rate when they are subscribed and visible in a Client window, that is, when a Client needs the value. If the tags are not subscribed and the window or client is closed, the scan class runs at the slow polling rate.

To add a leased scan class

Let's add a new leased scan class that updates the tag at a 1 second rate when a client needs the tag and doesn't poll when the client is closed.

1. On the **Scan Class Editor**, click on the **green +** icon to add a new scan class.
2. Set the following:
   Scan Class Name: **Leased**
   Mode: **Leased**
   Slow Rate: **0**
   Fast Rate: **1000**



Scan Class Editor
For the default provider

Scan Classes: Default, Default Historical, Leased, One Shot, Time Driven

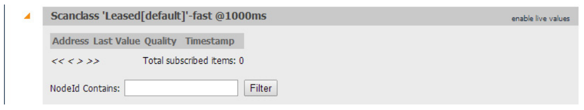
Edit Selected Scan Class
Scan Class Name: Leased
Mode: Leased
Slow Rate (ms): 0
Fast Rate (ms): 1,000
Stale Timeout (ms): 10,000

Driven Properties
Driving Tag:
Operator: =
Value: 0.0
Execution Properties
One-shot execution

Advanced Properties
OPC Data Mode: Subscribed
If set to *read*, OPC values will be read synchronously on execution. *Subscription* mode is more efficient and generally recommended.

3. Click **OK** to save.
4. Set the scan class on the tags to use this new **Leased** scan class.

   The tags are only polled when a component that is bound to it is showing in a Client.
5. To verify that Ignition is working the way it should, go to the **Status** tab of the Ignition Gateway and click on the **OPC Connections** link on the left side.
   If nothing is active, the list is empty for the **Leased** scan class.



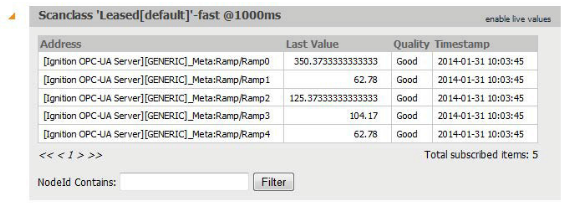
Scanclass 'Leased[default]'-fast @1000ms

Address	Last Value	Quality	Timestamp

<< < > >>   Total subscribed items: 0

NodeId Contains:

As tags are needed, the list will fill in with only the tags that are currently being looked at.



Scanclass 'Leased[default]'-fast @1000ms

Address	Last Value	Quality	Timestamp
[Ignition OPC-UA Server][GENERIC]_Meta:Ramp/Ramp0	350.3733333333333	Good	2014-01-31 10:03:45
[Ignition OPC-UA Server][GENERIC]_Meta:Ramp/Ramp1	62.78	Good	2014-01-31 10:03:45
[Ignition OPC-UA Server][GENERIC]_Meta:Ramp/Ramp2	125.37333333333333	Good	2014-01-31 10:03:45
[Ignition OPC-UA Server][GENERIC]_Meta:Ramp/Ramp3	104.17	Good	2014-01-31 10:03:45
[Ignition OPC-UA Server][GENERIC]_Meta:Ramp/Ramp4	62.78	Good	2014-01-31 10:03:45

<< < 1 > >>   Total subscribed items: 5

NodeId Contains:

INDUCTIVE UNIVERSITY

Leased Scan Class

Watch the Video

Next ...

- OPC-UA Stale Threshold

OPC-UA Stale Threshold

Ignition uses the OPC-UA **Stale Threshold** setting to determine when updates from a driver have taken too long to complete. This time period will be calculated as the fastest sampling rate for that node multiplied by the **Stale Threshold** setting value.

1. Go to the **Configure** section of the Gateway, and then go to **OPC-UA > Settings**.
   The **Settings** page is displayed.

2. Scroll down to the **Other** section and then to the **Stale Threshold** setting.
   This is a multiplier by which the server determines that updates from a driver have become stale.

OPC-UA Stale Threshold

Watch the Video

Next ...

- add link here

System, Client, and Diagnostic Tags

Here is a brief description on each of the System, Client, and Diagnostic tags.

System Tags

System tags provide status about the system, such as memory usage, performance metrics, and so on. They exist for the Client and the Gateway. You can modify the Gateway system tags to perform alarming, history, and scaling, but you cannot modify the Client tags.

Client Tags

Client tags, as the name implies, are only available for use in the Clients. This means that their values are isolated to a Client runtime. Even though they are created in the Designer, each Client creates their own instances. This makes them very useful as in-project variables, for passing information between screens and between other parts of the Clients, such as scripting. Client tags are a hybrid of memory, expression, and SQL query tags. However, they do not have a scan class.

Diagnostics Tags

Every device connection in Ignition comes with several tags that provide diagnostic information, such as, if the device is connected, last time values were read, or last time a write occurred.

In this section ...

System Tags

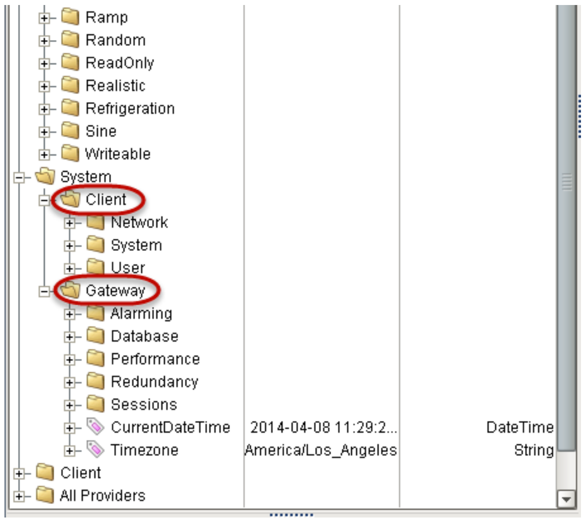
System tags provide status information about the system, such as memory usage, performance metrics, and so on. They exist for the Client and the Gateway. You can change the Gateway System tags to perform alarming, history, and scaling, but you cannot change the Client tags.

System Tags

Watch the Video

Client and Gateway System Tags

Ignition comes with a number of predefined System tags to provide more information about the Client and the Gateway. In the **Tag Browser**, under the **System** folder, you can see several **Client** and **Gateway** tags.



For **Client**, the System tags provide the following information:

- Client IP Address
- Client Hostname
- Client OS
- Who is logged in and what roles they have
- Which project is opened
- Gateway's address
- and more...

For **Gateway**, the System tags provide the following information:

- Database connection information
- Ignition performance stats
- Redundancy information
- Number of client sessions opened
- and more...

Next ...

- Client Tags

Client Tags

Client Tags, as the name implies, are only available for use in Clients. This means that their values are isolated to a Client runtime, and even though they are created in the Designer, each Client will create their own instances. This makes them very useful as in-project variables, for passing information between screens and between other parts of the Clients, such as scripting.

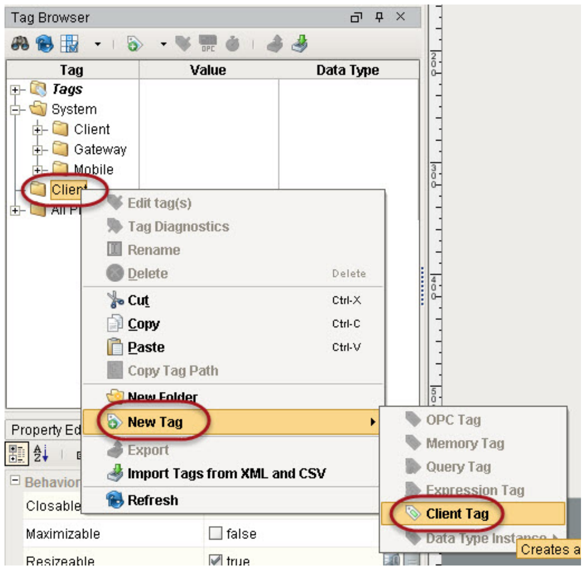
INDUCTIVE UNIVERSITY

Client Tags

Watch the Video

To create a Client tag

1. From the **Tag Browser**, right-click on the **Client** folder and select **New tag > Client Tag**.



The **Tag Editor** is displayed.

2. In the **Tag Editor**, for example, you can enter the following:

   Name: **Area**
   Value: **Processing**
   Data Type: **String**

3. Click **OK**.
   A new tag called **Area** is created in the **Client** folder.

Next ...

- Using Client Tags for Indirection

Using Client Tags for Indirection

Client tags can be used as a variable across all windows to indirectly point to a set of tags, such as an area of the plant. Suppose that a window has a drop-down that allows a user to select different areas of the plant. If the drop-down is bidirectionally bound to a Client tag's selected string value property, the user can change the drop-down's value, therefore resulting in the indirect binding throughout the project reflecting the change in the Client tag.

The Client tag can be bound to a Custom property on a window's Root Container. This Custom property can be bound to the Client tag. The components inside the window can have indirect binding on their properties that leverage the Custom property on the window. Therefore, the components on the window will be dependent on the Client tag. Client tags are managed inside the Client therefore each Client will be independent of other Clients.

Using Client Tags for Indirection

Watch the Video

Next ...

- Device Diagnostic Tags

Device Diagnostic Tags

INDUCTIVE UNIVERSITY

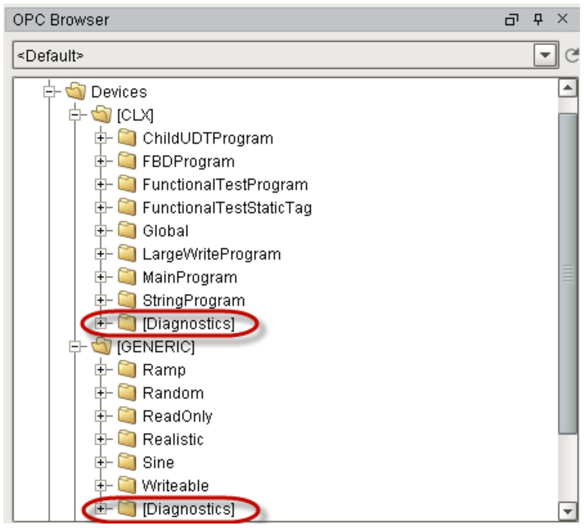
Device Diagnostic Tags

Watch the Video

Every device in Ignition's OPC-UA server exposes diagnostic tags that provide status and performance information about the device.

You add these tags into the tag database the same way you add tags from the actual PLC, that is through the **OPC Browser**. Under each device, you can see a special folder called **[Diagnostics]**. You can drag any of the [Diagnostics] tags into the **Tag Browser**.

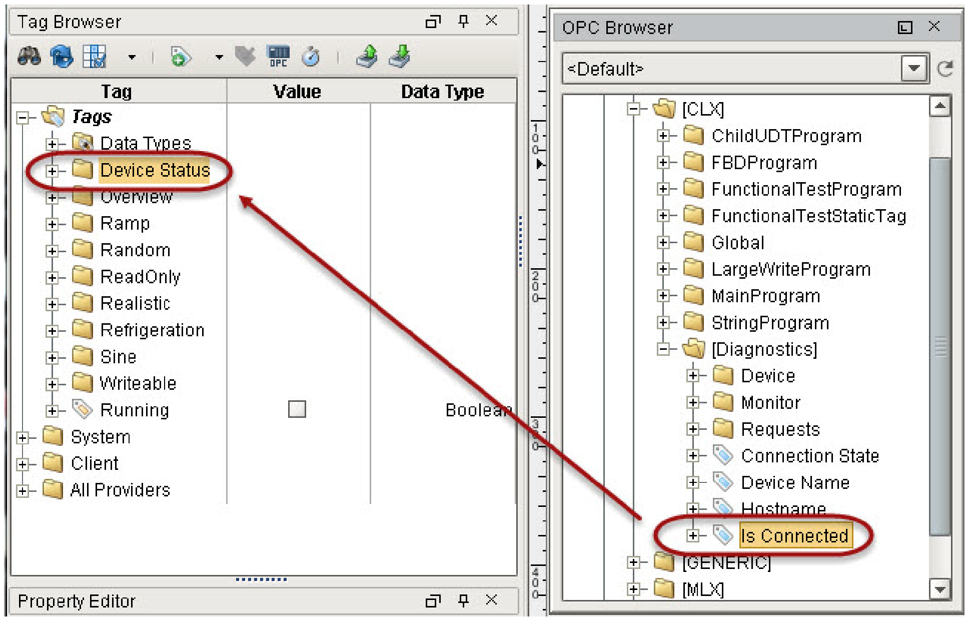
In the [Diagnostics] folder, the most commonly used tag is the **Is Connected** tag because it tells whether Ignition is connected to the device or not.



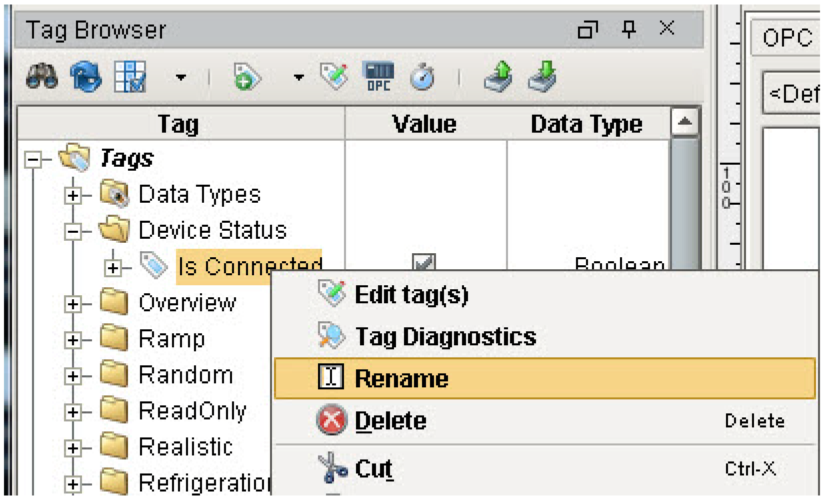
To use a diagnostic tag to show the PLC status

Let's use the **Is Connected** tag to show the status of each PLC.

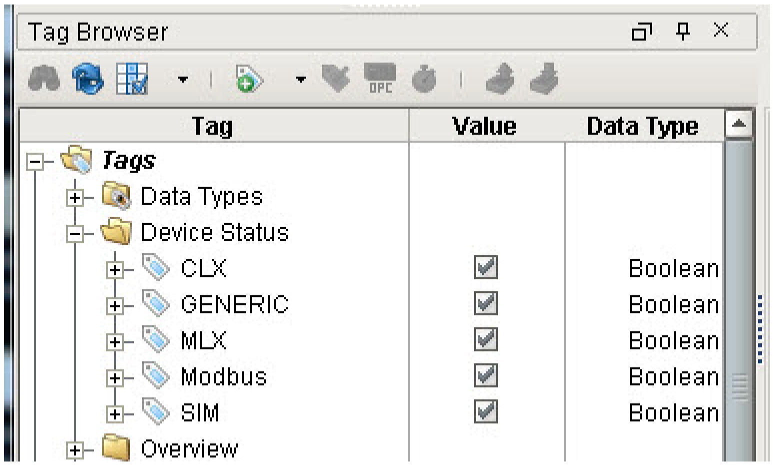
1. Create a folder in **Tag Browser** to contain the status bits by right-clicking on the **Tags** folder and selecting **New Folder**.
2. Name the folder **Device Status** and click **OK**.
3. Drag in the **Is Connected** tag from the **CLX** device into the **Device Status** folder.



4. Right-click on the **Is Connected** tag and rename it to **CLX**.



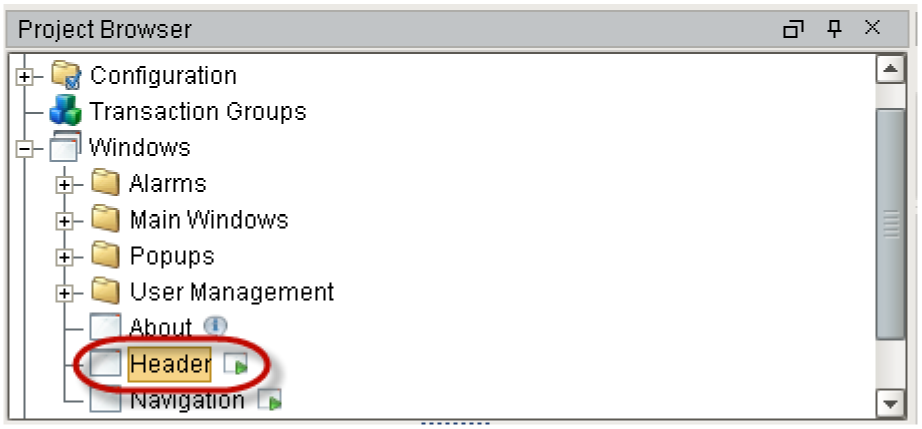
5. Repeat step 3 and 4 for all device connections you have in Ignition. If you have 5 devices, you will have 5 tags.



Tag	Value	Data Type
CLX	✓	Boolean
GENERIC	✓	Boolean
MLX	✓	Boolean
Modbus	✓	Boolean
SIM	✓	Boolean

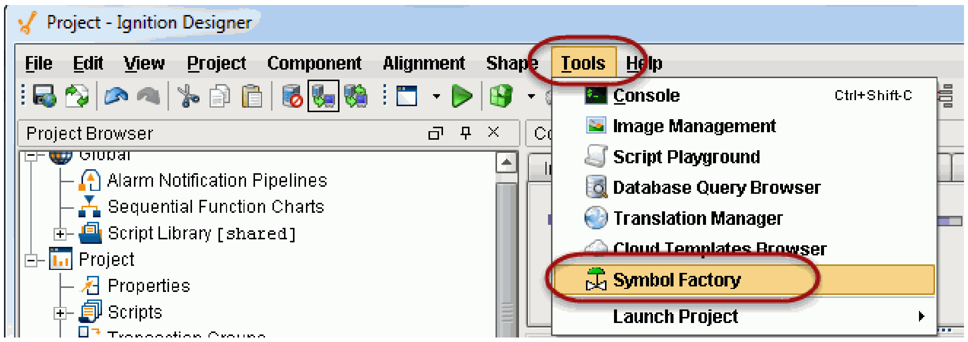
Now that we have a tag for each of the 5 PLCs, let's display the status of each PLC. For example in our **Header** window, we can make a PLC graphic and color it based on the state: **0 = red**, **1 = green**.

6. In the **Project Browser**, double-click the **Header** window to open it.



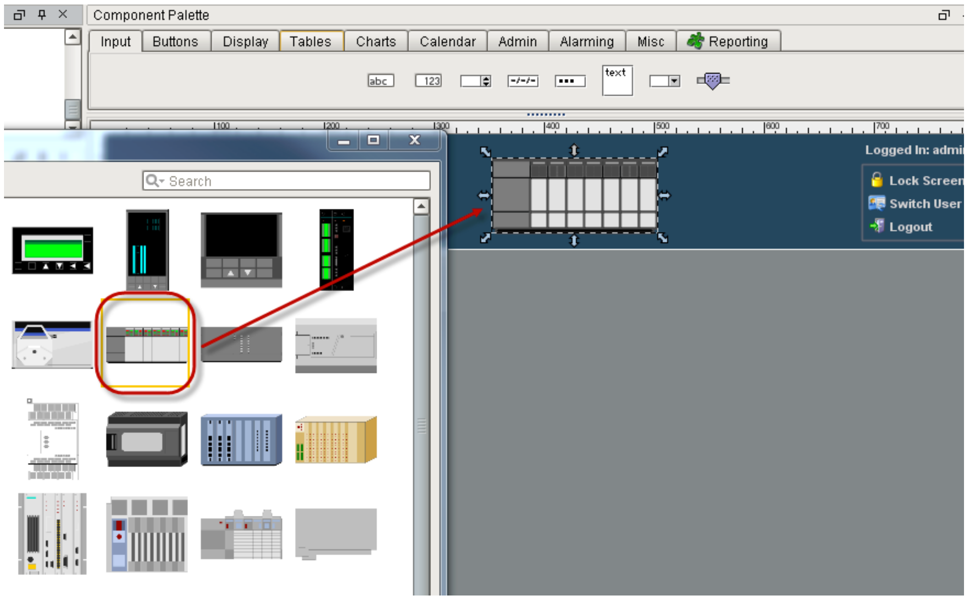
We will add a PLC graphic from Symbol Factory to the Header.

7. From the **Tools** menu, select **Symbol Factory**.



8. From the Symbol Factory window, select **Controllers** in the list.

9. Drag any device on the window to create the component (the PLC graphic).

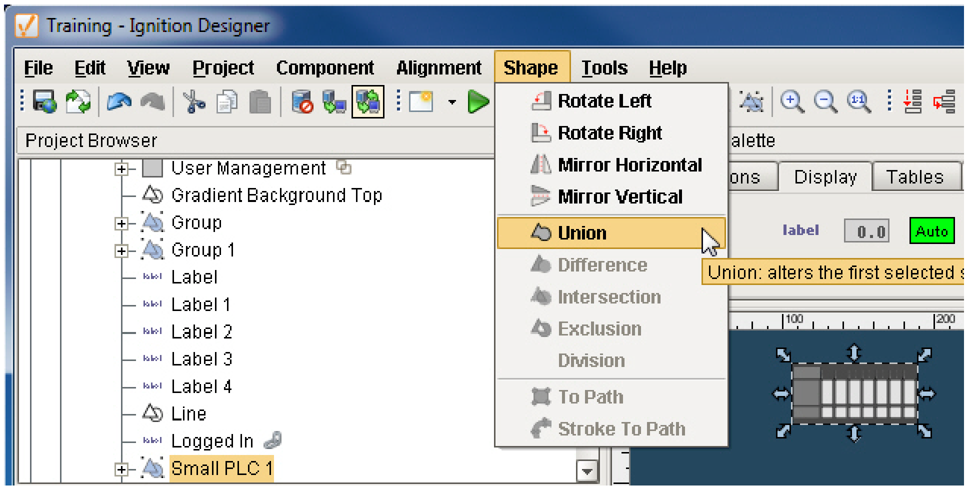


Symbol Factory components are made of several subcomponets or paths that make up a vector-based graphic. If you expand the component in the Project Browser, you can see all the paths. You can also double-click on the component in the design space to select the individual subcomponets. Fill color can be applied to each of these sub-components (or paths) dynamically through property binding. But there are two problems with applying color to the entire component. One is that you would have to change the fill color of each subcomponent dynamically through property binding which can take some time. Two, you will notice that you cannot bind the fill paint to a gradient color. To resolve these two problems we use a special method as described here to apply a tint to the entire component.

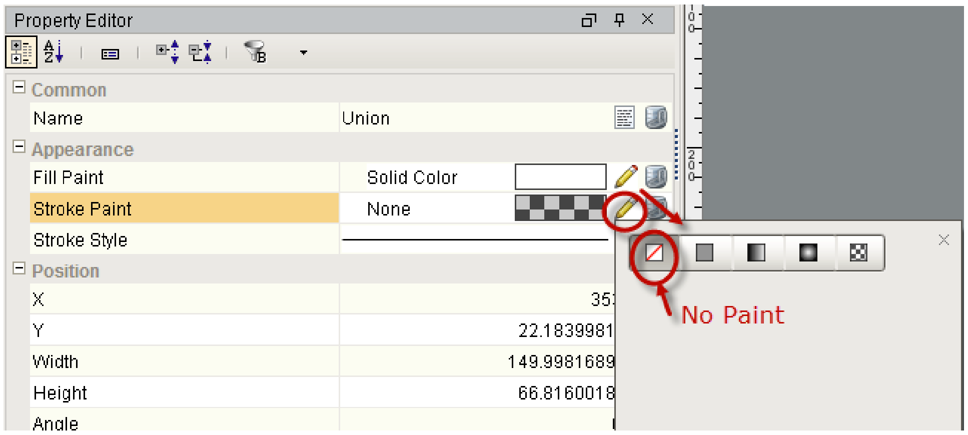
10. To apply tint to the entire component, duplicate the component by selecting it and then select **Edit > Duplicate** from menu or press Ctrl-D.

11. Select the duplicated component, which is above the original component.

12. Select **Shape > Union** from menu or click the **Union** button in the toolbar.
This flattens the duplicated shape into a single shape.



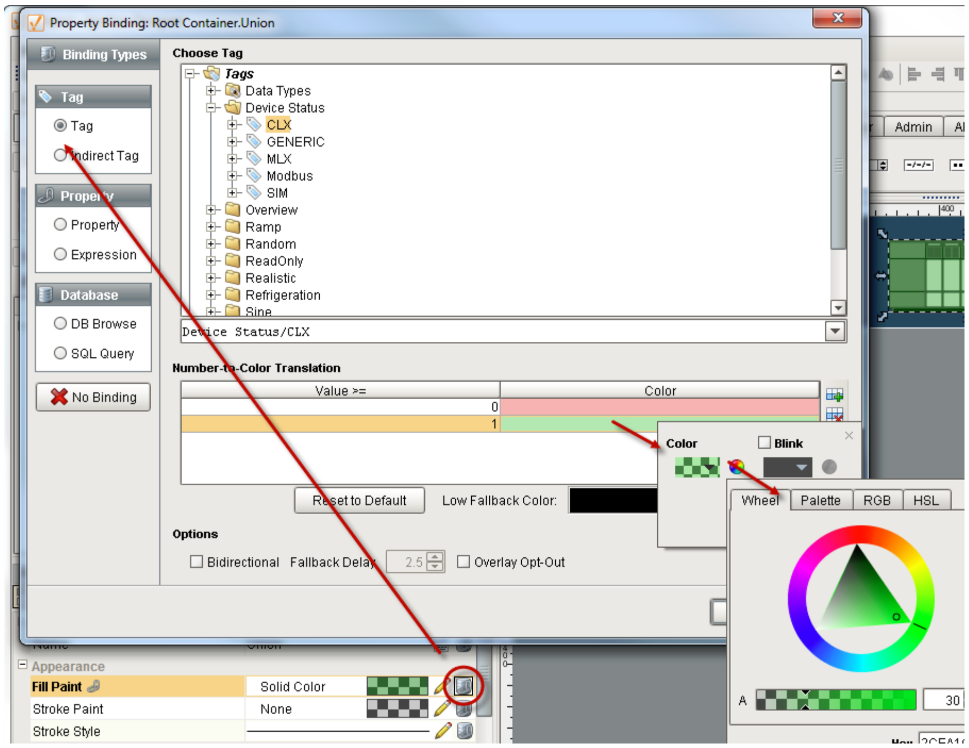
13. Remove the outline of the shape by clicking on the pencil icon of the **Stroke Paint** property and selecting **No Paint**.



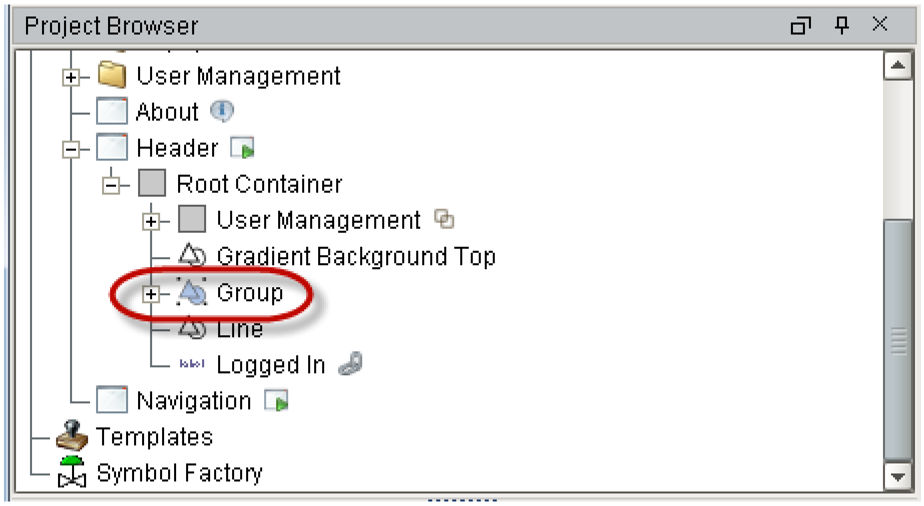
Property	Value
Common	
Name	Union
Appearance	
Fill Paint	Solid Color
Stroke Paint	None
Stroke Style	
Position	
X	35
Y	22.1839981
Width	149.9981689
Height	66.8160018
Angle	

14. Bind the **Fill Paint** property to the **CLX** status tag. Notice when binding a color property to a tag, you get to choose a color for each state of the tag in **Number-to-Color Translation**. You can also make the colors to blink between two different colors.

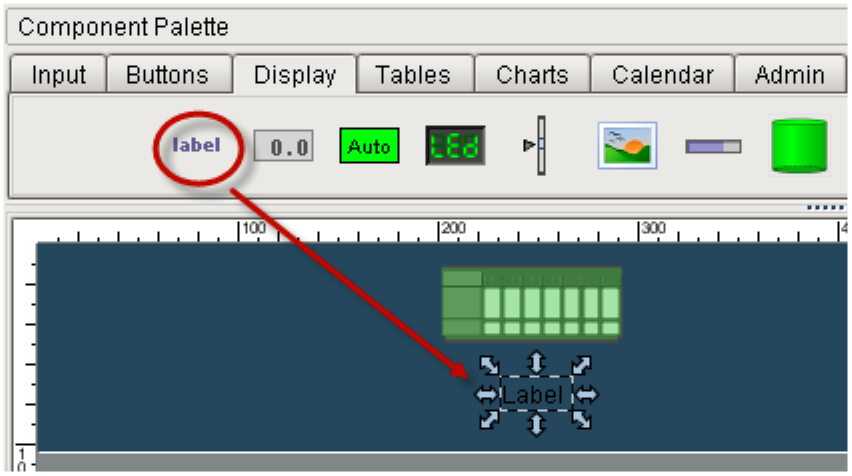
15. In **Number-to-Color Translation**, set value **0 = red**, value **1 = green**, and alpha channel of each color to **30** (make sure each color is semitransparent so you can see the actual symbol behind it).



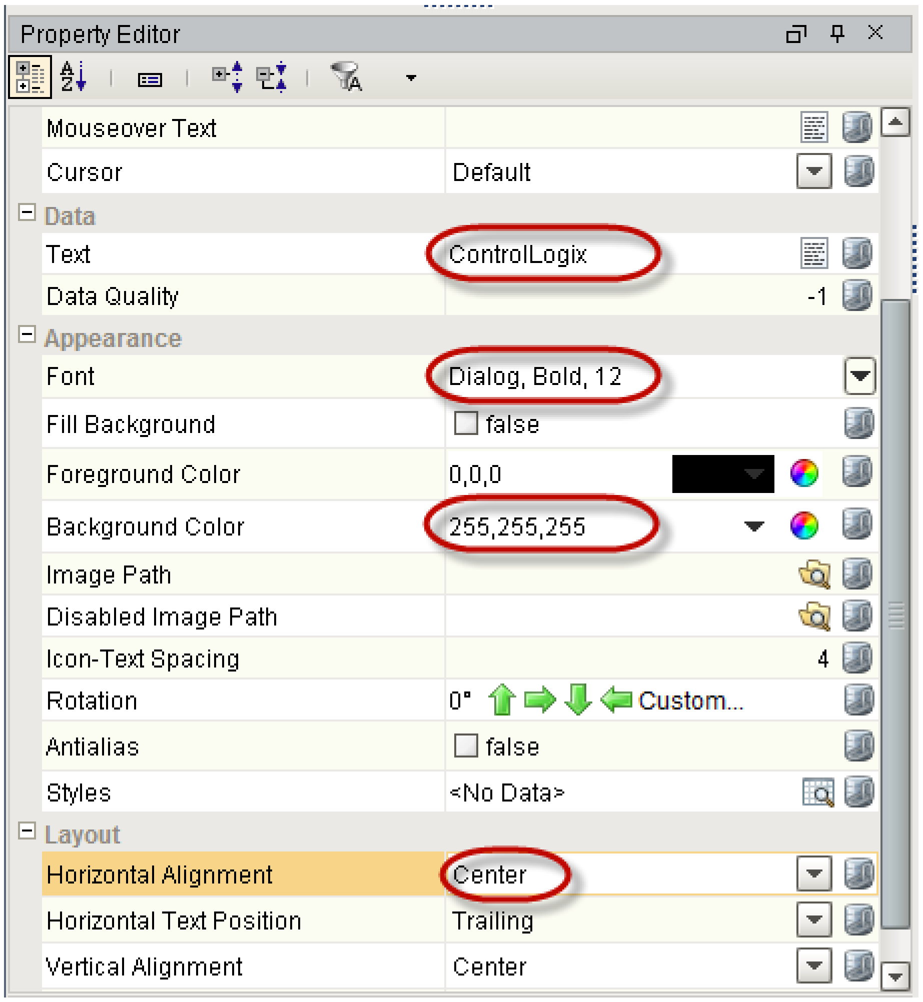
16. Group the two components together into one by selecting both components from the Project Browser and clicking the **Component > Group** or Combine icon in the toolbar. Now they are one component.



17. To name the device, drag in the **label** component from the **Display** tab into the window under the graphic.

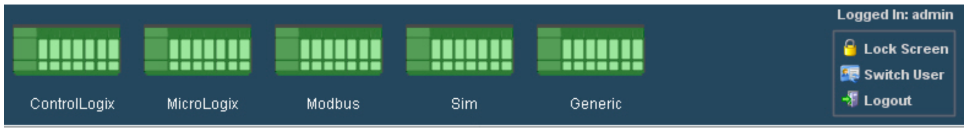


18. Set the label properties as follows:
Text: **ControlLogix**
Font: **Dialog, Bold, 12**
Foreground Color: **255,255,255**
Horizontal Alignment: **Center**



Mouseover Text	
Cursor	Default
Data	
Text	ControlLogix
Data Quality	-1
Appearance	
Font	Dialog, Bold, 12
Fill Background	false
Foreground Color	0,0,0
Background Color	255,255,255
Image Path	
Disabled Image Path	
Icon-Text Spacing	4
Rotation	0° Custom...
Antialias	false
Styles	<No Data>
Layout	
Horizontal Alignment	Center
Horizontal Text Position	Trailing
Vertical Alignment	Center

19. Repeat for each PLC.



You can now see the status of each PLC based on the colors they display (0=red, 1=green).

Next ...

- add link here

Tag Historian

Tag Historian™ makes it easier that ever to store and use historical data. Simply by selecting 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. The available options for partitioning and deleting old data help to ensure the system stays properly maintained with minimal extra work. Also, you can drag-and-drop tags directly onto an Easy Chart to create trends or onto a table to display historical values. Tags Historian's robust querying features provide you great flexibility in how you retrieve the stored data.

You must first install the **Tag Historian module** so that the standard Tag Historian features are added to Ignition and the Historian Providers show up in the Historian Configuration section of the Gateway.

The various aspects of how Tag Historian stores and queries data are described next.

Historian Providers

The settings for the Tag Historian providers are in the Gateway under **Tags > History**. Historian providers are automatically created and removed according to the configured database connections. By default, they are created with a one month partition size, and will not delete old data.

Tag Configuration and Historical Value Generation

The first step to storing historical data is to configure tags to report values. This is done from the History page in the **Tag Editor** in the Designer. The properties include a historical scan class, that will be used to check for new values. Once values surpass the specified deadband, they are reported to the history system, which then places them in the proper store and forward engine.

Data storage

As mentioned, the historical tag values pass through the store and forward engine before ultimately being stored in the database connection associated with the historian provider. The data is stored according to its datatype directly to a table in the SQL database, with its quality and a millisecond resolution timestamp. The data is only stored on-change, according to the value mode and deadband settings on each tag, thereby avoiding duplicate and unnecessary data storage. The storage of scan class execution statistics ensures the integrity of the data. While advanced users may change the table according to their database to be more efficient (for example, using a compressed engine), Ignition does not perform binary compression or encrypt the data in any way.

Table Partitioning

Ignition has the ability to automatically break up data into different tables of fixed duration. This can help make data maintenance easier, by preventing tables from becoming too large. Tables can easily be deleted in order to prune old data, and the database is able to better optimize access to frequently retrieved rows. The built-in partitioning feature can be used with any database.

It is important to note the difference between this feature and any partitioning options that the database might provide. Most modern databases offer their own faculties for defining "partitions", offering similar and greater benefits. While Ignition cannot use these features directly, advanced users may choose to apply these features on top of what Ignition currently offers.

Data Compression

As mentioned above, Ignition does not perform any binary compression on the data. That is, values are stored directly in standard database tables. However, to reduce the number of values stored, Ignition offers two different algorithms for pre-compressing the data (trimming unnecessary values). The two modes correspond to the value mode property of the tag: Discrete, and Analog.

- **Discrete**
  The value uses a simple deadband, and is only stored when a new value is +/- the deadband value away from the previously stored value.

- **Analog**
  The deadband is used to form a corridor along the trajectory of the value. A new value is only stored when it falls outside the previous corridor. When this occurs, the trajectory is recalculated, and a new corridor formed.

See Value Mode, in History Properties for more information about the difference between Discrete and Analog values.

Querying

While the data is stored openly in the database, the format does not lend itself well to direct querying. Instead, Ignition offers a range of querying options that are very powerful and flexible. In addition to the simple on-change querying, the system can perform advanced functions such as querying many tags from multiple providers, calculating their quality, interpolating their values, and coordinating their timestamps to provide fixed resolution returns.

Querying can be performed on tables and charts through the Historical Binding, and through scripting.

In this section ...

Configuring Historian

You can configure the **Database Tag History Provider** in the **Configure** section of the Gateway by going to **Tags > History**. A History Provider is created automatically for each database connection, and will be removed if the connection is removed. Although enabled by default, the providers won't interact with the database unless data is logged to them.

Main	
Data Connection	Name of the Data Connection for the Tag History Provider.
Enabled	If the check box is selected (enabled), the provider is turned on and accepts tag history data. If disabled, the database is not shown in the list of history providers when configuring tag history from the Designer. Also, any data logged to the provider, will error out and be quarantined by the store and forward engine, if possible.
Data Partitioning	
Enable Partitioning	To improve query performance, Tag Historian can partition the data based on time. Partitions will only be queried if the query time range includes their data, thereby avoiding partitions that aren't applicable and reducing database processing. On the other hand, the system must execute a query per partition. It is therefore best to avoid both very large partitions, and partitions that are too small and fragment the data too much. When choosing a partition size, it is also useful to examine the most common time span of queries.
Partition Length and Units	The size of each partition, the default is one month. Many systems whose primary goal is to show only recent data might use smaller values, such as a week, or even a day.
Data Pruning	
Enable Data Pruning	Partitions with data older than a specific age are deleted. The check box is not selected/enabled by default. **Important:** Data pruning works by deleting old partitions. Therefore, data will only be removed when a partition has no data younger than the prune age.
Prune Age and Units	The maximum age of data. As mentioned, the data is deleted by the partition, and could therefore surpass this threshold by quite a bit before all of the data in the partition is old enough to be dropped.

In this section ...

Set up Tags to Log

Logging data is easy with Tag Historian. Once you have a database connection, all you do is set the tags to store history and Ignition takes care of the work. Ignition creates the tables, logs the data, and maintains the database.

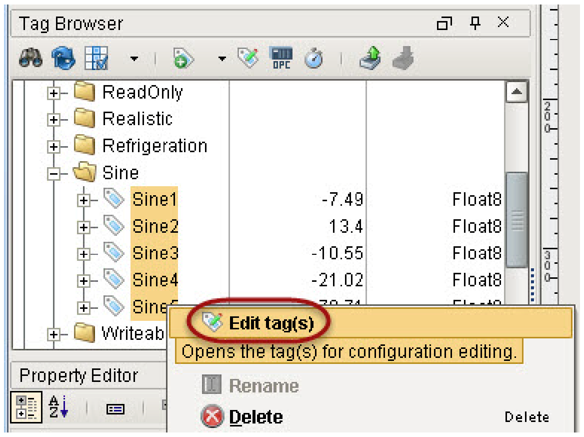
As mentioned previously, the historical tag values pass through the store-and-forward engine before ultimately being stored in the database connection associated with the historian provider. The data is stored according to its datatype directly to a table in the SQL database, along with its quality and a millisecond resolution time stamp. The data is only stored on-change, according to the value mode and deadband settings on each tag, thereby avoiding duplicate and unnecessary data storage.The storage of scan class execution statistics ensures the integrity of the data.

To log tag history data

Do the following steps to log history data for your tags:

1. Go to the Designer, from **Tag Browser**, select one or more tags.
   For example, select all the **Sine** tags in the Sine folder.

2. Right-click on the selected tags, and then select **Edit tag(s)**.
   The Tag Editor window is displayed. Here, you can edit the tag and change the name, data type, scaling options, metadata, permissions, history, and alarming.



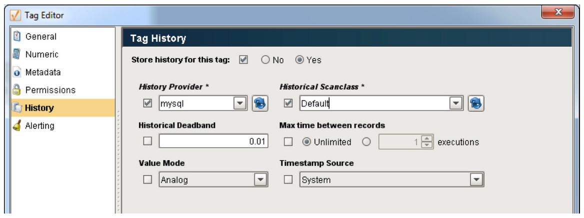
Sine1	-7.49	Float8
Sine2	13.4	Float8
Sine3	-10.55	Float8
Sine4	-21.02	Float8

3. Click on **History** on the left side of **Tag Editor**.

4. Click **Yes** for **Store history for this tag**.

5. Choose a database from the **History Provider** dropdown.
   For example, choose **mysql** as the database you want to log to.

6. Choose **Default** for **Historical Scanclass**.
   This sets the rate at which data is logged. The default is set to 1 second which means data is logged every second.



Tag History

Store history for this tag: ☑ No [✓ Yes]

History Provider * ☑ mysql

Historical Scanclass * ☑ Default

Historical Deadband ☐ 0.01

Max time between records ☐ [✓ Unlimited] 1 executions

Value Mode ☐ Analog

Timestamp Source ☐ System

7. Click **OK**.
   Now look in the **Tag Browser**, to the right of each **Sine** tag that is storing history, an icon appears letting you know it is setup.

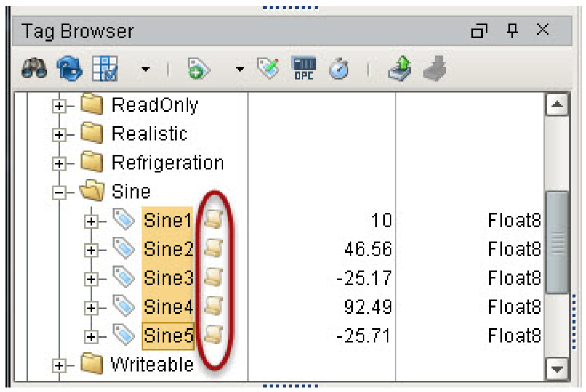
INDUCTIVE UNIVERSITY

Set up Tags to Log

Watch the Video

To Continue on the step-by-step simple workflow...

After you **History to the Tags**, as described on this page, click the link below for Step 8 to continue.



Sine1	10	Float8
Sine2	46.56	Float8
Sine3	-25.17	Float8
Sine4	92.49	Float8
Sine5	-25.71	Float8

If you were to look in your database, you can see all the tables and data Ignition has created for you.

Next...

- Add History to Tags in UDT

Add History to Tags in UDT

.

You can set the UDT to log history data, then all the instances of that UDT will log data without having to edit the individual instances.

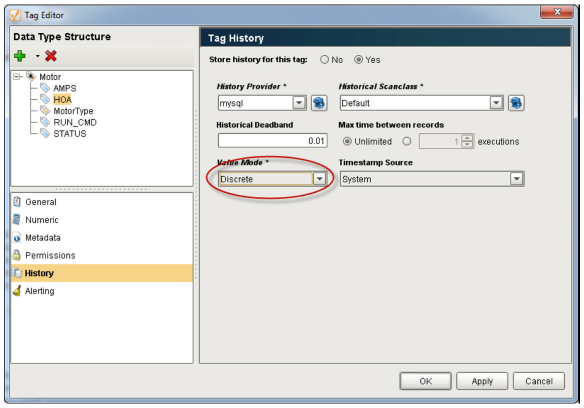
INDUCTIVE UNIVERSITY

Add History to Tags in UDT

Watch the Video

To set the UDT to log history data

1. **Right-click on the UDT (for example, a Motor UDT) and select Edit tag(s).**
   The Tag Editor is displayed.
2. **In Tag Editor, click a tag (for example, the AMPS tag), select History, and set Store history for this tag to Yes.**
3. **Set the following in Tag Editor and click Apply:**
   History Provider: mysql
   Historical Scanclass: Default
   Value Mode: Analog
4. **Do the same for the HOA tag but set Value Mode to Discrete because HOA is a discrete state.**



Tag Editor

Data Type Structure

- Motor
  - AMPS
  - HOA
  - MotorType
  - RUN_CMD
  - STATUS

General
Numeric
Metadata
Permissions
History
Alerting

Tag History

Store history for this tag: No [✓ Yes]

History Provider *: mysql

Historical Scanclass *: Default

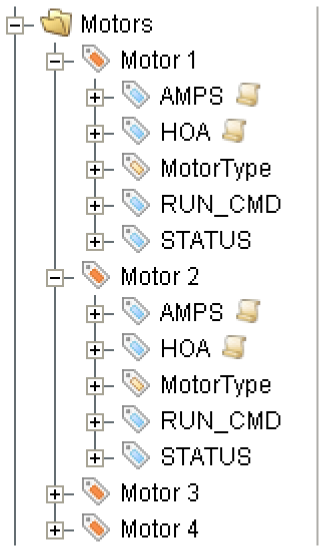
Historical Deadband: 0.01

Max time between records: [✓ Unlimited] 1 executions

Value Mode *: Discrete

Timestamp Source: System

5. Click **OK** to save.
   Now every motor instance automatically starts logging data.



- Motors
  - Motor 1
    - AMPS
    - HOA
    - MotorType
    - RUN_CMD
    - STATUS
  - Motor 2
    - AMPS
    - HOA
    - MotorType
    - RUN_CMD
    - STATUS
  - Motor 3
  - Motor 4

Next...

- Discrete vs. Analog Mode

Discrete vs. Analog Mode

Ignition offers two different algorithms for pre-compressing the data (trimming unnecessary values). The two algorithms corresponding to the two value mode properties of the tag are: Discrete and Analog.

The value mode (Discrete or Analog) dictates the type of value that the tag represents, affects how the deadband is applied to values, and how interpolation is performed when querying.

- **Discrete**
  The value uses a simple deadband and is only stored when a new value is + /- the deadband value away from the previously stored value.
- **Analog**
  The deadband is used to form a corridor along the trajectory of the value. A new value is only stored when it falls outside the previous corridor. When this occurs, the trajectory is recalculated and a new corridor is formed.

Typically, **Discrete** is used for boolean or integers that represent state, and **Analog** is used for floats or integers that change more often (which is why you want to perform compression).

You choose and set the **Value Mode** on the **History** section of the **Tag Editor** window.

While advanced users can change the table according to their database to be more efficient (for example, using a compressed engine), Ignition does not perform binary compression or encrypt the data in any way.

INDUCTIVE UNIVERSITY

Discrete vs. Analog Mode

Watch the Video

Next...

- Store and Forward

Store and Forward

The Store and Forward system provides a reliable way for Ignition to store historical data to the database. Systems such as the Tag Historian and Transaction Groups use this system to prevent data loss and store data efficiently using a record cache.

For more information on Store and Forward, see the following sections:

- About Store and Forward
- Using Store and Forward
- Configuring Store and Forward
- Controlling Quarantine Data

INDUCTIVE UNIVERSITY

Store and Forward

Watch the Video

Next...

- Data Partitioning and Pruning

Data Partitioning and Pruning

Tag Historian will partition data into separate tables according to the time setting so that one table doesn't grow indefinitely, and then will delete old data to ensure the system is maintained for query performance. The data prune feature will delete partitions with data older than a specific age/time.

To partition and prune data

1. Go to the **Configure** section of the Gateway.
2. Select **Tag > History** from the menu on the left.
   The **Database Tag History Providers** page is displayed. You can see the Databases that have Enabled tag history on and their Status shows as Running.
3. Click on **edit** at the far right of the database you want make changes to.

Main	
Data Connection	Name of the Data Connection for the Tag History Provider.
Enabled	By default, the check box is selected (enabled) meaning the provider is turned on and accepts tag history data.
Data Partitioning	
Enable Partitioning	By default, partitioning is enabled to improve query performance. Tag Historian partitions and breaks up the data into separate tables based on time. Partitions will only be queried if the query time range includes their data, thereby avoiding partitions that aren't applicable and reducing database processing. On the other hand, the system must execute a query per partition. It is therefore best to avoid both very large partitions, and partitions that are too small and fragment the data too much. When choosing a partition size, it is also useful to examine the most common time span of queries.
Partition Length and Units	The size of each partition, the default is one table per month. Many systems whose primary goal is to show only recent data might use smaller values, such as a week, or even a day.
Enable Pre-processed Partitions	This option is not selected by default. Pre-processed partitions will use more space in the database, but can improve query speed by summarizing data, reducing the amount that must be loaded.
Pre-processed Window Size (seconds)	When pre-processing is turned on, the data will be summarized into blocks of this size
Data Pruning	

INDUCTIVE UNIVERSITY

Data Partitioning and Pruning

Watch the Video

Enable Data Pruning	The check box is not selected/enabled by default. Partitions with data older than a specific age are deleted and if the data is not archived, the data is then lost. **Important:** Data pruning works by deleting old partitions. Therefore, data will only be removed when a partition has no data younger than the prune age.
Prune Age and Units	The maximum age of data. As mentioned, the data is deleted by the partition, and could therefore surpass this threshold by quite a bit before all of the data in the partition is old enough to be dropped.

History Table Timestamps

If you've looked behind the scenes of SQLTags Historian, you've probably noticed the timestamps are not stored as standard SQL timestamps. They are stored in a variant of Unix Time, or the number of milliseconds since January 1, 1970 00:00:00. The time may come when you need to convert that timestamp to a more human-readable format. This article will quickly cover how to do it in MySQL and MSSQL.

MySQL
It's pretty easy to deal with unix timestamp in MySQL because they have a built-in function for doing so. The FROM_UNIXTIME() function will take in a unix timestamp and spit out the current timestamp.
Usage:

```
SELECT FROM_UNIXTIME(t_stamp/1000) FROM sqlt_data_1_2016_08
```

MSSQL
In Microsoft SQL Server, it's a little more verbose. We use the DATEADD() function to figure out the timestamp.
Usage:

```
SELECT DATEADD(s,t_stamp/1000,'1970-01-01 00:00:00') FROM sqlt_data_1_2016_08
```

Next...

- Tag History Splitter

Tag History Splitter

You can have the tag history data be logged in two different database providers by downloading and installing the Tag History Splitter module.

Tag History Splitter

Watch the Video

To use the tag history splitter

Once the module is installed, to create the additional tag history provider, do the following steps:

1. Go to the **Configure** section of the Gateway, and choose **Tags>History** from the menu on the left.
2. Go to the bottom of the **Database Tag History Providers** page, and click **Create new Historical Tag Provider**.
   The Add Historical Tag Provider Step 1: Choose Type page is displayed.
3. Select **Tag History Splitter** and click **Next**.
   The New Historical Tag Provider page is displayed.
4. Enter a name for the **Provider Name**, and from the dropdown choose a database for the **First Connection** (for the primary data) and one for the **Second Connection** (for secondary data).
5. Click **Create New Historical Tag Provider**.
   Now the splitter tag provider is created and you can use it to log the tag history data in the Designer.
   Check this by going back to the Designer, selecting a tag and then right-clicking to select the Edit tags. On the Tag Editor window go to the History tab and see the new splitter provider in the dropdown of the History Provider.

Next...

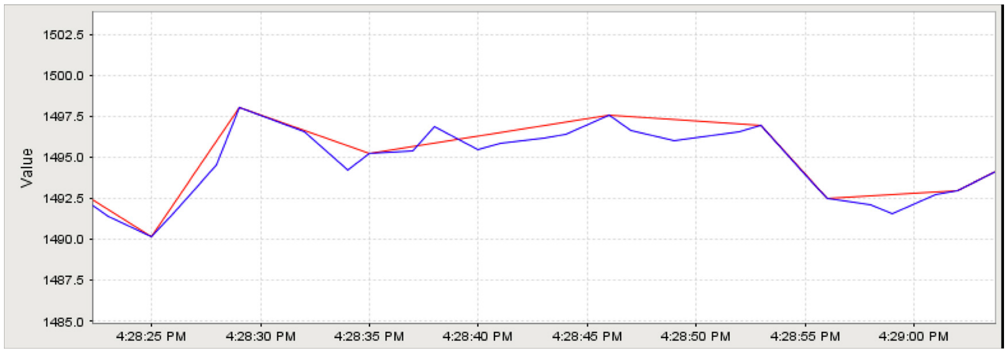
Historian Charts

Use the EasyChart™ component in the Designer to display historical data or realtime data depending on the range of your time selection. Simply drag-and-drop history-enabled tags onto a chart to create chart pens and data trends.

Let's look at trending tags and displaying the information in a graph. In both of the following examples, data was logged first.

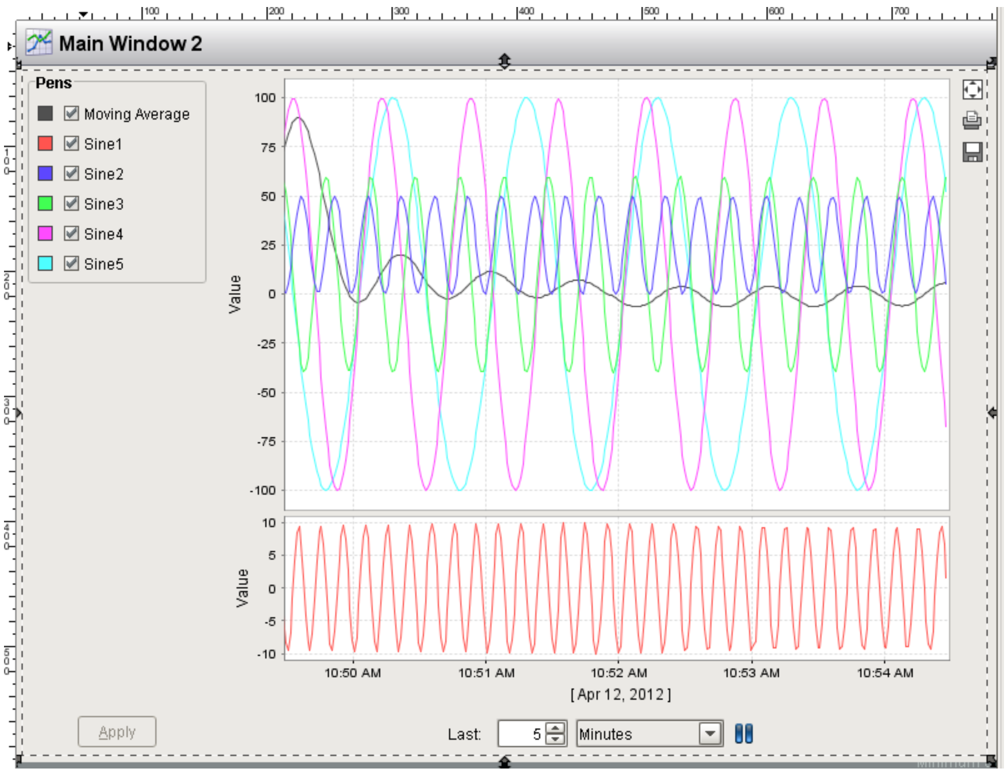
Example 1

Here is an EasyChart example of raw (blue) vs. compressed (red) data using the analog mode:



Example 2

Here is another example:



In this section ...

Basic Easy Chart

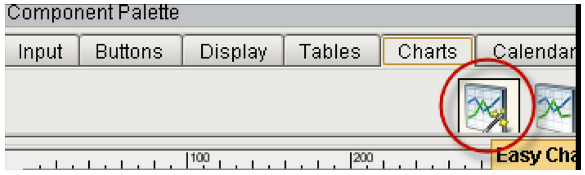
The **Easy Chart** component allows you to make powerful and runtime-configurable timeseries charts /graphs from the Tag Historian data.

Creating a Tag History Graph

You can see the tag history data, by creating a graph. Your graph can include subplots, axes, and a moving average pen. You can view the runtime data in realtime or historical mode.

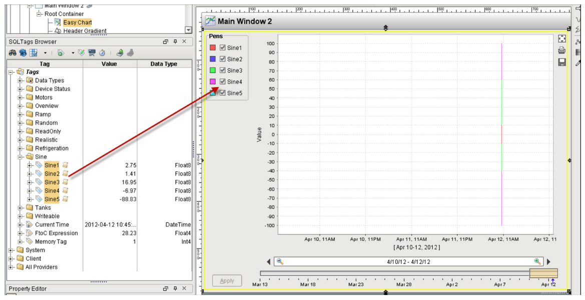
To create a tag history graph

1. Open Main Window 2, from the Charts tab of the Component Palette, drag in the Easy Chart component.



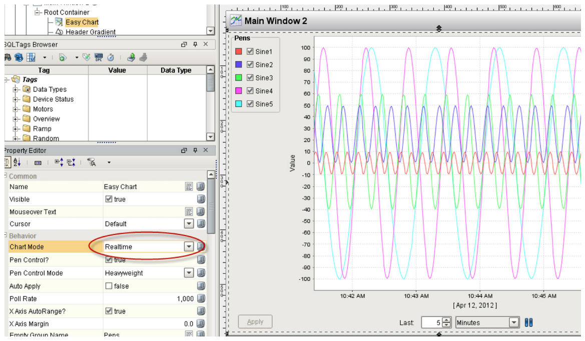
We can add the tags to the graph the same way we created the status and control windows, that is, through drag and drop.

2. Drag all the Sine tags from Tags Browser to the graph.



The tags are displayed on Main Window 2 but because we have just started logging data, not much data shows up.

3. To show more data, set the Chart Mode property of Easy Chart to Realtime.



Property Editor

Common	
Name	Easy Chart
Visible	true
Mouseover Text	
Cursor	Default
Behavior	
Chart Mode	Realtime
Pen Control?	true
Pen Control Mode	Heavyweight
Auto Apply	false
Poll Rate	1,000
X Axis AutoRange?	true
X Axis Margin	0.0
Empty Group Name	Pens

Next...

- Easy Chart - Axes

INDUCTIVE UNIVERSITY

Basic Easy Chart

Watch the Video

The step-by-step simple workflow...

After you **Create a Graph and View Graphs in Clients**, you have completed the "simple workflow", so Congratulations!

Step 1. **Launch the Gateway**

Step 2. **Connect to a Device**

Step 3. **Connect to a Database**

Step 4. **Launch the Designer**

Step 5. **Create Tags**

Step 6. **Launch Clients**

Step 7. **Add History to Tags**

Step 8. **Create a Graph and View Graphs in Clients**

Easy Chart - Axes

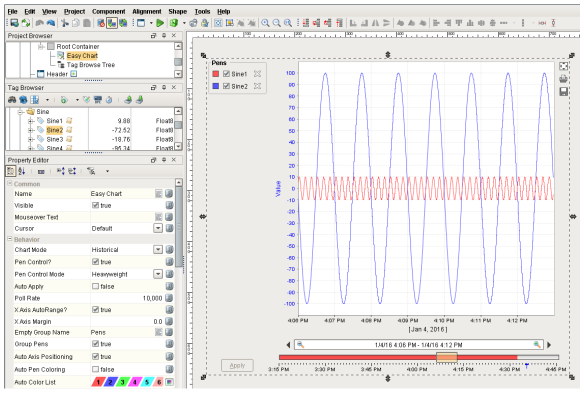
The Easy Chart supports the use of multiple axes for displaying data from the Tag Historian. Drag your tags over from the Tag Browser to your Easy Chart that you want to display. Make sure your tags are logged in the Tag Historian. Now, let's look at how we can configure multiple axes on an Easy Chart component.

INDUCTIVE UNIVERSITY

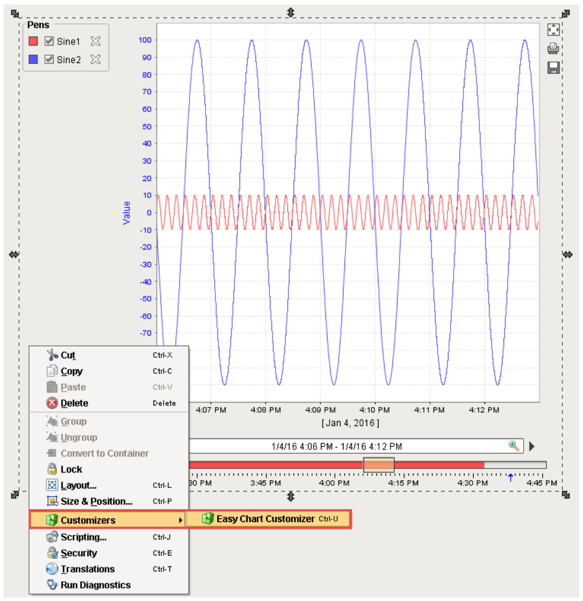
Easy Chart - Axes

Watch the Video

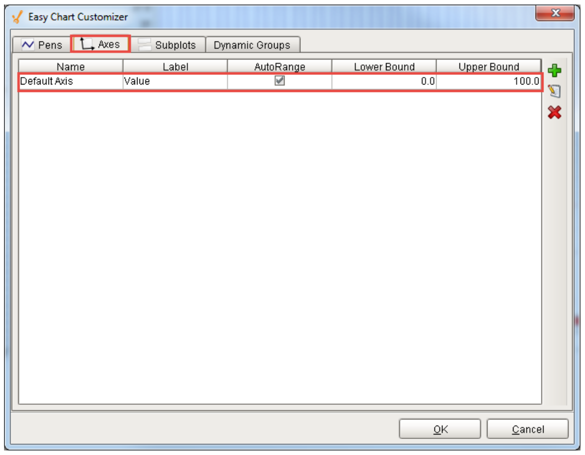
The following chart shows 2 tags: Sine 1 and Sine 2. They both have completely different value ranges. Sine 1's values range between -10 and 10. Sine 2 values range between -100 and 100. Since both Sines are on the same axis, it is hard to see the details of Sine 1's values since Sine 2 is throwing off the axis because of its wide range of values.



You have the option of putting your tags into different axes. You can do that in the Customizer of the Easy Chart component. **Right click** on the **Easy Chart component** and scroll down to **Customizers > Easy Chart Customizer.**



Click on the Axes tab so you can add on one or more axes to your component. There is already one axis showing called Default Axis. Both of the Sine tags are sharing the Default Axis.



Name	Label	AutoRange	Lower Bound	Upper Bound
Default Axis	Value	✓	0.0	100.0

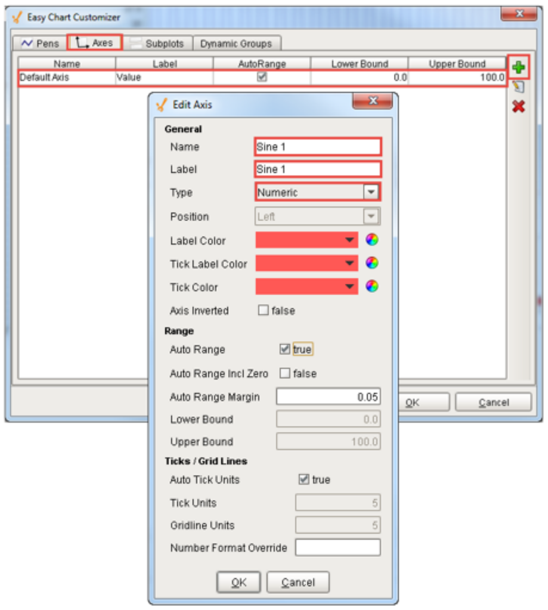
On the top right of the Easy Chart Customizer screen, **click the green arrow** to add a new axis. Call this Sine 1. The Label is the label name that will be displayed on your chart. Call this Sine 1 as well.

The axis can be a Numeric, Logarithmic or Symbol. Make Sine 1 a **'Numeric.'** You can specify the colors on the **Label**, **Tick Label** and **Tick Color**. In this example, make them all red.

You will notice that the **Position** dropdown is disabled. There is an **Auto Axis Positioning** property in the Property Editor of the Easy Chart component that is true by default. If you turn off **Auto Axis Positioning**, you can then specify the left/right position of each of the axes.

You can also specify the axis Range to be **Auto Range** on or not. If you don't set Auto Range, you can enter lower and upper bound values. If you set Auto Range to **'True'**, you can enter a margin value which is a plus or minus value for the top and bottom of the chart so you get a little padding. The margin value will enable you to see the entire trend so the values don't go all the way to the very top or bottom of the chart. **Set the Auto Range** to **'True'** and the **Auto Range Margin** to **'0.05.'**

You can set some tick units if you want and then **Press OK.**



Name	Label	AutoRange	Lower Bound	Upper Bound
Default Axis	Value	✓	0.0	100.0

General

Name	Sine 1
Label	Sine 1
Type	Numeric
Position	Left
Label Color	🟥
Tick Label Color	🟥
Tick Color	🟥
Axis Inverted	false

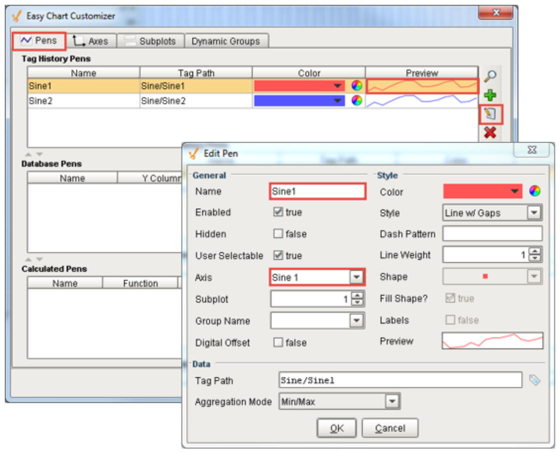
Range

Auto Range	✓ true
Auto Range Incl Zero	false
Auto Range Margin	0.05
Lower Bound	0.0
Upper Bound	100.0

Ticks / Grid Lines

Auto Tick Units	✓ true
Tick Units	5
Gridline Units	5
Number Format Override	

Now you have 2 axes, one for Sine 1 and another for Sine 2. **Click on the Pens tab for Sine 1**, **select the Tag row** then **click the Edit Pen icon** to change the Axis. Select the **'Sine 1'** Axis from the dropdown menu and **click OK**.



Tag History Pens

Name	Tag Path	Color	Preview
Sine1	Sine/Sine1	🟥	
Sine2	Sine/Sine2	🟦	

Database Pens

Name	Y Column

Calculated Pens

Name	Function

General

Name	Sine1
Enabled	true
Hidden	false
User Selectable	true
Axis	Sine 1
Subplot	1
Group Name	
Digital Offset	false

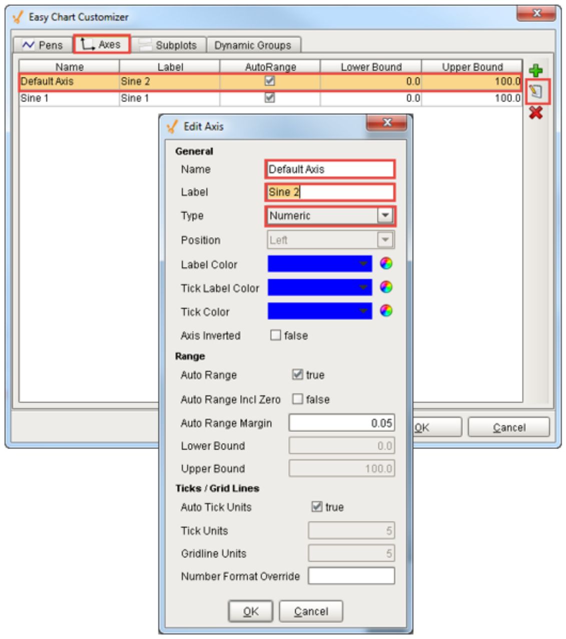
Style

Color	🟥
Style	Line w/ Gaps
Dash Pattern	
Line Weight	1
Shape	
Fill Shape?	true
Labels	false
Preview	

Data

Tag Path	Sine/Sine1
Aggregation Mode	Min/Max

Now repeat the steps for Sine 2. **Click on the Axes tab.** You can leave the **Name** as **'Default Axis'**, and enter a **Label** and call it **'Sine 2.'** Keep the **Type** as **'Numeric.'** You can specify the colors on the **Label**, **Tick Label** and **Tick Color**. In this example, make them all blue and click **OK**.



Name	Label	AutoRange	Lower Bound	Upper Bound
Default Axis	Sine 2	✓	0.0	100.0
Sine 1	Sine 1	✓	0.0	100.0

General

Name	Default Axis
Label	Sine 2
Type	Numeric
Position	Left
Label Color	🟦
Tick Label Color	🟦
Tick Color	🟦
Axis Inverted	false

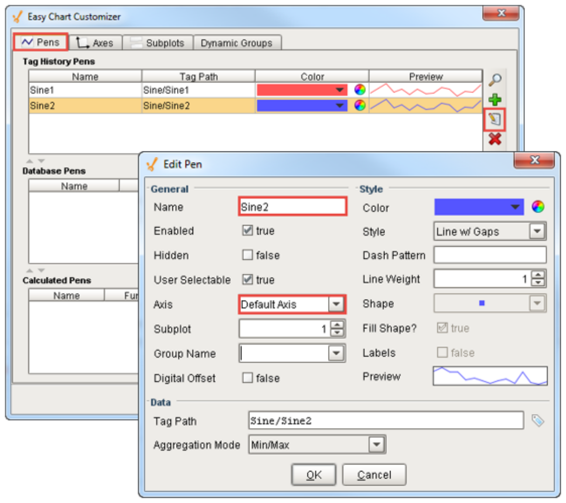
Range

Auto Range	true
Auto Range Incl Zero	false
Auto Range Margin	0.05
Lower Bound	0.0
Upper Bound	100.0

Ticks / Grid Lines

Auto Tick Units	true
Tick Units	5
Gridline Units	5
Number Format Override	

For the Sine 2 pen, we don't need to change the Axis because we left the default name and only changed the Label to **'Sine 2.'** To check, **click on the Pens tab for Sine 2**, **select the Tag row** then **clic k the Edit Pen icon** to change the Axis, otherwise, the **'Default Axis'** will remain the default. **Click OK** and now both axes are different.



Tag History Pens

Name	Tag Path	Color	Preview
Sine1	Sine/Sine1	🟥	
Sine2	Sine/Sine2	🟦	

Database Pens

Calculated Pens

Edit Pen

General

Name	Sine2
Enabled	☑ true
Hidden	☐ false
User Selectable	☑ true
Axis	Default Axis
Subplot	1
Group Name	
Digital Offset	☐ false

Style

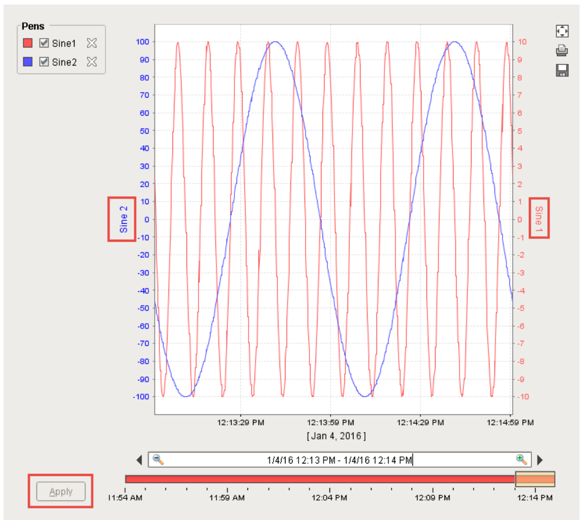
Color	🟦
Style	Line w/ Gaps
Dash Pattern	
Line Weight	1
Shape	
Fill Shape?	☑ true
Labels	☐ false
Preview	

Data

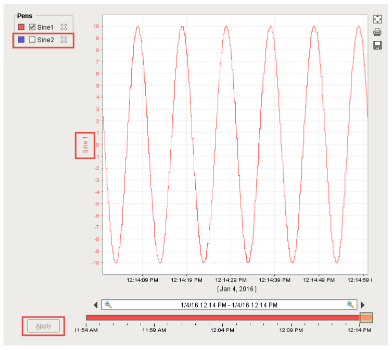
Tag Path	Sine/Sine2
Aggregation Mode	Min/Max

OK Cancel

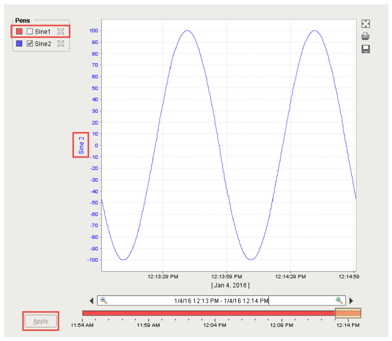
Once you press OK, you will immediately see Sine 1 and Sine 2 values change on the Easy Chart. On the right side of the chart, you can see the Sine 1 label and values. On the left side, you can see Sine 2 label and values. Now, you are using a different axis for each tag.



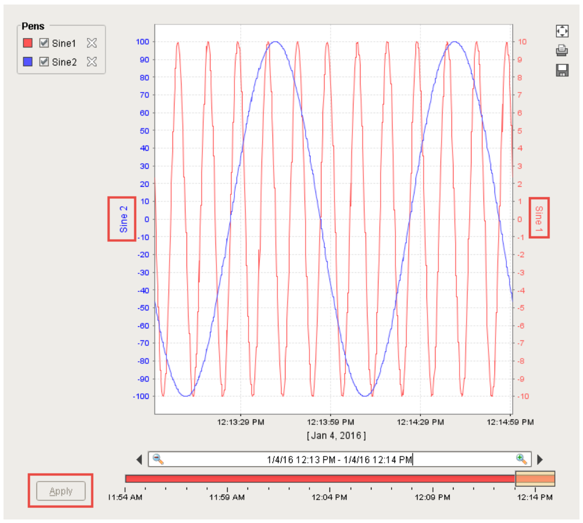
Go to **Preview Mode**, **uncheck Sine 2** and **click Apply**. Now, the Easy Chart will only show the Sine 1 axis for the Sine 1 values.



To see the chart for Sine 2 only, **check Sine 2**, **uncheck Sine 1** and **click Apply**. You will only see the values for Sine 2 which is on the Default axis.

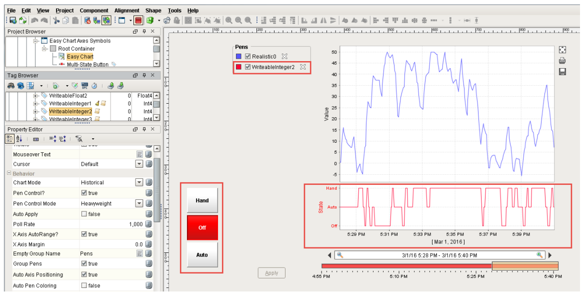


When looking at both axes in **Preview Mode**, you will see both values. Now, one Sine is not going to throw off the values for the other Sine. Select both pens and **click Apply**.



Symbol Axis Type Example

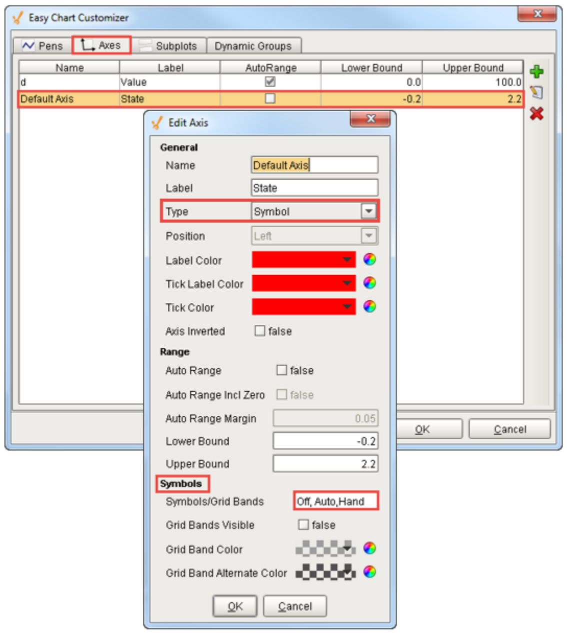
Another feature of the Easy Chart is the use of the **Symbol** axis type**.** The second subplot in this Easy Chart uses a Multi-State Button component to demonstrate the use of the **Symbol** type. The Multi-State Button component is bound to a tag named 'WritableInteger2'. Instead of using the default Numeric axis type to record the component's state as '**0, 1, 2**,' you can use **On**, **Off**, and **Auto** simply by typing the text in the Symbols field. This is helpful to an operator who immediately knows the state of the equipment instead of having to learn what the numeric values mean.



WriteableFloat2	0	Float4
WriteableInteger1	0	Int4
WriteableInteger2	0	Int4
WriteableInteger3	0	Int4

Mouseover Text	
Cursor	Default
Behavior	
Chart Mode	Historical
Pen Control?	✓ true
Pen Control Mode	Heavyweight
Auto Apply	☐ false
Poll Rate	1,000
X Axis AutoRange?	✓ true
X Axis Margin	0.0
Empty Group Name	Pens
Group Pens	✓ true
Auto Axis Positioning	✓ true
Auto Pen Coloring	☐ false

Go to the **Easy Chart Customizer**, click on the **Axes tab** and edit **Axis** for the WritableInteger2 pen. In this example, it is the Default Axis with the State label. You will notice that the type of axis selected is Symbol, lower and upper bound limits were set to provide a little padding at the top and bottom of the chart, and **Off, Auto, Hand** were entered in the **Symbols/Grid Bands** field. When entering text in the Symbols/Grid Bands field, always enter the text separated by a comma, and no spaces. Click **OK** two times.


Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Name	Label	AutoRange	Lower Bound	Upper Bound
d	Value	☑	0.0	100.0
Default Axis	State	☐	-0.2	2.2

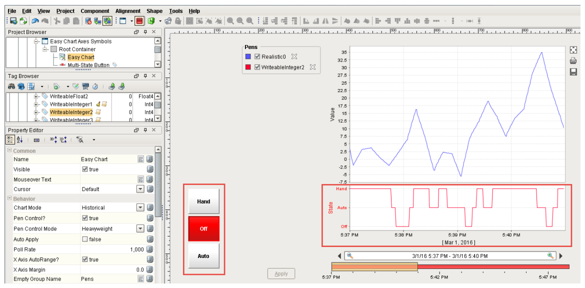
Edit Axis

General	
Name	Default Axis
Label	State
Type	Symbol
Position	Left
Label Color	🟥
Tick Label Color	🟥
Tick Color	🟥
Axis Inverted	☐ false
Range	
Auto Range	☐ false
Auto Range Incl Zero	☐ false
Auto Range Margin	0.05
Lower Bound	-0.2
Upper Bound	2.2
Symbols	
Symbols/Grid Bands	Off, Auto,Hand
Grid Bands Visible	☐ false
Grid Band Color	▨
Grid Band Alternate Color	▨

OK Cancel

Put your chart in **Preview Mode** and begin toggling the Multi-State buttons. You will then begin logging data to your Easy Chart. Data for the Multi-State Button component is collected in the second subplot which is using the Symbol axis type. The first subplot is data for another tag using the Numeric axis type.

This Easy Chart example shows the subplot feature of the Easy Chart component and how easy it is to break up the chart plot area into multiple distinct subplots sharing the same X axis. It is a good way to display lots of data from different tags in one Easy Chart.

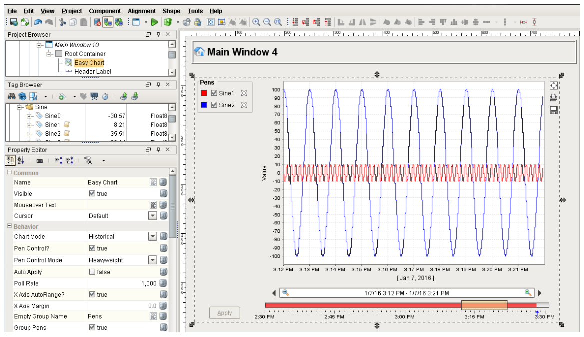


Next...

- Easy Chart - Subplots

Easy Chart - Subplots

The subplot feature of the Easy Chart component allows you to break up the chart plot area into multiple distinct subplots that share the 'X' axis, but they have their own 'Y' axis. By default, the Easy Chart has one subplot which is the main white area shown in the example below. Both of the Sine tags are inside this chart which is why you are seeing the history for both tags in this one area.

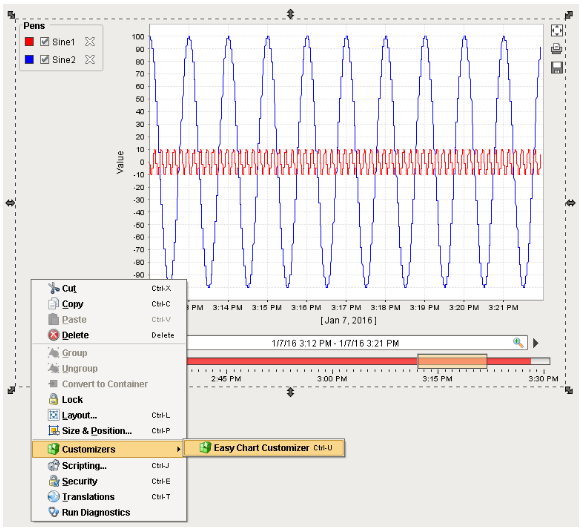


Tag	Value	Type
Sine0	-30.57	Float8
Sine1	8.21	Float8
Sine2	-35.51	Float8

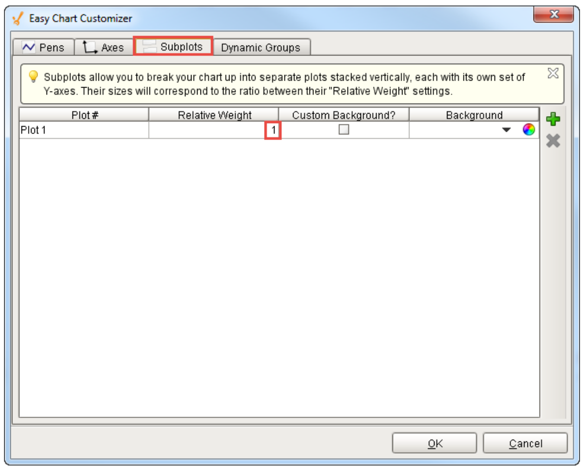
Property Editor

Common	
Name	Easy Chart
Visible	☑ true
Mouseover Text	
Cursor	Default
Behavior	
Chart Mode	Historical
Pen Control?	☑ true
Pen Control Mode	Heavyweight
Auto Apply	☐ false
Poll Rate	1,000
X Axis AutoRange?	☑ true
X Axis Margin	0.0
Empty Group Name	Pens
Group Pens	☑ true

It’s possible to break up your tags into multiple subplots which is often useful for digital data. In this example, the Sine 1 tag represents digital data in the very center of the chart, but perhaps you may want it to be its own subplot. **Right click** on the **Easy Chart component** and scroll down to **Customizers > Easy Chart Customizer**.



 The Subplots tab allows you to add one or more subplots to your Easy Chart. In this example, there is one subplot with a Relative Weight of 1 and a white Background.



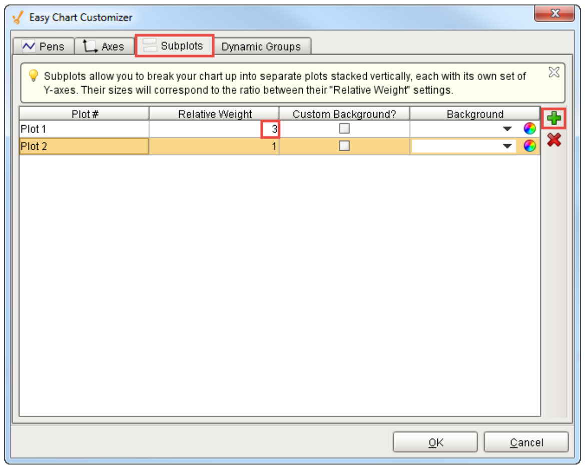
Subplots allow you to break your chart up into separate plots stacked vertically, each with its own set of Y-axes. Their sizes will correspond to the ratio between their "Relative Weight" settings.

Plot #	Relative Weight	Custom Background?	Background
Plot 1	1	☐	

Easy Chart - Subplots

Watch the Video

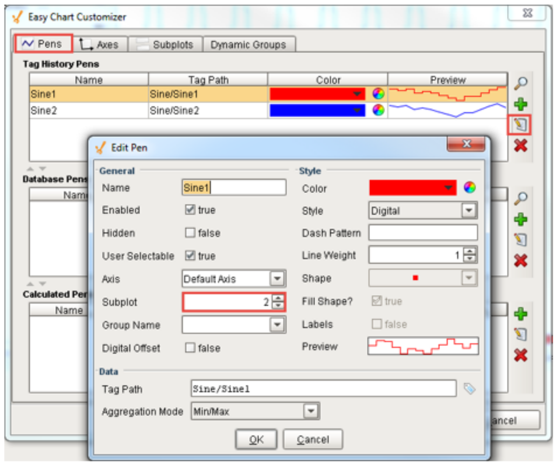
Click on the green plus icon to add a second Subplot and specify the Weight and the Custom Background. The Weight is the ratio between all the subplots. So, if you make the second subplot a 1:1 ratio, both subplots will share 50% of the space on the chart. In this example, make Subplot 2's Relative Weight **'1'** and Subplot 1's Relative Weight **'3'**, then Subplot 1 will be 3 times larger than Subplot 2. This is very common for digital data.



Subplots allow you to break your chart up into separate plots stacked vertically, each with its own set of Y-axes. Their sizes will correspond to the ratio between their "Relative Weight" settings.

Plot #	Relative Weight	Custom Background?	Background
Plot 1	3	☐	
Plot 2	1	☐	

Now, we have to put one of the pens into a different subplot. **Click on the Pens tab and select Sine 1.** Since Subplot 1 contains digital data, edit that pen to put it into Subplot 2. Increment the spinner to **'2'** or type the value **'2'** in the Subplot field. **Press OK**.



Tag History Pens

Name	Tag Path	Color	Preview
Sine1	Sine/Sine1	🟥	
Sine2	Sine/Sine2	🟦	

General

Name	Sine1
Enabled	☑ true
Hidden	☐ false
User Selectable	☑ true
Axis	Default Axis
Subplot	2
Group Name	
Digital Offset	☐ false

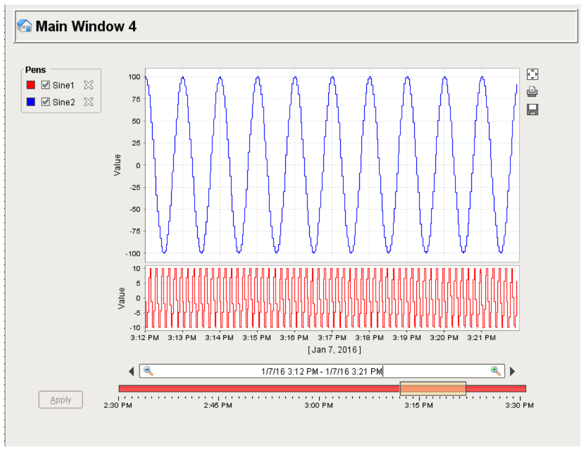
Style

Color	🟥
Style	Digital
Dash Pattern	
Line Weight	1
Shape	■
Fill Shape?	☑ true
Labels	☐ false
Preview	

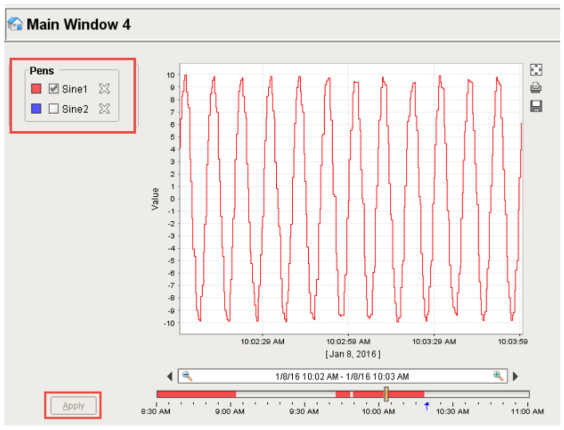
Data

Tag Path	Sine/Sine1
Aggregation Mode	Min/Max

Press OK again and now you have 2 different distinct plots on the Easy Chart. The second subplot on the chart has the digital data values and the top subplot has the Sine 2 tag values. You are not limited to the number of subplots on one Easy Chart.



You can view one subplot at a time by **unchecking the tag** in the Pens box and clicking **Apply**. The Easy Chart only shows subplots that have active tags.

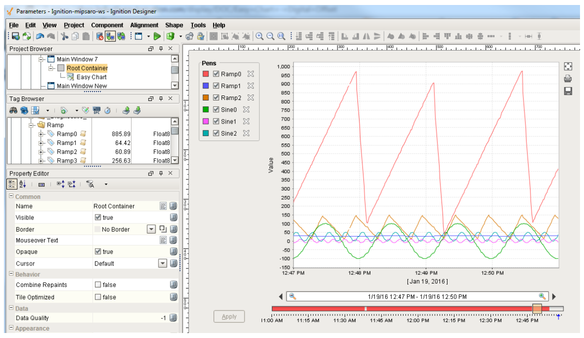


Next...

- Easy Chart - Pen Names and Groups

Easy Chart - Pen Names and Groups

You can organize pens on the Easy Chart by providing custom names and groups for each pen. By default, when you drag tags from the Tag Browser on to the Easy Chart component, the pen name is the same as name as the tag name. Also by default, all pens are placed into a single category called **'Pens.'**



Pens
- ☑ Ramp0
- ☑ Ramp1
- ☑ Ramp2
- ☑ Sine0
- ☑ Sine1
- ☑ Sine2

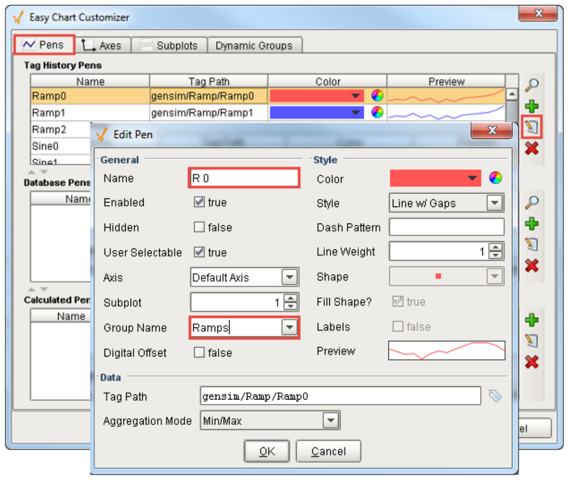
Ramp0	885.89	Float8
Ramp1	64.42	Float8
Ramp2	60.89	Float8
Ramp3	256.63	Float8

Property	Value
Name	Root Container
Visible	☑ true
Border	No Border
Mouseover Text	
Opaque	☑ true
Cursor	Default
Combine Repaints	☐ false
Tile Optimized	☐ false
Data Quality	-1

You may want to customize pen names and organize pens into different groups to make it easier for the operator. You can do this in the Customizer. **Right click on the Easy Chart component** and **scroll down to Customizers > Easy Chart Customizer.**

On the **Pens tab**, you can edit each of the individual pens and give them a different name as well as put them inside a different group.

Click the Edit icon and rename the first tag from **'Ramp0'** to **'R 0'** as shown in the following example.   At the bottom of the **Edit Pen window**, under **Group Name**, create a new group called **'Ramp s.'** If other groups exist, you can select one from the dropdown list.   In this example, this is the first group to be created, so simply type **'Ramps'** and **press OK**.



Tag History Pens

Name	Tag Path
Ramp0	gensim/Ramp/Ramp0
Ramp1	gensim/Ramp/Ramp1
Ramp2	
Sine0	

Edit Pen

General
- Name: R 0
- Enabled: ☑ true
- Hidden: ☐ false
- User Selectable: ☑ true
- Axis: Default Axis
- Subplot: 1
- Group Name: Ramps
- Digital Offset: ☐ false

Style
- Color
- Style: Line w/ Gaps
- Dash Pattern
- Line Weight: 1
- Shape
- Fill Shape?: ☑ true
- Labels: ☐ false
- Preview

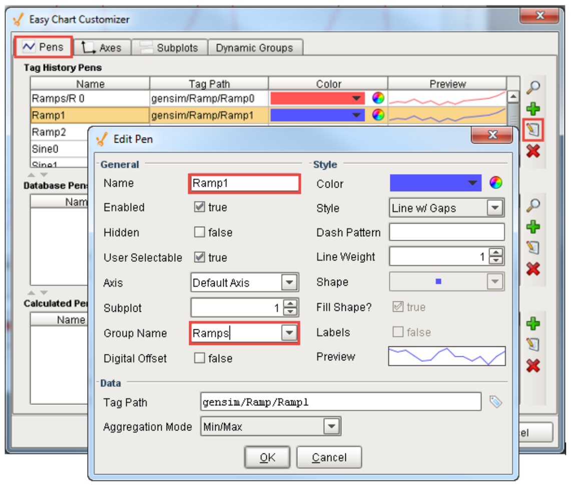
Data
- Tag Path: gensim/Ramp/Ramp0
- Aggregation Mode: Min/Max

OK   Cancel

Select the second pen and **click the Edit icon**. **Rename 'Ramp1'** to **'R 1'** and put it under the **'Ramps'** group by selecting it from the dropdown list, and **press OK**.

Easy Chart - Pen Names and Groups

Watch the Video


Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path
Ramps/R 0	gensim/Ramp/Ramp0
Ramp1	gensim/Ramp/Ramp1
Ramp2	
Sine0	

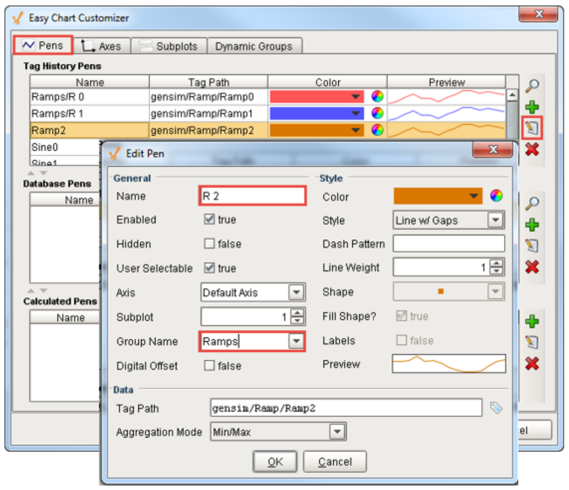
Edit Pen

General
Name: Ramp1
Enabled: true
Hidden: false
User Selectable: true
Axis: Default Axis
Subplot: 1
Group Name: Ramps
Digital Offset: false

Style
Color
Style: Line w/ Gaps
Dash Pattern
Line Weight: 1
Shape
Fill Shape?: true
Labels: false
Preview

Data
Tag Path: gensim/Ramp/Ramp1
Aggregation Mode: Min/Max

Let's do the same for the last Ramp pen. **Click the Edit icon** and **rename 'Ramp2'** to **'R 2'** and put it under the **'Ramps'** group, and **click OK.**


Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path
Ramps/R 0	gensim/Ramp/Ramp0
Ramps/R 1	gensim/Ramp/Ramp1
Ramp2	gensim/Ramp/Ramp2
Sine0	

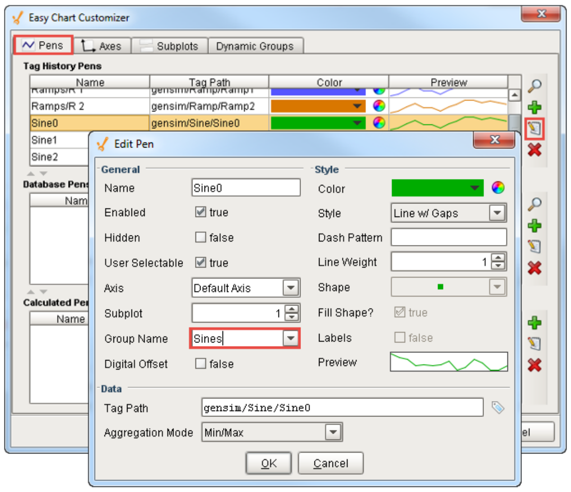
Edit Pen

General
Name: R 2
Enabled: true
Hidden: false
User Selectable: true
Axis: Default Axis
Subplot: 1
Group Name: Ramps
Digital Offset: false

Style
Color
Style: Line w/ Gaps
Dash Pattern
Line Weight: 1
Shape
Fill Shape?: true
Labels: false
Preview

Data
Tag Path: gensim/Ramp/Ramp2
Aggregation Mode: Min/Max

For the Sine names, keep the names the same and create a new group. **Select** and **Edit the fourth pen**. Create a new **Group Name** and call it **'Sines,'** and **press OK.**


Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path
Ramps/R 2	gensim/Ramp/Ramp2
Sine0	gensim/Sine/Sine0
Sine1	
Sine2	

Edit Pen

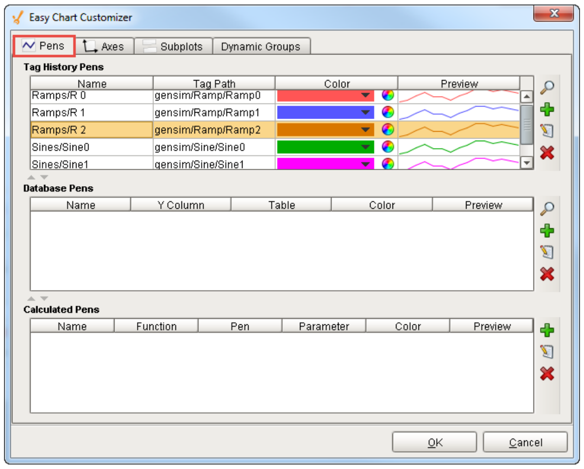
General
Name: Sine0
Enabled: true
Hidden: false
User Selectable: true
Axis: Default Axis
Subplot: 1
Group Name: Sines
Digital Offset: false

Style
Color
Style: Line w/ Gaps
Dash Pattern
Line Weight: 1
Shape
Fill Shape?: true
Labels: false
Preview

Data
Tag Path: gensim/Sine/Sine0
Aggregation Mode: Min/Max

Edit the remaining Sine pens and put each Sine pen into the **'Sines'** group using the dropdown list, and **press OK**.

Now that you customized the Name and the Group for each of the pens, you can see under the Pens tab all your updates in one view.   You can also change the pen and group names on this view by selecting the pen you want to update and editing the Name and/or the Group. It's a faster way to edit the Name and Group then having to go to the Edit icon for each pen.   Once all your edits are complete, **click OK.**



Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path	Color	Preview
Ramps/R 0	gensim/Ramp/Ramp0	🟥	
Ramps/R 1	gensim/Ramp/Ramp1	🟦	
Ramps/R 2	gensim/Ramp/Ramp2	🟧	
Sines/Sine0	gensim/Sine/Sine0	🟩	
Sines/Sine1	gensim/Sine/Sine1	🟪	

Database Pens

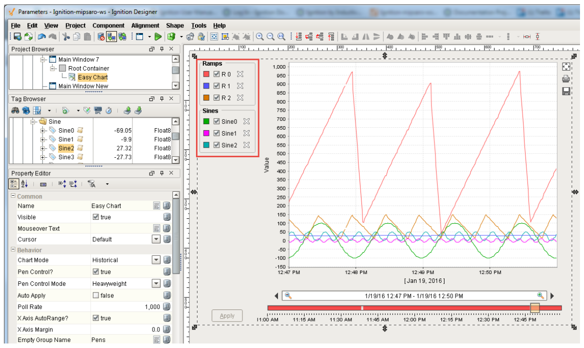
Name	Y Column	Table	Color	Preview

Calculated Pens

Name	Function	Pen	Parameter	Color	Preview

OK | Cancel

Now you see all the pen names that the operators will see on the Easy Chart as well as the 2 new groups for your pens: **'Ramps'** and **'Sines.'**

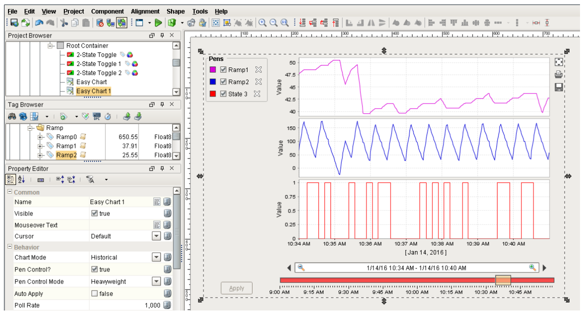


Next...

- Easy Chart - Pen Renderer

Easy Chart - Pen Renderer

You can customize the renderer of each pen on the Easy Chart to change its style, shape, weight and color. When you add tags to an Easy Chart component, the default renderer or the style of each pen is a simple line as you can see in each of the subplots in the example below.



Name	Easy Chart 1
Visible	true
Mouseover Text	
Cursor	Default
Chart Mode	Historical
Pen Control?	true
Pen Control Mode	Heavyweight
Auto Apply	false
Poll Rate	1,000

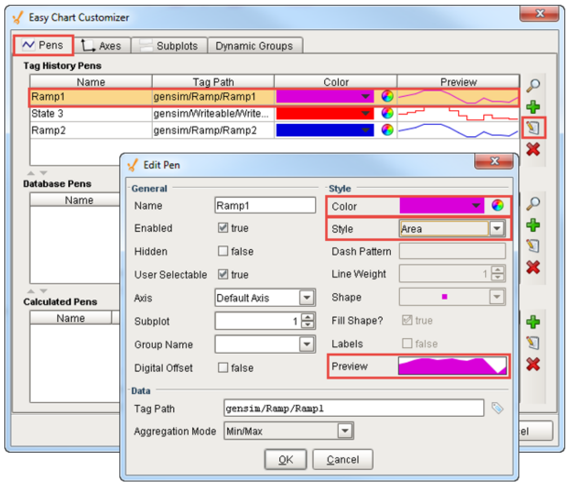
Easy Chart - Pen Renderer

Watch the Video

To customize the pen renderer, **right click on the Easy Chart component** and scroll down to **Customizers > Easy Chart Customizer**.

In the Pens tab, go to each of the pens and **click the Edit Icon**. On the right side of the Edit Pen window under Style, you can change the line color, style, weight, and shape. The Style, by default, is **'Line with Gaps,'** which means that if you lose communication to the PLC or don't have data for a particular time period, you will see a gap. You can change the Style from 'Line w/ Gaps' to any other style type listed in the dropdown that fits your needs and preferences. In this example, let's change the pen renderer for the 3 subplots on this Easy Chart.

Select the first pen, click the Edit Icon and change the color to your choice and change the Style to **'Area.'** You can preview your changes in the Preview field and **press OK**.



Tag History Pens

Name	Tag Path	Color	Preview
Ramp1	gensim/Ramp/Ramp1		
State 3	gensim/Writeable/Write...		
Ramp2	gensim/Ramp/Ramp2		

General

Name	Ramp1
Enabled	true
Hidden	false
User Selectable	true
Axis	Default Axis
Subplot	1
Group Name	
Digital Offset	false

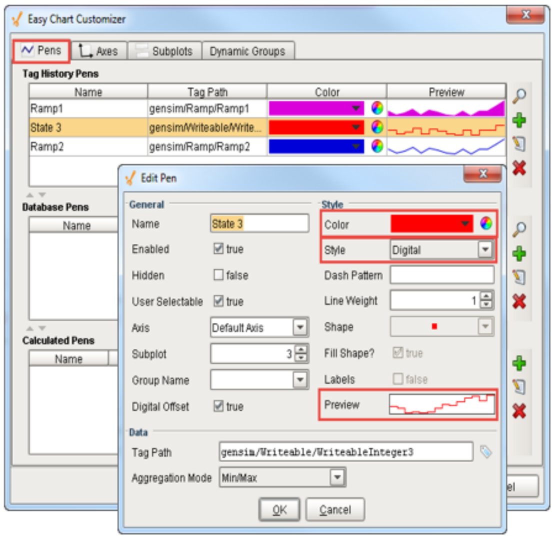
Style

Color	
Style	Area
Dash Pattern	
Line Weight	1
Shape	
Fill Shape?	true
Labels	false
Preview	

Data

Tag Path	gensim/Ramp/Ramp1
Aggregation Mode	Min/Max

Select the second pen, change the color, and set the Style to **'Digital'** since it's a State tag (On or Off / 0 or 1). Preview your changes and **press OK**.



Easy Chart Customizer — Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path	Color	Preview
Ramp1	gensim/Ramp/Ramp1	🟪	
State 3	gensim/Writeable/Write...	🟥	
Ramp2	gensim/Ramp/Ramp2	🟦	

Edit Pen

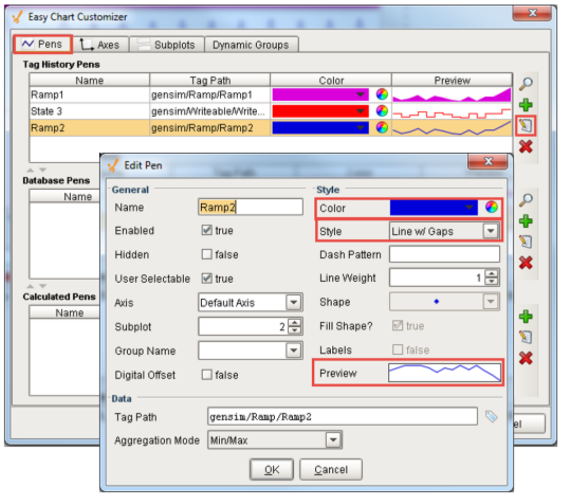
General		Style	
Name	State 3	Color	🟥
Enabled	☑ true	Style	Digital
Hidden	☐ false	Dash Pattern	
User Selectable	☑ true	Line Weight	1
Axis	Default Axis	Shape	
Subplot	3	Fill Shape?	☑ true
Group Name		Labels	☐ false
Digital Offset	☑ true	Preview	

Data

Tag Path: gensim/Writeable/WriteableInteger3

Aggregation Mode: Min/Max

Select the third pen, change the color and keep the default Style to **'Line w/ Gaps'.** Preview your changes and **press OK**. Once you set the Style of all 3 pens, **click OK**.



Easy Chart Customizer — Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path	Color	Preview
Ramp1	gensim/Ramp/Ramp1	🟪	
State 3	gensim/Writeable/Write...	🟥	
Ramp2	gensim/Ramp/Ramp2	🟦	

Edit Pen

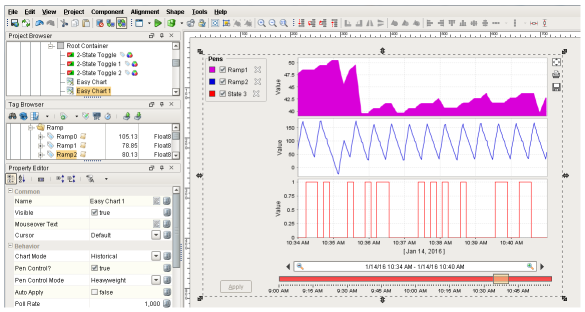
General		Style	
Name	Ramp2	Color	🟦
Enabled	☑ true	Style	Line w/ Gaps
Hidden	☐ false	Dash Pattern	
User Selectable	☑ true	Line Weight	1
Axis	Default Axis	Shape	
Subplot	2	Fill Shape?	☑ true
Group Name		Labels	☐ false
Digital Offset	☐ false	Preview	

Data

Tag Path: gensim/Ramp/Ramp2

Aggregation Mode: Min/Max

Now, your customized changes to the pen renderer for these 3 pens will be reflected on each subplot on the Easy Chart.



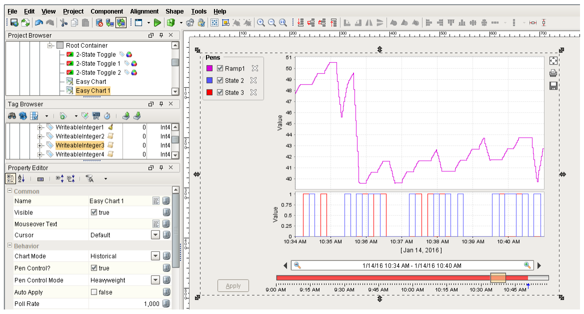
Next...

- Easy Chart - Digital Offset

Easy Chart - Digital Offset

Digital pens often share the same subplot on the **Easy Chart** component. You can offset digital pens in the subplot order to see the values better.

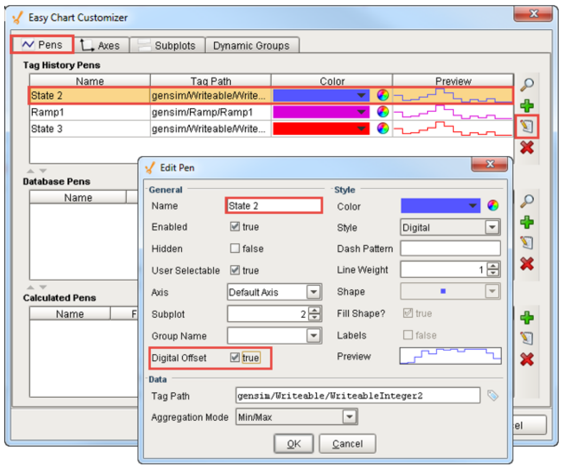
The Easy Chart below shows 2 digital pens on the same subplot: State 2 and State 3. The values are difficult to see because they overlap on top of each other.



Name	Easy Chart 1
Visible	☑ true
Mouseover Text	
Cursor	Default
Chart Mode	Historical
Pen Control?	☑ true
Pen Control Mode	Heavyweight
Auto Apply	☐ false
Poll Rate	1,000

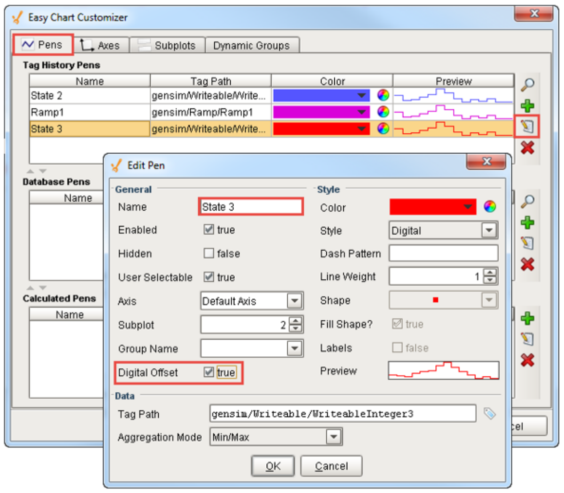
We can apply a digital offset on our digital renderer so we can see the values better. **Right click** on the **E asy Chart component** and scroll down to **Customizers > Easy Chart Customizer**.

In the Pens tab, go to each of the State pens and **check the 'Digital Offset' box** to **'true'** and **click OK**.



Name	Tag Path	Color
State 2	gensim/Writeable/Write...	🟦
Ramp1	gensim/Ramp/Ramp1	🟪
State 3	gensim/Writeable/Write...	🟥

Name	State 2	Color	🟦
Enabled	☑ true	Style	Digital
Hidden	☐ false	Dash Pattern	
User Selectable	☑ true	Line Weight	1
Axis	Default Axis	Shape	
Subplot	2	Fill Shape?	☑ true
Group Name		Labels	☐ false
Digital Offset	☑ true	Preview	
Tag Path	gensim/Writeable/WriteableInteger2		
Aggregation Mode	Min/Max		



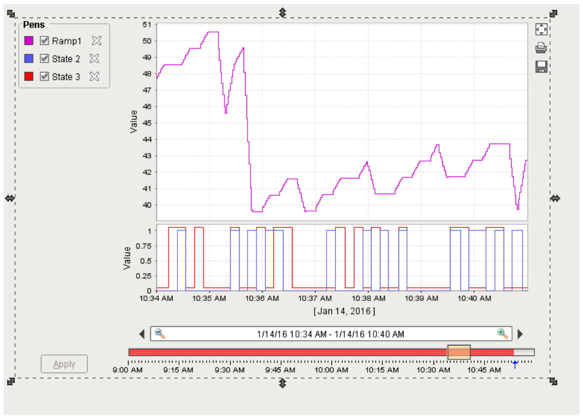
Name	Tag Path	Color
State 2	gensim/Writeable/Write...	🟦
Ramp1	gensim/Ramp/Ramp1	🟪
State 3	gensim/Writeable/Write...	🟥

Name	State 3	Color	🟥
Enabled	☑ true	Style	Digital
Hidden	☐ false	Dash Pattern	
User Selectable	☑ true	Line Weight	1
Axis	Default Axis	Shape	
Subplot	2	Fill Shape?	☑ true
Group Name		Labels	☐ false
Digital Offset	☑ true	Preview	
Tag Path	gensim/Writeable/WriteableInteger3		
Aggregation Mode	Min/Max		

Easy Chart - Digital Offset

Watch the Video

You will see a little offset in the values between the State pens so they don't overlap making it a lot easier for the operator to read the digital values.



Next...

- Easy Chart - Calculated Pens

Easy Chart - Calculated Pens

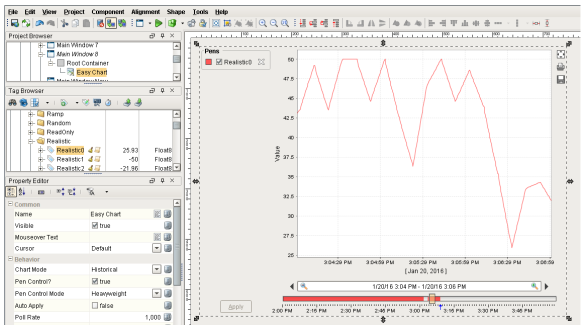
Calculated pens display a value that is dynamically calculated based on another pen, such as a Moving Average or an SPC Function including a Constant value, Upper Control Limit (UCL), Upper Warning Limit (UWL), Average (Avg), Lower Warning Limit (LWL), Lower Control Limit (LCL), Multiply, Min, and Max.

Using calculated pens on an Easy Cart component allows you to lay a graph on the calculation based off of another pen selected from the Function dropdown list in the Easy Chart Customizer.

To add a calculated pen, r**ight click on the Easy Chart component**, and **scroll down to Customizers > Easy Chart Customizer.**

Easy Chart - Calculated Pens

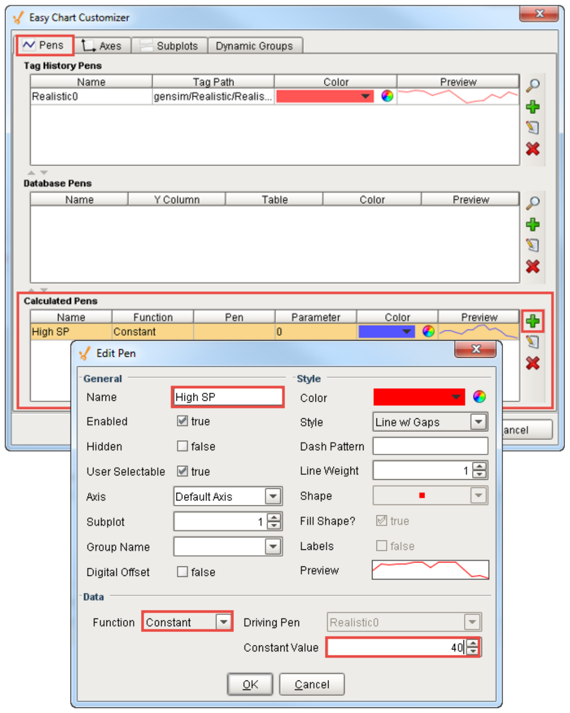
Watch the Video



At the bottom of the Pens tab there is a table for all the Calculated Pens. You can add one or more calculated pens by clicking the green plus icon.

As you add new pens from the the Edit Pen window, you need to give each pen a name. They are just like other pens where you can specify the Style, Color, Axis and Subplot. In the Function dropdown list, you can select which calculation you want to use. The type of functions are: Constant, UCL, UWL, Avg, LWL, LCL, MovingAvg, Multiply, Min and Max.

In this example, add a pen called **'High SP.' In the Function field, select 'Constant'**, set the Constant value at **'40'**, make the line color **'red'**, and **press OK**. **Press OK** again.



Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path	Color	Preview
Realistic0	gensim/Realistic/Realis...	🟥	

Database Pens

Name	Y Column	Table	Color	Preview

Calculated Pens

Name	Function	Pen	Parameter	Color	Preview
High SP	Constant		0	🟦	

Edit Pen

General

Name	High SP
Enabled	✓ true
Hidden	false
User Selectable	✓ true
Axis	Default Axis
Subplot	1
Group Name	
Digital Offset	false

Style

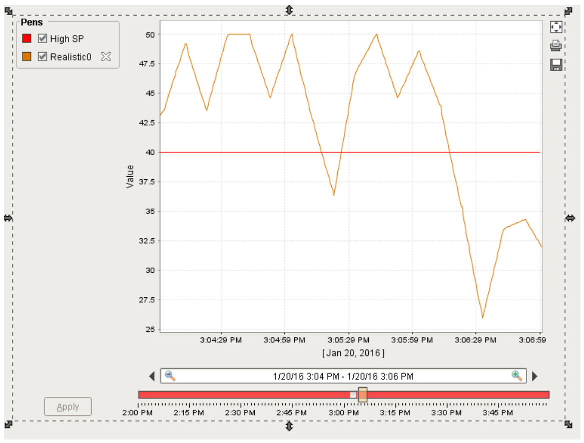
Color	🟥
Style	Line w/ Gaps
Dash Pattern	
Line Weight	1
Shape	
Fill Shape?	✓ true
Labels	false
Preview	

Data

Function	Constant
Driving Pen	Realistic0
Constant Value	40

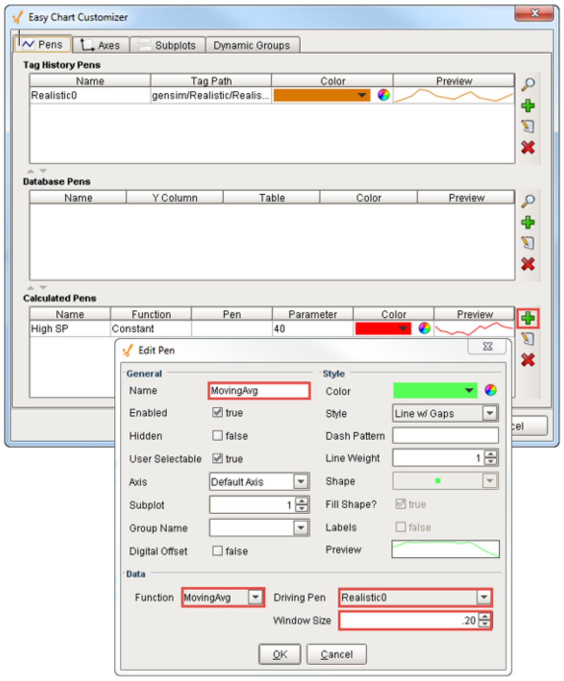
OK | Cancel

A red horizontal line will be drawn at the value of 40 on the Easy Chart for the High SP value.



Add a second calculated pen and call this one a **'MovingAvg.'**

On the Edit Pen window, name your pen **'Moving Average.' In the Function field**, select **'MovingAvg'** from the dropdown list. Set the Driving Pen to **'Realistic0.'** Specify the **Window Size** to the percentage of the total time period that you are going to do a Moving Average on. For this example, set it to **20 percent (0.2)** and **press OK**.



Name	Tag Path	Color	Preview
Realistic0	gensim/Realistic/Realis...	🟧	

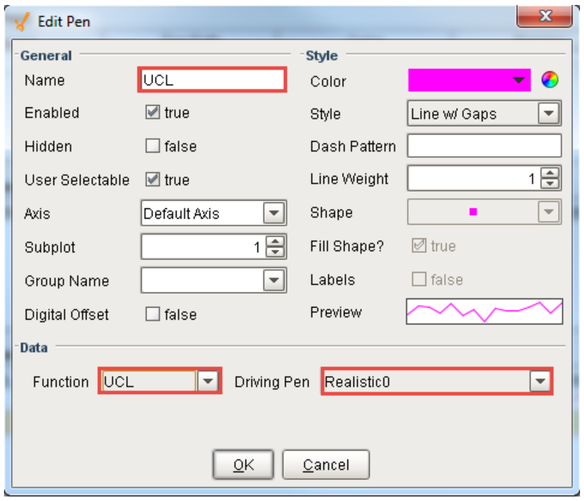
Name	Y Column	Table	Color	Preview

Name	Function	Pen	Parameter	Color	Preview
High SP	Constant		40	🟥	

Name	MovingAvg	Color	🟩
Enabled	✓ true	Style	Line w/ Gaps
Hidden	☐ false	Dash Pattern	
User Selectable	✓ true	Line Weight	1
Axis	Default Axis	Shape	
Subplot	1	Fill Shape?	✓ true
Group Name		Labels	☐ false
Digital Offset	☐ false	Preview	

Function: MovingAvg Driving Pen: Realistic0
Window Size: .20

Let's add 2 more calculated pens. Call the third pen **'UCL.' In the Function field**, **select 'UCL'** from the dropdown list and set the Driving Pen to **'Realistic0'**. **Press OK.**


Edit Pen

General

Name	UCL
Enabled	✓ true
Hidden	false
User Selectable	✓ true
Axis	Default Axis
Subplot	1
Group Name	
Digital Offset	false

Style

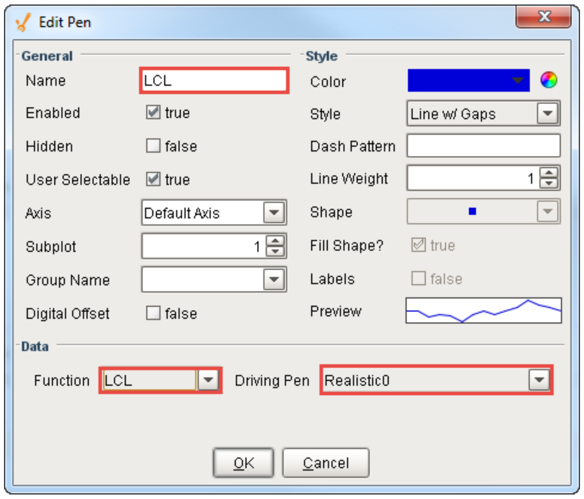
Color	🟪
Style	Line w/ Gaps
Dash Pattern	
Line Weight	1
Shape	■
Fill Shape?	✓ true
Labels	false
Preview	

Data

Function	UCL
Driving Pen	Realistic0

OK Cancel

Call the fourth pen **'LCL.'** **In the Function field, select 'LCL'** and set the Driving Pen to **'Realistic0.'** **P ress OK.**


Edit Pen

General

Name	LCL
Enabled	✓ true
Hidden	false
User Selectable	✓ true
Axis	Default Axis
Subplot	1
Group Name	
Digital Offset	false

Style

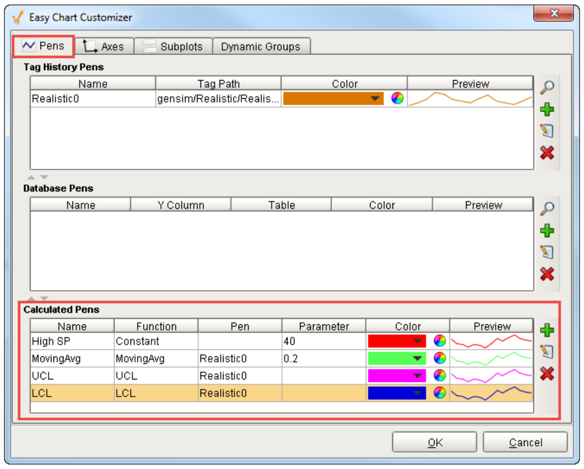
Color	🟦
Style	Line w/ Gaps
Dash Pattern	
Line Weight	1
Shape	■
Fill Shape?	✓ true
Labels	false
Preview	

Data

Function	LCL
Driving Pen	Realistic0

OK Cancel

Once you added all your pens, you will see all 4 pens listed under the Calculated Pens table: High SP, MovingAvg, UCL and LCL. **Press OK.**


Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path	Color	Preview
Realistic0	gensim/Realistic/Realis...	🟧	

Database Pens

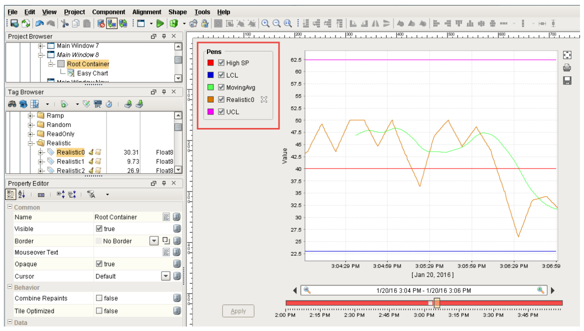
Name	Y Column	Table	Color	Preview

Calculated Pens

Name	Function	Pen	Parameter	Color	Preview
High SP	Constant		40	🟥	
MovingAvg	MovingAvg	Realistic0	0.2	🟩	
UCL	UCL	Realistic0		🟪	
LCL	LCL	Realistic0		🟦	

OK Cancel

Now you see all your calculations displayed on the Easy Chart. The green pen is your Moving Average of that value. The UCL and LCL are displayed by pink and blue pens respectively, as well as the Constant value displayed with a red line at 40.



Note

You cannot bind the Calculated Pen values inside the Easy Chart Customizer. To bind the function values, use the Cell Update Binding.

Next...

- Ad Hoc Charting

Ad Hoc Charting

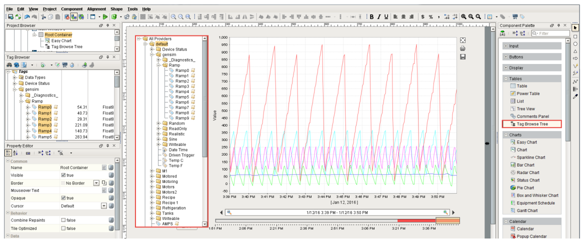
INDUCTIVE UNIVERSITY

Ad Hoc Charting

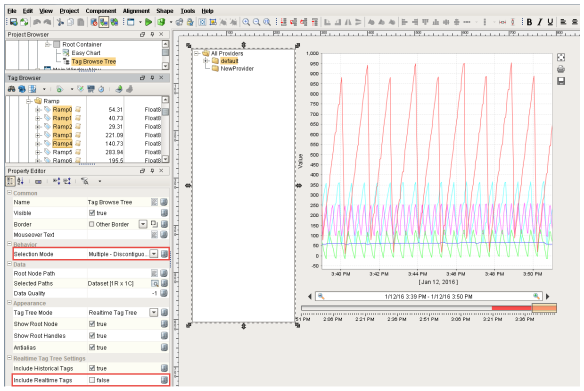
Watch the Video

In Designer, we can effectively make an Ad Hoc Chart where we can pick and choose which pens we want to put on the chart.   We can provide this same functionality for an operator in the Runtime by using a special built-in component called the Tag Browser Tree.

Drag the Tag Browse Tree component from the Tables Tab of the Component Palette to the left side of the chart. The Tag Browser Tree component can give an operator the same functionality as the Tag Browser in Runtime or Preview Mode. Expand the folders to see all the tags in your system. By default, the Tag Browser Tree component shows you all tags even those tags that are not logged in Historian.

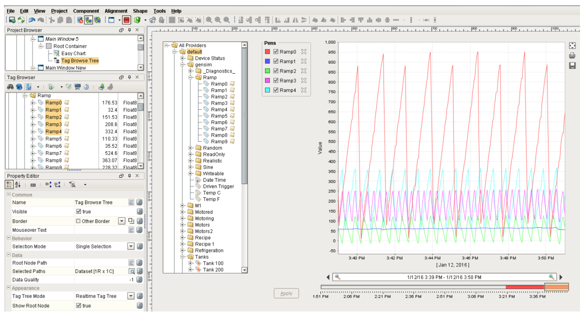


There are a couple of properties worth mentioning that will help you select tags on the Tag Browse Tree component. There is an **'Include Realtime Tag'** property in the Property Editor on the bottom left that you can uncheck to not include Realtime Tags in the event you only want to see Historical tags. In this example, **uncheck 'Include Realtime Tags.'** There is also another property called **'Selection Mode'** where you can make a single tag selection or you can select multiple tags. **Select 'Multiple – Discontiguous'** from the dropdown.



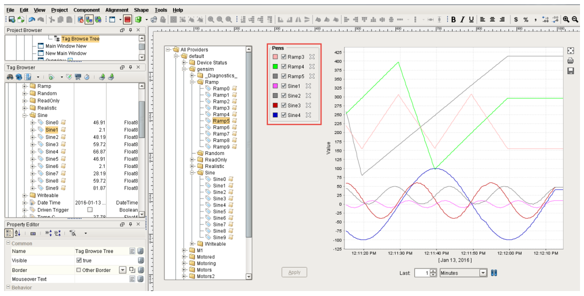
Property	Value
Common	
Name	Tag Browse Tree
Visible	☑ true
Border	☐ Other Border
Mouseover Text	
Behavior	
Selection Mode	Multiple - Discontiguo...
Data	
Root Node Path	
Selected Paths	Dataset [1R x 1C]
Data Quality	-1
Appearance	
Tag Tree Mode	Realtime Tag Tree
Show Root Node	☑ true
Show Root Handles	☑ true
Antialias	☑ true
Realtime Tag Tree Settings	
Include Historical Tags	☑ true
Include Realtime Tags	☐ false

By adding the Tag Browse Tree component on the window, each of the components knows exactly how to work together. Format the Tag Browse Tree component, pens and graph so you can see all three nicely on the window. **Click Save** and go into the **Runtime** or **Preview Mode**.



Now, you can browse for tags in the Tag Browse Tree. Select one tag or multiple tags and drag them on to the chart to make an Ad Hoc Chart. In this example, the **'Ramp'** and **'Sine'** tags were randomly selected to create the following Ad Hoc Chart.

You can press the **'X'** icon to the right of any of the pens to remove a tag from the chart. You can also remove all pens and go back to an empty chart, and pick and choose which of the tags you want to drag on to the chart, thus, creating an Ad Hoc Chart.



When working in the Designer, whatever pens you have on your Easy Chart when you saved, the same pens will also be displayed on the chart when the client is opened.

You may have multiple axes set up for your Easy Chart, but when dragging tags from the Tag Browse Tree component to an Easy Chart there is no way for the user to set which axis to use. Because of this limitation, any tag that is added in this way will attempt to match their Engineering Units property to an axis on the chart. If no match is found, the default axis will be used.

Next...

- Indirect Easy Chart

Indirect Easy Chart

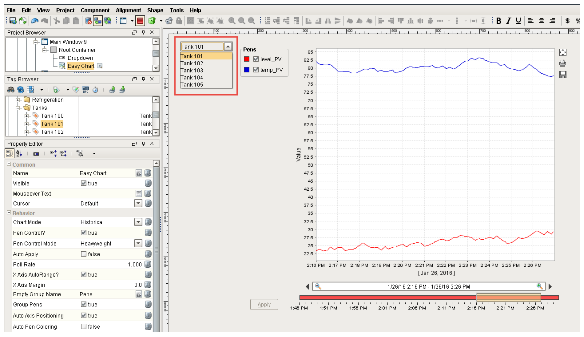
INDUCTIVE UNIVERSITY

Indirect Easy Chart

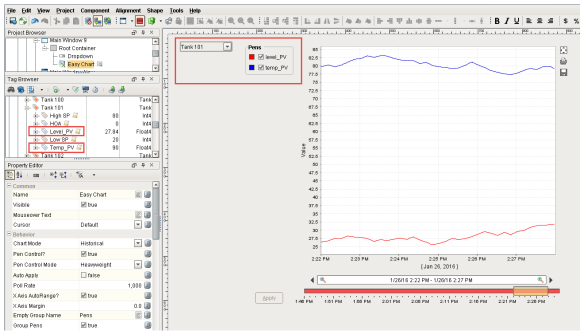
Watch the Video

It is possible for the Easy Chart component to be indirect and point to a set of historical tags based on any parameter using the Cell Update Binding type.

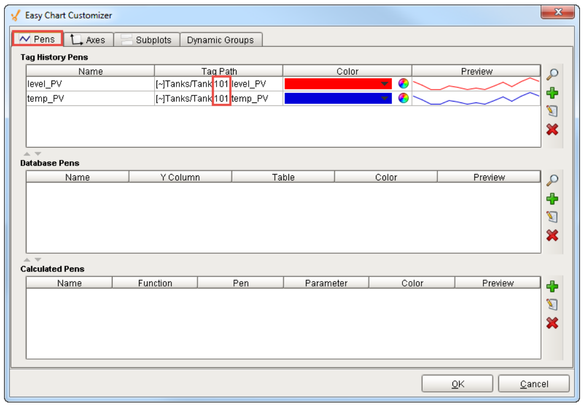
In this exercise, we are going to use a Tank dropdown list so an operator can select a specific tank and see the history of the **level_PV** and **temp_PV** tags for that tank.   We will be using Tank 101 to Tank 105 for this example. Fill in your dropdown component's data property as shown below.



Every tank is identical and has the same tags. They have a **level_PV** tag and a **temp_PV** tag which are set to log to the Historian as you can see by the scroll icon next to the tag name. Once you have Tank 101 selected, drag these 2 tags on to your chart so you can see the tank's history.



Right click on the **Easy Chart** and scroll down to **Customizers > Easy Chart Customizer** and you will see that the Tag Path is pointing to directly to Tank 101. The only difference between Tank 101 and the other tanks is the Tank number in the Tag_Path. You can manually point to a different Tag Path by replacing **'101'** in the Tag Path with a different tank number such as **'102.' Click OK.**


Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path	Color	Preview
level_PV	[~]Tanks/Tank101/level_PV	🟥	
temp_PV	[~]Tanks/Tank101/temp_PV	🟦	

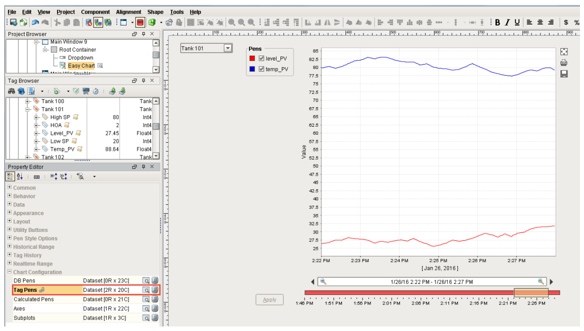
Database Pens

Name	Y Column	Table	Color	Preview

Calculated Pens

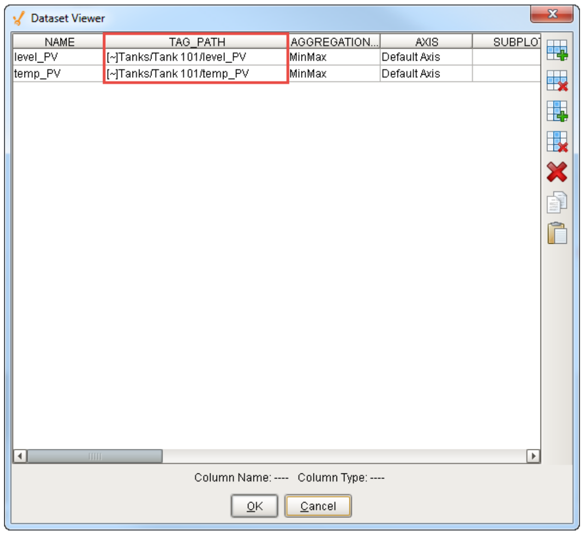
Name	Function	Pen	Parameter	Color	Preview

Now lets use the Cell Update Binding type to change the Tag_Path. Go to your Easy Chart component and scroll down the **Property Editor** to the property called **Tag Pens**. You will notice that the Tag Pens property is a Dataset.


Property Editor — Chart Configuration

DB Pens	Dataset [0R x 23C]
Tag Pens	Dataset [2R x 20C]
Calculated Pens	Dataset [0R x 21C]
Axes	Dataset [1R x 22C]
Subplots	Dataset [1R x 3C]

Click on the **Dataset icon** and you will see 2 rows. The second column is the Tag_Path that you need to modify for each row in order for it to change dynamically when an operator selects a different tank. **Click OK.**

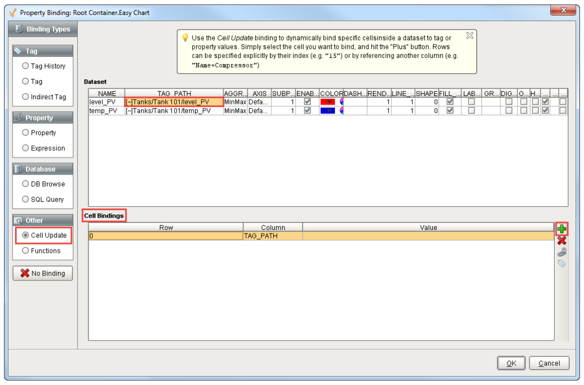

Dataset Viewer

NAME	TAG_PATH	AGGREGATION...	AXIS	SUBPLOT
level_PV	[~]Tanks/Tank 101/level_PV	MinMax	Default Axis	
temp_PV	[~]Tanks/Tank 101/temp_PV	MinMax	Default Axis	

Column Name: ---- Column Type: ----

To change an individual cell of a dataset, you have to bind a dataset to a Cell Update Binding. In a **Cell Update Binding,** you can select one or more cells and dynamically bind them to a property or to an existing tag in the system. To change the **temp_PV** and **level_PV** tag paths dynamically, use the following instructions:

1. Click on the **Binding icon** for the Tag Pens property.
2. When the Property Binding window opens, click the **Cell Update** button.
3. In the Dataset table, select the first row under the Tag_Path column for **level_PV**.
4. Click the **green plus icon** under the Cell Binding table. A row will be added in the Cell Bindings table for **level_PV**.
5. **Repeat steps 3** and **4** to add a second row under the Tag_Path column for **temp_PV**.


Property Binding: Root Container.Easy Chart

Use the *Cell Update* binding to dynamically bind specific cellsinside a dataset to tag or property values. Simply select the cell you want to bind, and hit the "Plus" button. Rows can be specified explicitly by their index (e.g. "15") or by referencing another column (e.g. "Name=Compressor")

Dataset

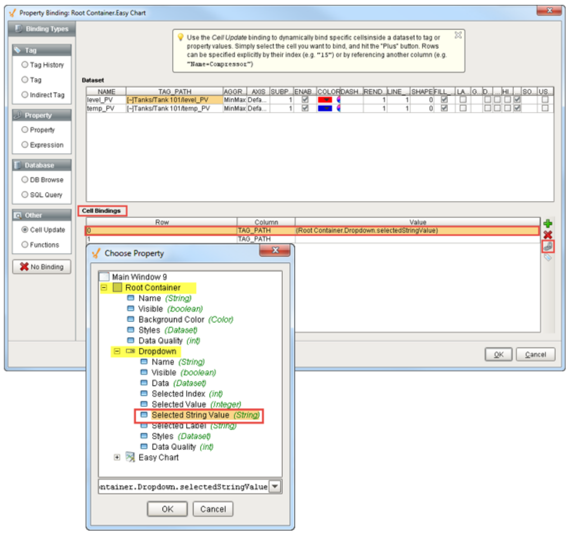
NAME	TAG_PATH	AGGR...	AXIS	SUBP...	ENAB...	COLOR	DASH...	REND...	LINE_...	SHAPE	FILL_...	LAB...	GR...	DIG...	O...	H...	...	...	...
level_PV	[~]Tanks/Tank 101/level_PV	MinMax	Defa...	1	✓	🟥		1	1	0	✓	☐		☐	☐	☐	✓		☐
temp_PV	[~]Tanks/Tank 101/temp_PV	MinMax	Defa...	1	✓	🟦		1	1	0	✓	☐		☐	☐	☐	✓		☐

Cell Bindings

Row	Column	Value
0	TAG_PATH	

6. Now, you need to update the Value column in the Cell Bindings table for each tag. Select the first row under the Cell Binding table and **click the Link icon** on the right side of the window.
7. When the property window opens, navigate to **Root Container > Dropdown** and click on the **'Selected String Value'**. **Click OK**.

8. **Repeat steps 6** and **7** to add the second row under the Cell Bindings table.



Property Binding: Root Container.Easy Chart

Use the *Cell Update* binding to dynamically bind specific cellsinside a dataset to tag or property values. Simply select the cell you want to bind, and hit the "Plus" button. Rows can be specified explicitly by their index (e.g. "15") or by referencing another column (e.g. "Name=Compressor")

Dataset

NAME	TAG_PATH	AGGR...	AXIS	SUBP...	ENAB...	COLOR	DASH...	REND...	LINE_...	SHAPE	FILL_...
level_PV	[~]Tanks/Tank 101/level_PV	MinMax	Defa...	1	✓	🟥		1	1	0	✓
temp_PV	[~]Tanks/Tank 101/temp_PV	MinMax	Defa...	1	✓	🟦		1	1	0	✓

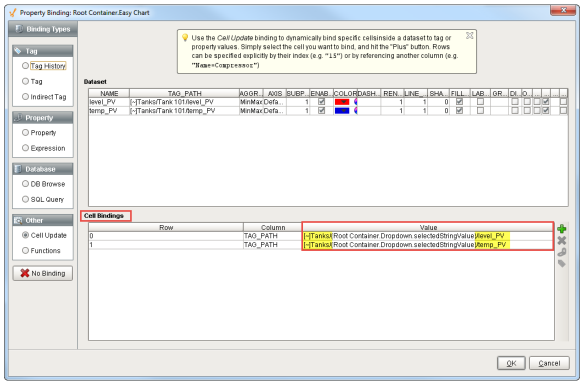
Cell Bindings

Row	Column	Value
0	TAG_PATH	{Root Container.Dropdown.selectedStringValue}
1	TAG_PATH	

9. Next, under the Cell Bindings table, update the Value column with the folder path and tag name for each row. The first row at the top is the level_PV tag and the second row is temp_PV based on the row indexes. Double click on the first row and enter the highlighted text as shown below to match your tag paths. Repeat this step for the second row.

```
[~]Tanks/{Root Container.Dropdown.selectedStringValue}/level_PV
```

```
[~]Tanks/{Root Container.Dropdown.selectedStringValue}/temp_PV
```



Property Binding: Root Container.Easy Chart

Use the *Cell Update* binding to dynamically bind specific cellsinside a dataset to tag or property values. Simply select the cell you want to bind, and hit the "Plus" button. Rows can be specified explicitly by their index (e.g. "15") or by referencing another column (e.g. "Name=Compressor")

Dataset

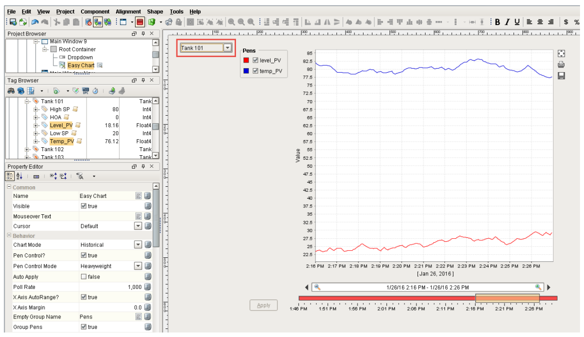
NAME	TAG_PATH	AGGR...	AXIS	SUBP...	ENAB...	COLOR	DASH...	REN...	LINE_...	SHA...	FILL...	LAB...
level_PV	[~]Tanks/Tank 101/level_PV	MinMax	Defa...	1	✓	🟥		1	1	0	✓	☐
temp_PV	[~]Tanks/Tank 101/temp_PV	MinMax	Defa...	1	✓	🟦		1	1	0	✓	☐

Cell Bindings

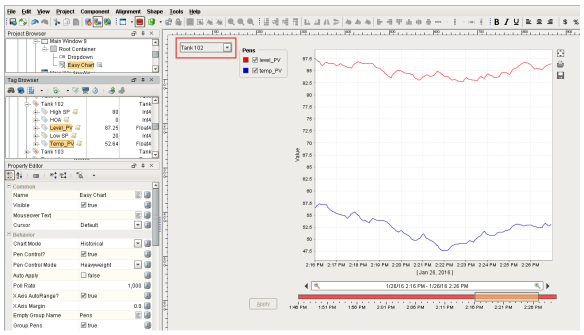
Row	Column	Value
0	TAG_PATH	[~]Tanks/{Root Container.Dropdown.selectedStringValue}/level_PV
1	TAG_PATH	[~]Tanks/{Root Container.Dropdown.selectedStringValue}/temp_PV

10. Once you added the folder path and tag name for each row, **click OK**.

Now the chart is indirect and when an operator selects a different tank, the **level_PV** and **temp_PV** tags will change.



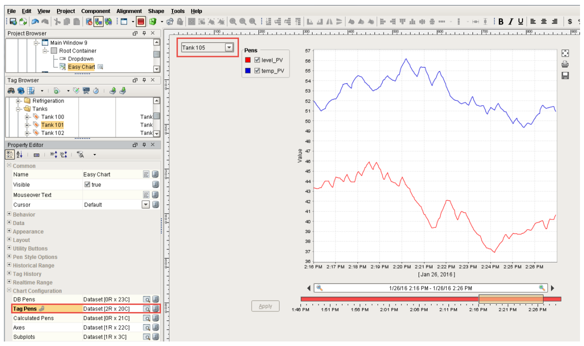
Go to your dropdown list and change from **Tank 101** to **Tank 102**. You will see the history for **Tank 102** come through on the chart.



Tag	Value	Type
Tank 102		Tank
High SP	80	Int4
HOA	0	Int4
Level_PV	87.25	Float4
Low SP	20	Int4
Temp_PV	52.64	Float4
Tank 103		Tank

Property	Value
Common	
Name	Easy Chart
Visible	true
Mouseover Text	
Cursor	Default
Behavior	
Chart Mode	Historical
Pen Control?	true
Pen Control Mode	Heavyweight
Auto Apply	false
Poll Rate	1,000
X Axis AutoRange?	true
X Axis Margin	0.0
Empty Group Name	Pens
Group Pens	true

Once again, go to your dropdown list and select **Tank 105** confirming that the **level_PV** and **temp_PV** tags are pointing to the correct values for that tank by looking at the Dataset property.



Property	Value
Common	
Name	Easy Chart
Visible	true
Mouseover Text	
Cursor	Default
Chart Configuration	
DB Pens	Dataset [0R x 23C]
Tag Pens	Dataset [2R x 20C]
Calculated Pens	Dataset [0R x 21C]
Axes	Dataset [1R x 22C]
Subplots	Dataset [1R x 3C]

Historian Tables

Use the Table component in the Designer to display the historical data. Easily drag-and-drop historian-enabled tags onto a table to display historical values in many different ways depending on how you have customized the table.

In this section ...

Table - Most Recent

You can use the **Tag History** binding type to bring back the most recent history for a set of tags.

Table – Most Recent

Watch the Video

To bring back the most recent history

1. Drag a **Table** (or a Power Table) from the Component Palette into your window.
2. In the Property Editor, go to the **Data** property, and click the binding icon.
   The Property Binding window is displayed.
3. Select the **Tag History** option.
4. From **Available Historical Tags**, find the desired historical tags by selecting and expanding the historical providers.
5. Once you have found the tag, drag it into to the right into **Selected Historical Tags**.
6. For **Date Range**, select **Realtime** from the drop-down list, and in the **Most Recent** text box, enter the amount of time in the past.
7. Select the appropriate **Aggregation Mode**.
8. Set **Polling Mode** to **Relative** to see table update at the project's Base Rate.
9. Click **OK**.

Next...

- Table - Data Range

Table - Data Range

You can use the **Tag History** binding type to bring back the history in a certain date range.

INDUCTIVE UNIVERSITY

Table – Data Range

Watch the Video

To bring back the history in a data range

1. Drag a **Table** (or a Power Table) from the Component Palette into your window.
2. In the Property Editor, go to the **Data** property, and click the binding icon.
   The Property Binding window is displayed.
3. Select the **Tag History** option.
4. From **Available Historical Tags**, find the desired historical tags by selecting and expanding the historical providers.
5. Once you have found the tag, drag it into to the right into **Selected Historical Tags**.
6. Enter a desired start and end date in the **Start Date Binding** and **End Date Binding** text boxes.
   If you want to enable the user to change the date range, bind these to the **Start Date** and **End Date** properties of a **Date** component.
7. Select the appropriate **Aggregation Mode**.
8. Set **Polling Mode** to **Relative** to see table update at the project's Base Rate.
9. Click **OK**.

Next...

- Table - Fixed Sample Size

Table - Fixed Sample Size

You can use the **Sample Size** and **Aggregation Mode** on the **Tag History** binding type to fix the number of records that are retrieved. This will ensure that the number of rows will remain the same without regard to the size of the data-set.

Selecting the **Min/Max** aggregation mode returns two rows of data for every row requested. Each pair represents a minimum and a maximum result from the underlying data. Therefore, a table with a fixed length, would return double the requested amount with Min/Max aggregation mode selected. With Min /Max aggregation mode selected, and with a fixed row length of one, the data set returns the oldest tag value of the time range.

Table – Fixed Sample Size

Watch the Video

To fix the number of retrieved records

1. Drag a **Table** (or a Power Table) from the Component Palette into your window.
2. In the Property Editor, go to the **Data** property, and click the binding icon.
   The Property Binding window is displayed.
3. Select the **Tag History** option.
4. From **Available Historical Tags**, find the desired historical tags by selecting and expanding the historical providers.
5. Once you have found the tag, drag it into to the right into **Selected Historical Tags**.
6. Enter a desired start and end date in the **Start Date Binding** and **End Date Binding** text boxes. To allow the user to change the date range, bind these to the **Start Date** and **End Date** properties of a **Date** component.
7. Select the appropriate **Aggregation Mode**.
8. For the **Sample Size**, select **Fixed** from the drop-down and enter the desired number of rows to be returned.
9. Set **Polling Mode** to **Relative** to see table update at the project's Base Rate.
10. Click **OK**.

Next...

- Table - Interval Sample Size

Table - Interval Sample Size

Selecting **Interval** for **Sample Size** will return a data set with the number of rows determined by the interval. For example, if the range of time is one hour and the interval is one minute, the number of rows in the data set will be sixty one. This is because there are sixty rows that represent sixty minutes in an hour plus an additional row that represents the current values that are building an interval.

Table – Interval Sample Size

Watch the Video

To set the Interval Sample Size

1. Drag a **Table** (or a Power Table) from the Component Palette into your window.
2. In the Property Editor, go to the **Data** property, and click the binding icon.
   The Property Binding window is displayed.
3. Select the **Tag History** option.
4. From **Available Historical Tags**, find the desired historical tags by selecting and expanding the historical providers.
5. Once you have found the tag, drag it into to the right into **Selected Historical Tags**.
6. Enter a desired start and end date in the **Start Date Binding** and **End Date Binding** text boxes. If you want to enable the user to change the date range, bind these to the **Start Date** and **End Date** properties of a **Date** component.
7. Select the appropriate **Aggregation Mode**.
8. For the **Sample Size**, select **Interval** from the drop-down and enter the desired number of rows to be returned.
9. Set **Polling Mode** to **Relative** to see table update at the project's Base Rate.
10. Click **OK**.

Next...

- Indirect Tag History Binding

Indirect Tag History Binding

You can use indirection to configure the tag history binding to point to a set of tags based on any parameter on the screen.

INDUCTIVE UNIVERSITY

Indirect Tag History Binding

Watch the Video

To set indirection on Tag History binding

1. Drag a **Table** (or a Power Table) from the Component Palette into your window.
2. In the Property Editor, go to the **Data** property, and click the binding icon.
   The Property Binding window is displayed.
3. Select the **Tag History** option.
4. From **Available Historical Tags**, find the desired historical tags by selecting and expanding the historical providers.
5. Once you have found the tag, drag it into to the right into **Selected Historical Tags**.
6. Open the **Indirection** box by clicking the plus sign next to it.
7. Double-click on one of the tags and enter a **{1}** in the part of the tag path that you want to create the indirection on.
   A new reference will appear in the indirection box. Bind the variable to the location.
8. Enter a desired start and end date in the **Start Date Binding** or **End Date Binding** text boxes.
   A better option is to bind these to the date properties of date components.
9. Select the appropriate **Aggregation Mode**.
10. For **Sample Size**, select the **Interval** option from the drop-down, and enter the desired interval spacing.
11. Set **Polling Mode** to **Relative** to see table update at the project's base rate.
12. Click **OK**.

Next...

Tag History Aggregates

Sample Size and Aggregation Mode

Tag History Aggregates

Watch the Video

The sample size determines how the query results will look.

- **Natural** - A Natural query will look up the logging rate for the queried tags (when possible), and return results spaced apart at that rate. This means that the return size will vary with the date range.
- **On Change/Raw** - An On Change query will return points as they were logged, and can be thought of as a "raw" query mode. This means that the results may not be evenly spaced. Also, it is important to note that every changed value will result in a row, and therefore if you are querying multiple tags and once, you may end up with more rows than you anticipated. For example, if tag A and tag B both change, you would end up with [[A_0, B_0],[A_1, B_0], [A_1, B_1]].

  Note: If you want to essentially retrieve raw values, while coalescing them down into fewer rows, try using the Interval sample mode, with an interval set to your largest acceptable time between rows, and select "prevent interpolation" from the advanced settings.
- **Fixed** - A Fixed query will return the given number of rows. Where data was sparse, interpolated values will be added. Where data is dense, the Aggregation Mode will come into play.
- **Interval** - Similar to fixed, but with the spacing based on time, rather than the number of requested results.

The aggregation mode dictates what happens when multiple raw values are encountered for a given sample window (the size of which is determined by the number of requested rows, or the interval size).

- **Min/Max** - The minimum and maximum values will be returned for the window. In other words, two rows will be returned. If only one value is seen in the interval, only one row will be returned.
- **Time-weighted Average** - The values are averaged together, weighted for the amount of time they cover in the interval.
- **Closest Value** - The value closest to the ending time of the interval will be returned.
- **Simple Average** - The values are summed together and divided by the number of values.
- **Sum** - The values in the interval are summed together.
- **Maximum** - The maximum value in the interval.
- **Minimum** - The minimum value in the interval.
- **Duration On** - Returns the number of seconds that the value was recorded as non-zero.
- **Duration Off** - Returns the number of seconds that the value recorded as zero.
- **Count On** - Returns the number of times the tag's value went from a zero value to non-zero.
- **Count Off** - Returns the number of times the tag's value changed from a non-zero value to zero.

  The following feature is new in Ignition version **7.8.1**
  Click here to check out the other new features

- **Count** - Returns the number of times a value was recorded

  The following feature is new in Ignition version **7.8.3**
  Click here to check out the other new features

- **Percent Good** - Time-weighted percentage of good values over the date range.
- **Percent Bad** - Time-weighted percentage of bad values over the date range.
- **Range** - Returns the range between the highest and lowest value for the period.
- **Standard Deviation** - Returns the standard deviation of values, or how much spread is present in the data; low standard deviation shows the values are close to the mean, and high standard deviation shows that the data points are spread out over a large range of values. Only good quality values are used when calculating.
- **Variance** - Returns the variance of values. Similar in concept Standard Deviation. Only good quality values are used when calculating.

Next...

Custom Tag History Aggregates

Python Aggregation Functions

Introduction

Ignition 7.8 introduces the ability to extend the “aggregation” functions provided by the tag history system with custom functions written in python. These functions are used for calculations across timeframes, and process multiple values in a “window” into a single result value.

For example, if a query defines a single row result, but covers an hour of time (either by requesting a single row, or using the Tag Calculations feature), the system must decide how to combine the values. There are many built in functions, such as Average, Sum, Count, etc. Using a custom python aggregate, however, allows you to extend these functions and perform any type of calculation.

Example

```
# this is a simple count function, called for each value in a time window
def count(qval, interpolated, finished, blockContext, queryContext):
        cnt = blockContext.getOrDefault('cnt',0)
        if qval.quality.isGood():
                blockContext['cnt']=int(cnt)+1

        if finished:
                return blockContext.getOrDefault('cnt', 0)
```

Parameters

- qval - The incoming QualifiedValue. This has:
    - value : Object
    - quality : Quality (which has ‘name’, ‘isGood()’)
    - timestamp : Date
- interpolated - Boolean indicating if the value is interpolated (true) or raw (false)
- finished - Boolean indicating that the window is finished. If true, the return of this particular call is what will be used for the results. If false, the return will be ignored.
- blockContext - A dictionary created fresh for this particular window. The function may use this as temporary storage for calculations. This object also has:
    - blockId - Integer roughly indicating the row id (doesn’t take into account aggregates that return multiple rows)
    - blockStart - Long UTC time of the start of the window
    - blockEnd - Long UTC time of the end of the window
    - previousRawValue - QualifiedValue, the previous non-interpolated value received before this window
    - previousBlockResults - QualifiedValue[], the results of the previous window.
    - insideBlock(long) - Returns boolean indicating if the time is covered by this window.
    - get(key, default) - A helper function that conforms to python’s dictionary “get with default return”.
- queryContext - A dictionary that is shared by all windows in a query. It also has:
    - queryId - String, an id that can be used to identify this query in logging
    - blockSize - Long, time in ms covered by each window
    - queryStart - Long, the start time of the query
    - queryEnd - Long, the end time of the query
    - logTrace(), logDebug(), logInfo() - all take (formatString, Object... args).

Return Value

- Object - Turned into Good Quality qualified value
- List - Used to return up to 2 values per window
- Tuple - (value, quality_int)
- List of quality tuples

Description

As values come in, they will be delivered to this function. The interpolator will create and deliver values.

For each window (or “data block”, the terms are used synonymously), the function will get a fresh copy of blockContext. The block context is a dictionary that can be used to as a memory space. The function should not use global variables. If values must be persisted across blocks, they can be stored in the queryContext, which is also a dictionary.

The function can choose what data to include, such as allowing interpolation or not, and allowing bad quality or not.

The window will receive the following values, many of which are generally interpolated (unless a raw value happens to fall exactly at the time):

1. The start of the window
2. 1 ms before each raw value (due to the difference between discrete and analog interpolation. A value equal to the previous raw value indicates discrete interpolation)
3. The raw value
4. The end of the window.

At the end of the window, the function will be called with “finished=true”. The function should return the calculated value(s). The resulting value will have a timestamp that corresponds to the beginning of the block timeframe.

Usage

Custom python aggregates can be used in two ways:

1. Defined as a shared script, where the full path to the function is passed to the query.
2. Defined as a string, prefaced with “python:”, and passed to the query.

Currently both options are only available through the system.tag.queryTagHistory/queryTagCalculations functions.

Both of these options are used with the “aggregationMode” and “aggregationModes” parameters to system.tag.queryTagHistory, and the “calculations” parameter of system.tag.queryTagCalculations. If the value is not an Enum value from the defined AggregationModes, it will be assumed to be a custom aggregate. The system will first see if it’s the path to a shared script, and if not, will then try to compile it as a full function.

For performance reasons, it is generally recommended to use the shared script whenever possible.

Example

```
#Return tag history using a custom aggregate function you wrote.

system.tag.queryTagHistory(paths=[‘MyTag’], rangeHours=1, aggregationModes=[‘shared.aggregates.
customFunction’])
```

Example

```
#Create a function on the fly to pass in as a custom aggregate.

wrapper = “””\
python:def wrapper(qval, interpolated, finished, blockContext, queryContext):
        return shared.aggregates.customFunction(qval, interpolated, finished, blockContext, queryContext)
“””
system.tag.queryTagHistory(paths=[‘MyTag’], rangeHours=1, aggregationModes=[wrapper])
```

Next...

Alarming

Alarming is a core feature of the Ignition platform. Alarms are conditions that are evaluated with respect to a specific numeric datapoint. Most commonly, alarms are configured on a tag or a Transaction Group item.

Any number of alarms can be attached to a datapoint. When the alarm condition becomes true, the alarm is considered to be "active". For example, on an analog tag you might add a high-level alarm that goes active when the tag's value is over 50.0. You could add another alarm onto the same tag for a high-high-level when the value is over 75.0. On a discrete tag, you might configure an alarm to be active when the value is non-zero. Each alarm has its own name, priority, and many other settings.

When the alarm condition becomes false after it has been true, the alarm is said to have "cleared". Each alarm has a numeric deadband and time delay deadbands that can affect when alarms go active and clear to prevent frequent oscillations of an alarm between active and clear when the value is hovering near the setpoint. At some point, an operator can acknowledge an alarm, confirming that they have seen the event.

The history of alarms is stored in any configured **Alarm Journals**. These will journal all state changes to each alarm, as well as acknowledgement, to an external SQL database of your choosing.

Information about configuring alarms can be found in Alarming Properties in the **Tag Intro** section.

In this section ...

Setting up Alarms

Alarming is a core feature of the Ignition platform. Alarms are conditions that are evaluated with respect to a specific numeric data point. Most commonly, alarms are configured on a tag or a Transaction Group item.

To set the general properties for alarms

1. Go to the **Config** section of the Gateway.
2. Choose **Alarming>General** from the menu on the left.
   The General Alarm Settings page is displayed.
3. Set the general alarm properties shown in the following table:

Alarm Evaluation	
Live Event Limit	Default is 5. The number of "live" events (active or unacknowledged) that can exist for a single alarm at a given time. When surpassed, older events will be acknowledged automatically by the system. This means as an alarm cycles on and off, Ignition will keep track of the last 5 times the alarm event happened until the user acknowledges them. This does not store history for those events.
Event Suppression	
Continuous Event Detection Window (min)	Default is 10. The amount of time to store events before shutdown to prevent new duplicate events from being created on startup. This setting prevents unacknowledged active events from being generated due to reboot. If set to 0, will not be used.
Notify Initial Events	Default is false. If false, active alarms caused by the "initial state" (that is, the first value checked after being created, or after the enabled state changes) won't be sent to the notification system. This means if you add an alarm to a tag, a notification won't be immediately sent when the new state is created.

In this section ...

Configure Alarm on a Tag

Alarming is a core feature of the Ignition platform. Alarms are conditions that are evaluated with respect to a specific numeric data point. Most commonly, alarms are configured on a Tag.

INDUCTIVE UNIVERSITY

Configure Alarm on a Tag

Watch the Video

To configure an alarm on a tag

1. Double-click on the tag to edit it.
   The Tag Editor window is displayed.
2. Click on **Alarming**, click the green plus sign **+** to add a **New Alarm** to the tag.
3. Enter a **Name** for the Alarm and set any other settings you wish.
4. In the **Alarm Mode Settings** choose the appropriate logic that determines if the alarm is active or not.
5. Click **Apply** or **OK** to commit the new alarm.
   Ignition will start monitoring the alarm immediately.

Alarm Attributes

All alarm states have a list of properties that you can edit or bind when creating or editing a tag. Here is the list of attributes paired with their scripting names. This is useful when using the system.tag.add() and system.tag.edit() functions.

If the attribute you want to reference is not listed, you can simply change the attribute on your tag, export it to XML, and view the reference to the alarm attribute.

Group	Property Name	Scripting Name	Description
Main			
	Priority	priority	The priority (or "severity") of this alarm. Diagnostic (0), Low (1), Medium (2), High (3), Critical (4)
	Timestamp Source	timestampSource	Dictates whether the timestamp used on the alarm comes from the system or the value. System (0), Value (1)
	Display Path	displayPath	An optional path (separated by "/") that will be used for display and browsing purposes.
	Ack Mode	ackMode	How acknowledgement works for the alarm: Auto (1) - Alarm is acknowledged when cleared. Manual (2) - Alarm acknowledged by user. Unused (0) - Alarm is always marked as 'acknowledged'.
	Notes	notes	Documentation about the alarm.
	Ack Notes Required	ackNotesReqd	Select this option to require operators to provide some explanation when the alarm is acknowledged. true or false
	Shelving Allowed	shelvingAllowed	true or false
Alarm Mode Settings			
	Mode	mode	The alarm mode. Equal (0), Not Equal (1), Above Setpoint (2), Below Setpoint (3), Between Setpoints (4), Outside Setpoints (5), Out of Range (6), Bad Quality (7), Any Change (8), Bit State (9), On Condition (10)
	Setpoint / Low Setpoint	setpointA	The value used to initiate an alarm. If the alarm mode calls for two setpoints, this is the low value.
	Inclusive / Low Inclusive	inclusiveA	If true, the Setpoint or Low Setpoint value is used inclusively for the condition to alarm.
	High Setpoint	setpointB	The high value used to initiate an alarm when the alarm mode calls for two setpoints.
	High Inclusive	inclusiveB	If true, the High Setpoint value is used inclusively for the condition to alarm.
	Any Change	anyChange	If true, will alarm on each value change given the alarm mode conditions are met.
	Bit Position	bitPosition	The position of the bit, starting at 0, that will be watched.
	On Zero	bitOnZero	If true, will alarm when the specified bit is not high, or "0".
Deadbands and Time Delays			
	Deadband	deadband	A numeric deadband of the alarm condition. The interpretation of this value depends on the deadband mode. Note that his deadband is separate from, and evaluated after, the tag's deadband.

	Deadband Mode	deadbandMode	Defines how the deadband value is used. If percent, the value (0-100) will be compared against the Eng. limits defined for the value. Absolute (0), Percent (1)
	Active Delay	timeOnDelaySeconds	The time, in seconds, before the alarm will be considered truly active after the active condition becomes true. Also known as a "rising edge time deadband".
	Clear Delay	timeOffDelaySeconds	The time, in seconds, before the alarm will be considered truly clear after the clear condition becomes true. Also know as a "falling edge time deadband".
Notifications			
	Active Pipeline	activePipeline	The pipeline that will be used to process active events generated by this alarm.
	Clear Pipeline	clearPipeline	The pipeline that will be used to process clear events generated by this alarm.
	Ack Pipeline	ackPipeline	The pipeline that will be notified when the alarm has been acknowledged.
Email Notification Settings			
	Custom Subject	CustomEmailSubject	If specified, will be used for the email subject. If blank, the subject defined in the notification block to be used.
	Custom Email	CustomEmailMessage	If specified, will be used for email message. If blank,. the message defined in the notification block will be used.
SMS Notification Settings			
	Custom Message	CustomSmsMessage	If specified, will be used for the SMS message. If blank, the message defined in the notification block will be used.
Associated Data			
	yourAssociatedData	yourAssociatedData	

Next...

- Multiple Alarms on a Tag

Multiple Alarms on a Tag

You can configure multiple alarms on one tag.

To configure multiple Alarms on one tag

1. Start by configuring an alarm and click **Apply** instead of **OK**.
2. Configure additional alarms by selecting the green plus + sign below the existing alarm.

The number of alarms that you can configure is virtually unlimited and will never exceed the bounds of reasonable design consideration.

Multiple Alarms on a Tag

Watch the Video

Next...

- Single Integer - Alarm on Each Bit

Single Integer - Alarm on Each Bit

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.

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.

To create the alarms on an integer tag value

1. Edit the tag that has an integer value.
2. Create an alarm by selecting the **Alarming** tab and clicking the green plus sign **+** to add an alarm.
3. Complete the appropriate properties, for example, enter the **Name** of the alarm.
4. From the **Mode** section drop-down, select **Bit State** and enter a value for the **Bit Position**.
5. Repeat the same process for the remainder of the bits.

Next...

- Dynamic Alarm Setpoints

Dynamic Alarm 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.

To configure dynamic setpoints on an alarm

1. Double-click on the tag to edit it.
   The **Tag Editor** window is displayed.
2. Select the **Alarming** tab and the alarm that you want to edit to associate a setpoint with.
3. Find the **Setpoint** property, click on the binding icon next to it, and select a Tag or Expression. This binds the Setpoint Property to a different tag or an expression.
4. Click **OK**.

Dynamic Alarm Setpoints

Watch the Video

Next...

- Configure Alarm in UDT

Configure Alarm in UDT

You can configure alarms inside of the UDT definitions. Every instance of the UDT will have the same alarm configurations, but they will be specific to that instance of UDT.

Configure Alarm in UDT

Watch the Video

To configure an alarm on a UDT

1. Edit the UDT definition by accessing it in the **Data Types** folder.
2. Select the tag and add an alarm inside the UDT definition. Its configurations are the same as the normal alarm configuration on a tag.
3. Click **OK**.

All instances of the UDT will have a similar alarm setup but each alarm will be unique to that UDT instance.

Next...

- UDT Alarm Dynamic Setpoints

UDT Alarm Dynamic Setpoints

You can configure alarms with dynamic set points inside of a UDT definition. It is similar to configuring a UDT alarm, however configuring a UDT alarm with dynamic set point requires additional tags to serve as these set points in all the deployed UDTs.

To configure dynamic set points inside a UDT

1. In the Tag Browser, go to the **Data Types** folder, and choose **Edit tags** to edit the UDT definition.
2. Create memory tags inside the UDT definition.
3. Select the tag and its alarm inside the UDT definition that has the alarm you want to associate the UDT with.
4. Bind the **Setpoint** property to the new UDT memory tags.
5. Click **OK**.
   All of the deployed UDT will have memory tags that serve as the alarm set points.

INDUCTIVE UNIVERSITY

UDT Alarm Dynamic Setpoints

Watch the Video

Next...

- Enabling Alarms Dynamically - Machine On

Enabling Alarms Dynamically - Machine On

Allowing a dynamic condition to determine if an alarm is enabled or disabled is possible inside the alarm's properties.

To configure the alarm's enabled property

1. Select a tag that has the alarm you want to configure.
2. Access the alarm properties by double-clicking on the tag and selecting the alarming properties tag.
3. Bind the enabled property to a boolean tag or an expression that evaluates a true or false condition.
4. Click **OK**.
   Now the alarm will only evaluate based upon the condition of the tag or expression.

Next...

- Enabling Alarms Dynamically - Time of Day

Enabling Alarms Dynamically - Time of Day

You can set alarms to automatically enable and disable at certain times of the day.

To enable the alarm dynamically

Alarms can be set to be enabled for part of the day and disabled for the remainder of the day.

1. From the **Tag Browser**, select the tag on which you want to configure the alarm.
2. Double-click on the tag, and in the Tag Editor window go to the Alarming section.
3. Bind the enabled property to an expression that evaluates a true or false condition based upon the current time.
   For example, the following expression would return true between the hours of 5 and 12.

   ```
   dateExtract(now(0), "hour" >= 5 && dateExtract(now(0), "hour" <= 12
   ```

4. Click **OK**.

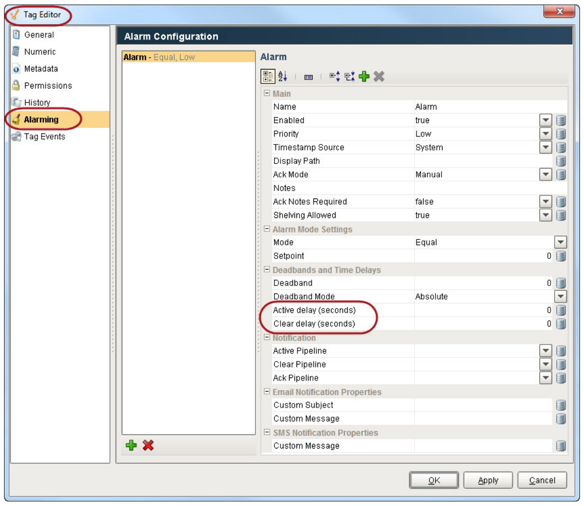
Next...

- Active and Clear Delays

Active and Clear Delays

To avoid nuisance alarms, the **Active delay** and **Clear delay** alarm settings guarantee that an alarm has been Active or Cleared for a specific amount of time before it actually becomes Active or Cleared.

You can access the Active and Clear delays configurations by editing the individual alarm on the tag using the **Tag Editor** window.



Main	
Name	Alarm
Enabled	true
Priority	Low
Timestamp Source	System
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
Deadband Mode	Absolute
Active delay (seconds)	0
Clear delay (seconds)	0
Notification	
Active Pipeline	
Clear Pipeline	
Ack Pipeline	
Email Notification Properties	
Custom Subject	
Custom Message	
SMS Notification Properties	
Custom Message	

Active delay

An Active delay will delay an alarm for as many seconds that is configured in the Active Delay property. You can bind this value to an expression or a tag for additional configuration.

Clear Delay

An Clear delay will delay an alarm for as many seconds that is configured in the Clear Delay. You can bind this value to an expression or a tag for additional configuration.

Next...

- Alarm Associated Data

Alarm Associated Data

You can extend the alarm properties by adding your own associated data to be associated with the alarm. A common scenario is to have an alarm on the Amps tag of a motor and associated data on a tag that is associated with the motor controls, therefore when the Amps tag becomes too high the current setting of the motor control will be ported with the alarm.

To create associated data

1. Right-click on the tag and select **Edit tag(s)**.
   The **Tag Editor** window is displayed.

2. Go to the **Alarming** section, select or create an alarm, and click the green plus sign **+** to add new associated data for the alarm.

3. Rename the **Associated Data**, bind it to a value, and click **OK**.
   When the alarm becomes active, it can grab the value of the associated data and make it available with the alarm diagnostics and auditing.

INDUCTIVE UNIVERSITY

Alarm Associated Data

Watch the Video

Next...

- Alarm Grouping

Alarm Grouping

Grouping alarms in Ignition is an important concept so you can view alarms for a particular area of your plant or for a specific set of alarms. A common design practice is to associate alarm groupings on the associated data of the alarm.

To create alarm grouping

1. Create an alarm.
2. Create associated data on the alarm.
   The associated data should be a static value that describes the group.
3. Click **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.

INDUCTIVE UNIVERSITY

Alarm Grouping

Watch the Video

Next...

Alarm Properties

Alarm Properties

An alarm event is made up of many pieces of information. The state, the value, the time, all of the configuration data for the alarm, and more are collectively known as the properties of the event. A "property" is generally a name, potentially along with a default value. There are many properties that an alarm may have, and if the property isn't present, the default value is usually assumed.

How Properties are Used

Properties are used in a number of ways, they:

- Define how an alarm behaves.
- Are referenced in messages, such as the body of an email, or sms message.
- Are referenced in Expressions, such as in the binding of a different property, or in an Expression block.
- Are created and used in pipelines as temporary variables, such as to make a counter.
- ... and various other places.

While properties are always accessed the same (for example by using the `{propertyName}` syntax in a message or Expression, or the `getProperty()` Expression function), there are some subtle variations on where the property value comes from, and what its lifecycle will be. To understand this better, consider that an Alarm Event has the following structure:

Alarm Event
   Active Event Properties
   Clear Event Properties
   Acknowledged Event Properties
   Runtime Properties

The first three collections of properties get created when the described type of event occurs.
Consequentially, it's possible for the same property to exist multiple times in an Alarm Event. For example, a bound configuration property or associated data is captured on both the active and clear events. When you reference a property, the alarm event will provide you with the most recent value. So, if you have a bound property call "MyData", and you reference it when the alarm is active, you may get a different result than when the reference is executed later and the alarm has cleared. The individual values are stored separately in the alarm journal, however.

Configuration Properties	
Name	The name of the Alarm.
Enabled	Set to False to turn off the alarm state and all associated actions.
Priority	Used for sorting/filtering. 0=Diagnostic, 1=Low, 2=Medium, 3=High, 4=Critical
Display Path	The unique path of the Alarm state.
Active Pipeline	Which pipeline (if any) to use when the alarm goes into the Active State.
ClearPipeline	Which pipeline (if any) to use when the alarm goes into the Clear State.
TimeOnDelaySeconds	The amount of consecutive seconds that the alarm state must be True before the Tag enters this alarm state.
TimeOffDelaySeconds	The amount of consecutive seconds that the alarm state must be False before the Tag exits this alarm state.
Notes	Free-form notes for the alarm state.

As described in the previous section, Runtime Properties are different in that they only exist while the alarm event is in memory (still "live", that is, not cleared, or not acknowledged). They do not get stored in the alarm journal. In addition to properties created through the Set Property block, the system also has a number of defined Runtime Properties that it may use for various purposes. Though these are used internally, they are technically still regular properties, and can be accessed and modified through the normal means.

Runtime Properties (generated for the event)	
IsInitialEvent	Set to "true" when the event is caused by the initial state of the alarm.
SystemAck	Set to "true" when the alarm has been acknowledged by the system, due to an overflow of the "live event queue". Live events are alarm events that are active or not acknowledged, and are limited for each alarm by the general alarm settings.
ShelfExpiration	When the shelf will expire for this event.
IsShelved	Is the alarm currently shelved?
EventCanceled	If set, the event will drop out as soon as possible from the pipelines
EventId	The unique id (uuid) of this alarm event. Each event gets a completely unique id.

Source	The qualified path to the item that generated this event.
DisplayPathOrSource	Gets the display path if defined, otherwise returns the source.
State	The current overall state of the alarm: ClearUnacked (0), ClearAcked (1), ActiveUnacked (2), ActiveAcked (3)
EventState	The transitional state that caused the current event: Active (0), Clear (1), Ack (2).
EventValue	The value associated with the current event.
AckUser	The user who acknowledged this event.
IsAcked	"True" if the event has been acknowledged.
IsActive	"True" if the event is still active.
IsClear	"True" if the event is not still active.
ActiveTime, ClearTime, AckTime	The timestamp for each event.
PipelineTransitionCount	How many transitions the event has made inside of the pipelines.

Associcated Data

You can extent alarm properties by creating your own associated data. Associated data properties can be configured on any alarms that are already configured in a project.

To learn more, refer to Alarm Associated Data.

Next ...

Alarm Status

In the Designer, you can use the **Alarm Status Table** component in the **Alarming** section of the **Component Palette** to display the current status of the alarms. You can set it up to show the active, unacknowledged, cleared, and acknowledged alarm statuses.

In this section ...

Display Status of Single Alarm

You can display the status of a single alarm on a screen. Filters can be set to show active/inactive, cleared/uncleard, or acknowledged/unacknowledged alarms.

Tags have multiple properties related to the current state of configured alarms on the tag. It's possible to bind a property on a component to one of these tag properties. This way components can respond to the state of an alarm on a tag

Display Status of Single Alarm

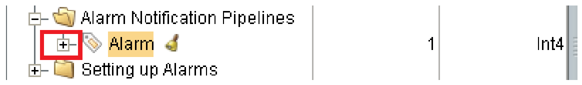
Watch the Video

Example

1) In the Designer, create a new Text Field component on an empty window

2) Create a tag named 'Alarm' with a datatype of Int4, and configure with an alarm to become active on a value of 1

3) Once the tag has been created, click on the '+' icon next to the tag in the tag browser



```
Alarm Notification Pipelines
    Alarm                     1        Int4
Setting up Alarms
```

This will expand the tag and show all of the tag properties, including the AlertActive property

4) With the Text Field component selected, drag and drop the AlertActive property fromt the tag to the Visible property on the Text Field component. This will create a binding directly to that tag property.

5) You'll find that the component will appear when an alarm is active, and become hidden once all alarms on the tag are cleared.

Next...

- Alarm Status Component

Alarm Status Component

The **Alarm Status Table** is a built-in component in Ignition that displays the current status of all alarms in the system into one view. The Alarm Status Table is highly customizable and can be configured to show active, unacknowledged, cleared, and acknowledged alarms.

Additional information on the Alarm Status Table component can be found on the Alarm Status Table page in the appendix.

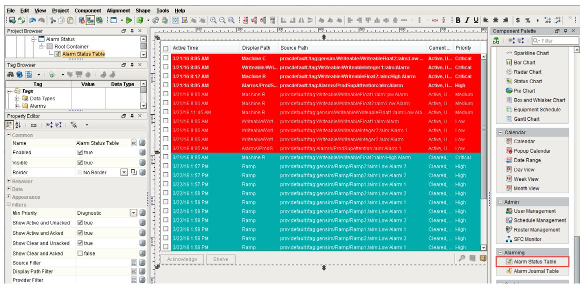
INDUCTIVE UNIVERSITY

Alarm Status - Component

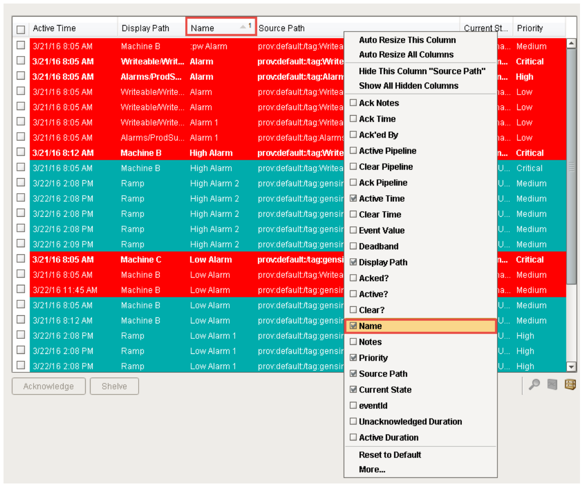
Watch the Video

The Alarm Status Table is very easy to setup.

1. In the **Component Palette in Designer**, under Alarming, drag the **Alarm Status Table component** on to your window. You will notice that the table is already configured to show the current status of all the alarms in your system. By default, the Alarm Status Table shows all the Active and Unacknowledged or Acknowledged alarms, and Cleared and Unacknowledged alarms. It also displays the time the alarm went Active as well as the current State and Priority of the alarm.
   Note: You must have alarms set up on at your tags for them to show up in the table.

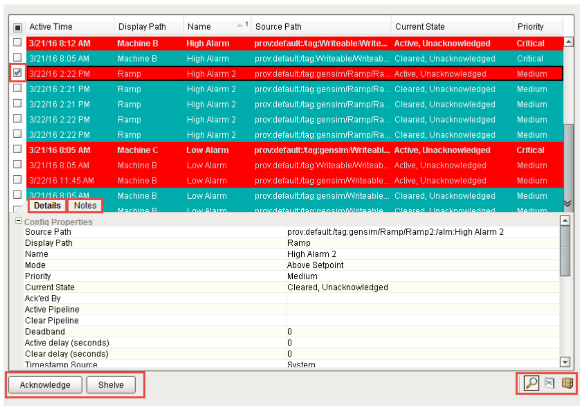


2. In **Preview Mode**, right click on the header to show or hide information on the Alarm Status Table. If you want to show the name of the alarm, you can add '**Name**' to the table by checking the box. You can also reorganize the columns to display how you want the columns to look by dragging the column header anywhere on the table header.



Active Time	Display Path	Name	Source Path	Current St...	Priority
3/21/16 8:05 AM	Machine B	:pw Alarm	prov:default:/tag:Writea	a...	Medium
3/21/16 8:05 AM	Writeable/Writ...	Alarm	prov:default:/tag:Write	n...	Critical
3/21/16 8:05 AM	Alarms/ProdS...	Alarm	prov:default:/tag:Alarm	n...	High
3/21/16 8:05 AM	Writeable/Write...	Alarm	prov:default:/tag:Writea	a...	Low
3/21/16 8:05 AM	Writeable/Write...	Alarm	prov:default:/tag:Writea	a...	Low
3/21/16 8:05 AM	Writeable/Write...	Alarm 1	prov:default:/tag:Writea	a...	Low
3/21/16 8:05 AM	Alarms/ProdSu...	Alarm 1	prov:default:/tag:Alarms	a...	Low
3/21/16 8:12 AM	Machine B	High Alarm	prov:default:/tag:Write	n...	Critical
3/21/16 8:05 AM	Machine B	High Alarm	prov:default:/tag:Writea	U...	Critical
3/22/16 2:08 PM	Ramp	High Alarm 2	prov:default:/tag:gensir	U...	Medium
3/22/16 2:08 PM	Ramp	High Alarm 2	prov:default:/tag:gensir	U...	Medium
3/22/16 2:08 PM	Ramp	High Alarm 2	prov:default:/tag:gensir	U...	Medium
3/22/16 2:08 PM	Ramp	High Alarm 2	prov:default:/tag:gensir	U...	Medium
3/22/16 2:09 PM	Ramp	High Alarm 2	prov:default:/tag:gensir	U...	Medium
3/21/16 8:05 AM	Machine C	Low Alarm	prov:default:/tag:gensi	n...	Critical
3/21/16 8:05 AM	Machine B	Low Alarm	prov:default:/tag:Writea	a...	Medium
3/22/16 11:45 AM	Machine B	Low Alarm	prov:default:/tag:gensir	a...	Medium
3/21/16 8:05 AM	Machine B	Low Alarm	prov:default:/tag:gensir	U...	Medium
3/21/16 8:12 AM	Machine B	Low Alarm	prov:default:/tag:gensir	U...	Medium
3/22/16 2:08 PM	Ramp	Low Alarm 1	prov:default:/tag:gensir	U...	High
3/22/16 2:08 PM	Ramp	Low Alarm 1	prov:default:/tag:gensir	U...	High
3/22/16 2:08 PM	Ramp	Low Alarm 1	prov:default:/tag:gensir	U...	High

3. You can click on an individual alarm and use the buttons at the bottom of the table to Acknowledge or Shelve an alarm. You can also use the icons to view the alarm Details, Notes, and History.



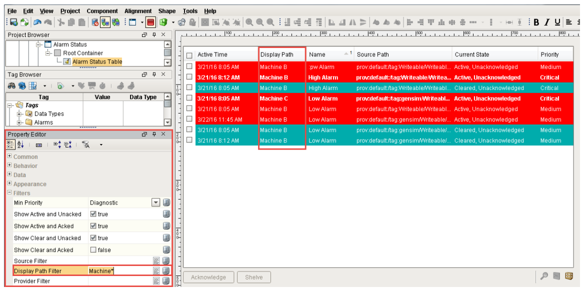
	Active Time	Display Path	Name	Source Path	Current State	Priority
☐	3/21/16 8:12 AM	Machine B	High Alarm	prov:default:/tag:Writeable/Write...	Active, Unacknowledged	Critical
☐	3/21/16 8:05 AM	Machine B	High Alarm	prov:default:/tag:Writeable/Writeab...	Cleared, Unacknowledged	Critical
✓	3/22/16 2:22 PM	Ramp	High Alarm 2	prov:default:/tag:gensim/Ramp/Ra...	Active, Unacknowledged	Medium
☐	3/22/16 2:21 PM	Ramp	High Alarm 2	prov:default:/tag:gensim/Ramp/Ra...	Cleared, Unacknowledged	Medium
☐	3/22/16 2:21 PM	Ramp	High Alarm 2	prov:default:/tag:gensim/Ramp/Ra...	Cleared, Unacknowledged	Medium
☐	3/22/16 2:22 PM	Ramp	High Alarm 2	prov:default:/tag:gensim/Ramp/Ra...	Cleared, Unacknowledged	Medium
☐	3/22/16 2:22 PM	Ramp	High Alarm 2	prov:default:/tag:gensim/Ramp/Ra...	Cleared, Unacknowledged	Medium
☐	3/21/16 8:05 AM	Machine C	Low Alarm	prov:default:/tag:gensim/Writeabl...	Active, Unacknowledged	Critical
☐	3/21/16 8:05 AM	Machine B	Low Alarm	prov:default:/tag:Writeable/Writeab...	Active, Unacknowledged	Medium
☐	3/22/16 11:45 AM	Machine B	Low Alarm	prov:default:/tag:gensim/Writeable...	Active, Unacknowledged	Medium
☐	3/21/16 8:05 AM	Machine B	Low Alarm	prov:default:/tag:gensim/Writeable...	Cleared, Unacknowledged	Medium

Config Properties

Source Path	prov:default:/tag:gensim/Ramp/Ramp2:/alm:High Alarm 2
Display Path	Ramp
Name	High Alarm 2
Mode	Above Setpoint
Priority	Medium
Current State	Cleared, Unacknowledged
Ack'ed By	
Active Pipeline	
Clear Pipeline	
Deadband	0
Active delay (seconds)	0
Clear delay (seconds)	0
Timestamp Source	System

4. In the **Property Editor**, there are a set of alarm properties for filtering alarms. Using the **Source Filter** and the **Display Path Filter** properties, you can filter out cleared alarms, filter for alarms in a particular Tag folder, and filter for alarms matching a certain criteria. This example uses the Display Path to filter for all 'Machines.'

   Filtering on alarms is discussed in detail in sections, Alarm Status General Filtering and Alarm Status Filter on Associated Data.



Min Priority	Diagnostic
Show Active and Unacked	✓ true
Show Active and Acked	✓ true
Show Clear and Unacked	✓ true
Show Clear and Acked	☐ false
Source Filter	
Display Path Filter	Machine*
Provider Filter	

	Active Time	Display Path	Name	Source Path	Current State	Priority
☐	3/21/16 8:05 AM	Machine B	:pw Alarm	prov:default:/tag:Writeable/Writeabl...	Active, Unacknowledged	Medium
☐	3/21/16 8:12 AM	Machine B	High Alarm	prov:default:/tag:Writeable/Writea...	Active, Unacknowledged	Critical
☐	3/21/16 8:05 AM	Machine B	High Alarm	prov:default:/tag:Writeable/Writeabl...	Cleared, Unacknowledged	Critical
☐	3/21/16 8:05 AM	Machine C	Low Alarm	prov:default:/tag:gensim/Writeabl...	Active, Unacknowledged	Critical
☐	3/21/16 8:05 AM	Machine B	Low Alarm	prov:default:/tag:Writeable/Writeabl...	Active, Unacknowledged	Medium
☐	3/22/16 11:45 AM	Machine B	Low Alarm	prov:default:/tag:gensim/Writeable/...	Active, Unacknowledged	Medium
☐	3/21/16 8:05 AM	Machine B	Low Alarm	prov:default:/tag:gensim/Writeable/...	Cleared, Unacknowledged	Medium
☐	3/21/16 8:12 AM	Machine B	Low Alarm	prov:default:/tag:gensim/Writeable/...	Cleared, Unacknowledged	Medium

As you can see, the Alarm Status Table contains a host of features already built in to the component which you can take advantage of to give you the current status of alarms in your system.

Next...

- Alarm Status - General Filtering
- Alarm Status

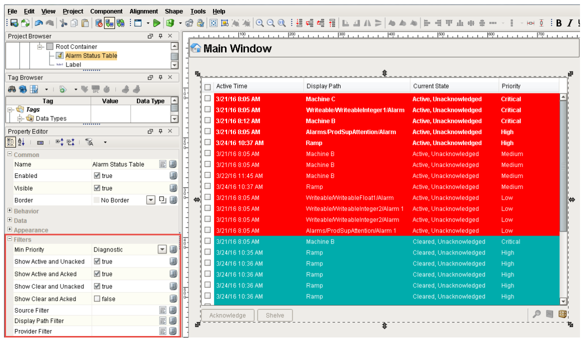
Alarm Status - General Filtering

INDUCTIVE UNIVERSITY

Alarm Status - General Filtering

Watch the Video

There is a lot of filtering you can do on the Alarm Status Table that comes with Ignition. Once you drag the Alarm Status Table component on to a window, by default it shows you all the alarms that are currently 'active and unacknowledged,' 'active and acknowledged,' and 'clear and unacknowledged' with any priority.

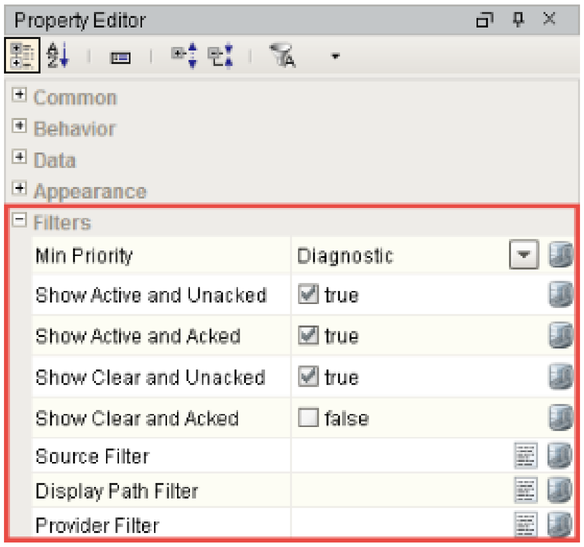


Main Window

Active Time	Display Path	Current State	Priority
3/21/16 8:05 AM	Machine C	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Writeable/WriteableInteger1/Alarm	Active, Unacknowledged	Critical
3/21/16 8:12 AM	Machine B	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm	Active, Unacknowledged	High
3/24/16 10:37 AM	Ramp	Active, Unacknowledged	High
3/21/16 8:05 AM	Machine B	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Machine B	Active, Unacknowledged	Medium
3/22/16 11:45 AM	Machine B	Active, Unacknowledged	Medium
3/24/16 10:37 AM	Ramp	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Writeable/WriteableFloat1/Alarm	Active, Unacknowledged	Low
3/21/16 8:05 AM	Writeable/WriteableInteger2/Alarm 1	Active, Unacknowledged	Low
3/21/16 8:05 AM	Writeable/WriteableInteger2/Alarm	Active, Unacknowledged	Low
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm 1	Active, Unacknowledged	Low
3/21/16 8:05 AM	Machine B	Cleared, Unacknowledged	Critical
3/24/16 10:35 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 10:36 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 10:36 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 10:36 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 10:36 AM	Ramp	Cleared, Unacknowledged	High

You can filter the alarm list in the table to be a shorter list using the filter properties in the Property Editor instead of scrolling through every single alarm in your system,

In the **Property Editor**, there is a group called **Filters** that you can configure to focus on only those alarms you want to see. You can filter on **Priority, State, Source or Tag Path,** and **Display Path**.

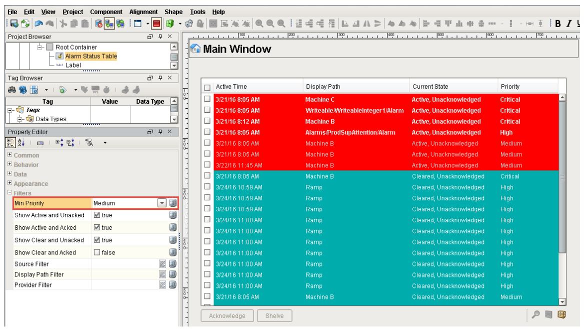


Property Editor

Common
Behavior
Data
Appearance
Filters

Min Priority	Diagnostic
Show Active and Unacked	✓ true
Show Active and Acked	✓ true
Show Clear and Unacked	✓ true
Show Clear and Acked	☐ false
Source Filter	
Display Path Filter	
Provider Filter	

This example shows how you can filter alarms based on **Priority**. From the **Min Priority** field, select **'Me dium'** from the dropdown list. You will then see all the Medium priority and higher alarms displayed which creates a smaller list of alarms for an operator to view.

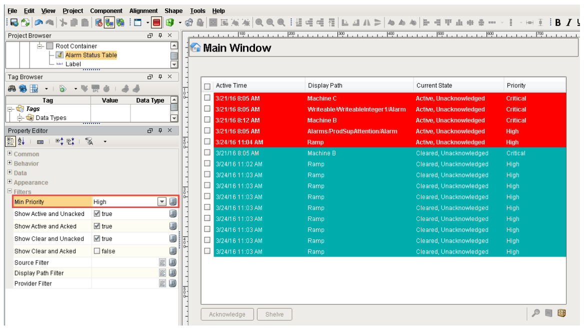


Min Priority	Medium
Show Active and Unacked	✓ true
Show Active and Acked	✓ true
Show Clear and Unacked	✓ true
Show Clear and Acked	☐ false

Main Window

Active Time	Display Path	Current State	Priority
3/21/16 8:05 AM	Machine C	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Writeable/WriteableInteger1/Alarm	Active, Unacknowledged	Critical
3/21/16 8:12 AM	Machine B	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm	Active, Unacknowledged	High
3/21/16 8:05 AM	Machine B	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Machine B	Active, Unacknowledged	Medium
3/22/16 11:45 AM	Machine B	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Machine B	Cleared, Unacknowledged	Critical
3/24/16 10:59 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 10:59 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 10:59 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:00 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:00 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:00 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:00 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:00 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:00 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:00 AM	Ramp	Cleared, Unacknowledged	High
3/21/16 8:05 AM	Machine B	Cleared, Unacknowledged	Medium

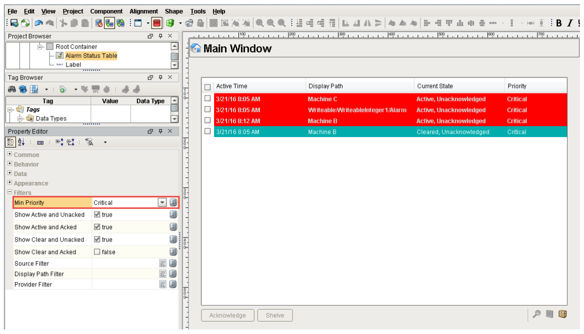
In this next example, select **'High'** from the dropdown list in the **Min Priority** field. Now, you will see all High and Critical priority alarms in the system which gives the operator an even smaller list of alarms to view.



Min Priority: High
Show Active and Unacked: ☑ true
Show Active and Acked: ☑ true
Show Clear and Unacked: ☑ true
Show Clear and Acked: ☐ false
Source Filter
Display Path Filter
Provider Filter

Active Time	Display Path	Current State	Priority
3/21/16 8:05 AM	Machine C	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Writeable/WriteableInteger1/Alarm	Active, Unacknowledged	Critical
3/21/16 8:12 AM	Machine B	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm	Active, Unacknowledged	High
3/24/16 11:04 AM	Ramp	Active, Unacknowledged	High
3/21/16 8:05 AM	Machine B	Cleared, Unacknowledged	Critical
3/24/16 11:02 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:03 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:03 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:03 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:03 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:03 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:03 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:03 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:03 AM	Ramp	Cleared, Unacknowledged	High

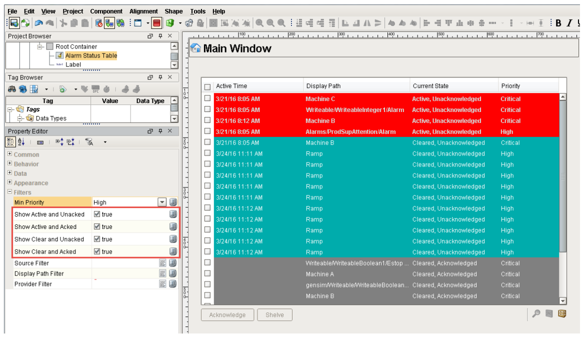
Lastly, select **'Critical'** from the dropdown list.   Notice how the alarm list keeps getting shorter by using the Priority filter.



Min Priority: Critical
Show Active and Unacked: ☑ true
Show Active and Acked: ☑ true
Show Clear and Unacked: ☑ true
Show Clear and Acked: ☐ false
Source Filter
Display Path Filter
Provider Filter

Active Time	Display Path	Current State	Priority
3/21/16 8:05 AM	Machine C	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Writeable/WriteableInteger1/Alarm	Active, Unacknowledged	Critical
3/21/16 8:12 AM	Machine B	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Machine B	Cleared, Unacknowledged	Critical

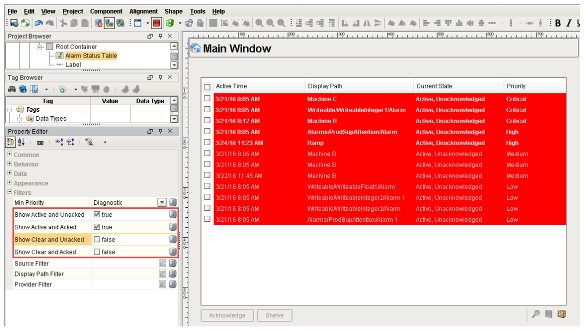
The second set of filters is based on **State**. There are 4 states an alarm can be in; Active and Unacknowledged, Active and Acknowledged, Clear and Unacknowledged, and Clear and Acknowledged. Each alarm has a different color representing one of the four States.



Min Priority: High
Show Active and Unacked: ☑ true
Show Active and Acked: ☑ true
Show Clear and Unacked: ☑ true
Show Clear and Acked: ☑ true
Source Filter
Display Path Filter
Provider Filter

Active Time	Display Path	Current State	Priority
3/21/16 8:05 AM	Machine C	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Writeable/WriteableInteger1/Alarm	Active, Unacknowledged	Critical
3/21/16 8:12 AM	Machine B	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm	Active, Unacknowledged	High
3/21/16 8:05 AM	Machine B	Cleared, Unacknowledged	Critical
3/24/16 11:11 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:11 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:11 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:11 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:11 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:12 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:12 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:12 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:12 AM	Ramp	Cleared, Unacknowledged	High
3/24/16 11:12 AM	Ramp	Cleared, Unacknowledged	High
	Writeable/WriteableBoolean1/Estop ...	Cleared, Acknowledged	Critical
	Machine A	Cleared, Acknowledged	Critical
	gensim/Writeable/WriteableBoolean...	Cleared, Acknowledged	Critical
	Machine B	Cleared, Acknowledged	Critical

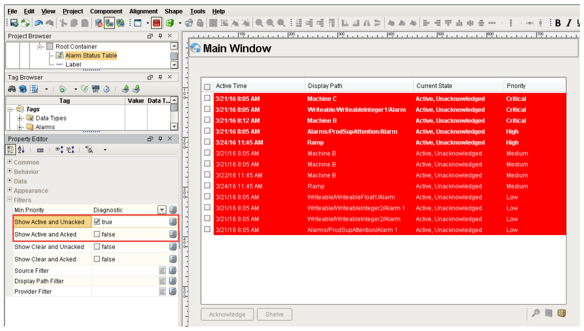
Let's make our alarm list smaller by **unchecking** both the **Clear** filters. This will eliminate all the Clear alarms from the Alarm Status Table and only show the Active alarms whether they are Acknowledged or Unacknowledged.



Min Priority: Diagnostic
Show Active and Unacked: ☑ true
Show Active and Acked: ☑ true
Show Clear and Unacked: ☐ false
Show Clear and Acked: ☐ false
Source Filter
Display Path Filter
Provider Filter

Active Time	Display Path	Current State	Priority
3/21/16 8:05 AM	Machine C	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Writeable/WriteableInteger1/Alarm	Active, Unacknowledged	Critical
3/21/16 8:12 AM	Machine B	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm	Active, Unacknowledged	High
3/24/16 11:23 AM	Ramp	Active, Unacknowledged	High
3/21/16 8:05 AM	Machine B	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Machine B	Active, Unacknowledged	Medium
3/22/16 11:45 AM	Machine B	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Writeable/WriteableFloat1/Alarm	Active, Unacknowledged	Low
3/21/16 8:05 AM	Writeable/WriteableInteger2/Alarm 1	Active, Unacknowledged	Low
3/21/16 8:05 AM	Writeable/WriteableInteger2/Alarm	Active, Unacknowledged	Low
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm 1	Active, Unacknowledged	Low

If you uncheck **Active and Acknowledged**, this will only show the alarms in the Alarm Status Table that are currently Active, and are not Acknowledged. A good business practice is to show only the Active alarms that are either Unacknowledged or Acknowledged. It also reduces the number of alarms an operator has to scroll through.

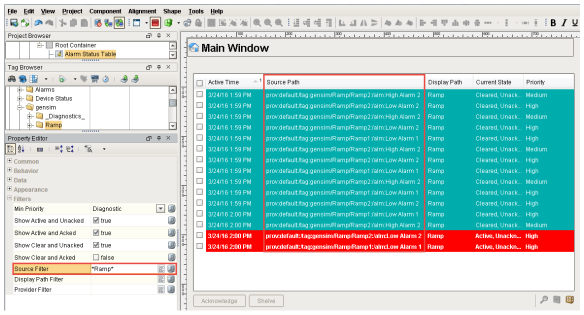


Property	Value
Min Priority	Diagnostic
Show Active and Unacked	✓ true
Show Active and Acked	false
Show Clear and Unacked	false
Show Clear and Acked	false
Source Filter	
Display Path Filter	
Provider Filter	

Active Time	Display Path	Current State	Priority
3/21/16 8:05 AM	Machine C	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Writeable/WriteableInteger1/Alarm	Active, Unacknowledged	Critical
3/21/16 8:12 AM	Machine B	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm	Active, Unacknowledged	High
3/24/16 11:45 AM	Ramp	Active, Unacknowledged	High
3/21/16 8:05 AM	Machine B	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Machine B	Active, Unacknowledged	Medium
3/22/16 11:45 AM	Machine B	Active, Unacknowledged	Medium
3/24/16 11:45 AM	Ramp	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Writeable/WriteableFloat1/Alarm	Active, Unacknowledged	Low
3/21/16 8:05 AM	Writeable/WriteableInteger2/Alarm 1	Active, Unacknowledged	Low
3/21/16 8:05 AM	Writeable/WriteableInteger2/Alarm	Active, Unacknowledged	Low
3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm 1	Active, Unacknowledged	Low

Ignition also has the capability to filter alarms based on the Source or Tag Path and the Display Path.

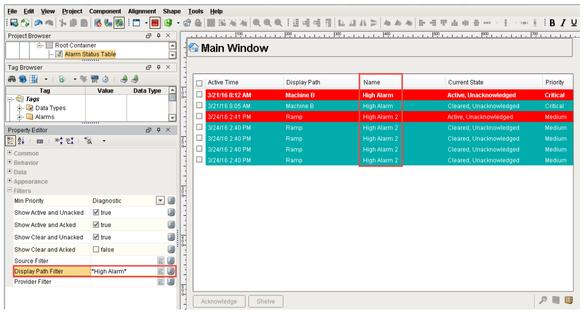
The **Source** is the actual **Tag path** which means you can also use Tag folders in Ignition to filter for specific alarms. For example, you may want to filter for all Ramp alarms in the 'Ramp' folder. **Enter *Ramp*** (using * as wildcards) to look for all alarms with the Ramp Tag Path. You can see in this example, all the alarms have 'Ramp' in the Source Path.



Property	Value
Min Priority	Diagnostic
Show Active and Unacked	✓ true
Show Active and Acked	✓ true
Show Clear and Unacked	✓ true
Show Clear and Acked	false
Source Filter	*Ramp*
Display Path Filter	
Provider Filter	

Active Time	Source Path	Display Path	Current State	Priority
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:High Alarm 2	Ramp	Cleared, Unack...	Medium
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:Low Alarm 2	Ramp	Cleared, Unack...	High
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:High Alarm 2	Ramp	Cleared, Unack...	Medium
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:Low Alarm 2	Ramp	Cleared, Unack...	High
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp1:/alm:Low Alarm 1	Ramp	Cleared, Unack...	High
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:High Alarm 2	Ramp	Cleared, Unack...	Medium
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:Low Alarm 2	Ramp	Cleared, Unack...	High
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp1:/alm:Low Alarm 1	Ramp	Cleared, Unack...	High
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:High Alarm 2	Ramp	Cleared, Unack...	Medium
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp1:/alm:Low Alarm 1	Ramp	Cleared, Unack...	High
3/24/16 1:59 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:Low Alarm 2	Ramp	Cleared, Unack...	High
3/24/16 2:00 PM	prov:default:/tag:gensim/Ramp/Ramp1:/alm:Low Alarm 1	Ramp	Cleared, Unack...	High
3/24/16 2:00 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:High Alarm 2	Ramp	Cleared, Unack...	Medium
3/24/16 2:00 PM	prov:default:/tag:gensim/Ramp/Ramp2:/alm:Low Alarm 2	Ramp	Active, Unackn...	High
3/24/16 2:00 PM	prov:default:/tag:gensim/Ramp/Ramp1:/alm:Low Alarm 1	Ramp	Active, Unackn...	High

Essentially, you can do the same thing using the **Display Path**. In this example, 'High Alarm' is the name that was setup in the Alarm Configuration, and the Display Path is set to the name you want the operator to see. So for example, the Alarm Status Table shows all the High Alarms.



Property	Value
Min Priority	Diagnostic
Show Active and Unacked	✓ true
Show Active and Acked	✓ true
Show Clear and Unacked	✓ true
Show Clear and Acked	false
Source Filter	
Display Path Filter	*High Alarm*
Provider Filter	

Active Time	Display Path	Name	Current State	Priority
3/21/16 8:12 AM	Machine B	High Alarm	Active, Unacknowledged	Critical
3/21/16 8:05 AM	Machine B	High Alarm	Cleared, Unacknowledged	Critical
3/24/16 2:41 PM	Ramp	High Alarm 2	Active, Unacknowledged	Medium
3/24/16 2:40 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
3/24/16 2:40 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
3/24/16 2:40 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
3/24/16 2:40 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium

So, it's really easy to do some basic filtering using the alarm filter properties using the Alarm Status Table component.

Next...

- Alarm Status - Filter on Associated Data

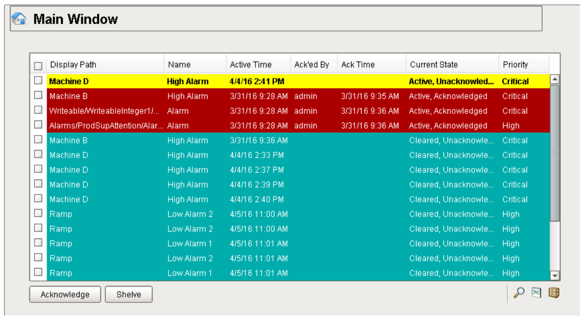
Alarm Status - Filter on Associated Data

INDUCTIVE UNIVERSITY

Alarm Status - Filter on Associated Data

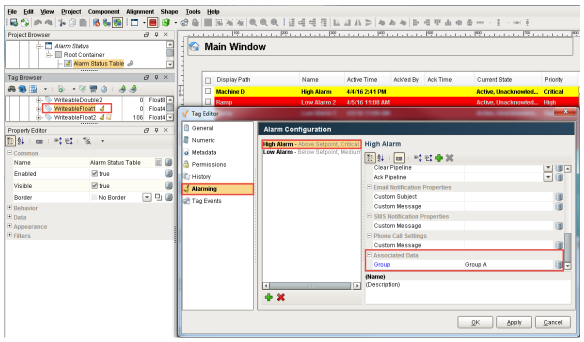
Watch the Video

Another way to filter alarms in the **Alarm Status Table** is using associated data that was added to an alarm. You can easily and quickly search, filter, and display on specific alarms based on associated data configured in an alarm.



Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
Machine D	High Alarm	4/4/16 2:41 PM			Active, Unacknowled...	Critical
Machine B	High Alarm	3/31/16 9:28 AM	admin	3/31/16 9:35 AM	Active, Acknowledged	Critical
Writeable/WriteableInteger1/...	Alarm	3/31/16 9:28 AM	admin	3/31/16 9:36 AM	Active, Acknowledged	Critical
Alarms/ProdSupAttention/Alar...	Alarm	3/31/16 9:28 AM	admin	3/31/16 9:36 AM	Active, Acknowledged	High
Machine B	High Alarm	3/31/16 9:36 AM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	4/4/16 2:33 PM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	4/4/16 2:37 PM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	4/4/16 2:39 PM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	4/4/16 2:40 PM			Cleared, Unacknowle...	Critical
Ramp	Low Alarm 2	4/5/16 11:00 AM			Cleared, Unacknowle...	High
Ramp	Low Alarm 2	4/5/16 11:00 AM			Cleared, Unacknowle...	High
Ramp	Low Alarm 1	4/5/16 11:01 AM			Cleared, Unacknowle...	High
Ramp	Low Alarm 2	4/5/16 11:01 AM			Cleared, Unacknowle...	High
Ramp	Low Alarm 1	4/5/16 11:01 AM			Cleared, Unacknowle...	High

In this example, **'WriteableFloat1'** was used and a **'High Alarm'** configured. Associated data was added to High Alarm called **'Group"** with a static value called **'Group A.'**



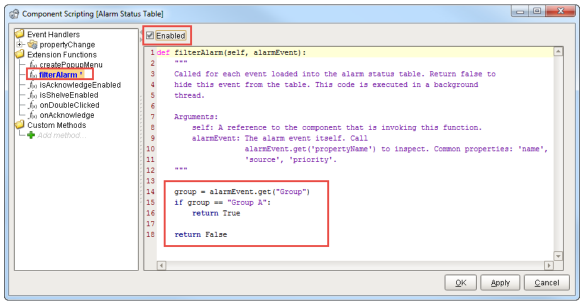
Let's create a script to filter for all the alarms in the Alarm Status Table that have the associated data named **'Group A.'**   Scripting allows you to use the **'filterAlarm' Extension Function** specifically for filtering on associated data.

1. Right click on the Alarm Status Table component, and scroll down and select **Scripting**.
2. In the Component Scripting window, under **Extension Functions**, select **'filterAlarm'** for filtering on associated data. You can filter on anything you want here, but in this example, filter on **'Group A.'**
3. Click **Enabled.**
4. Enter the code below into the **'filterAlarm'** script.

Extension Function - filterAlarm for 'Group A'

```
group = alarmEvent.get("Group")
if group == "Group A":
        return True
return False
```

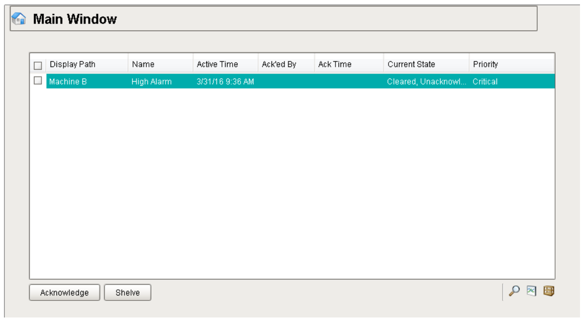
Once your code is entered, it will look like this...



```
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 == "Group A":
        return True

    return False
```

5. Click **OK**.

This script only displays alarms matching the associated data for **'Group A.'** For every alarm matching **‘Group A,’** it will return **‘True’** and show the alarm in the Alarm Status Table. If the associated data does not match **‘Group A,’** it will return **‘False,’** and the alarm will not show up in the Alarm Status Table.



☐	Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
☐	Machine B	High Alarm	3/31/16 9:36 AM			Cleared, Unacknowl...	Critical

Scripting gives you the ability to filter on associated data, but also gives you the freedom to filter on anything that you want.

Important Note - If an Error Filtering Script Occurs

If you have an error in this filtering script, it will return true for every alarm instance (and show all alarms) instead of displaying many errors to your users. You can find more information about the error in the Client Console.

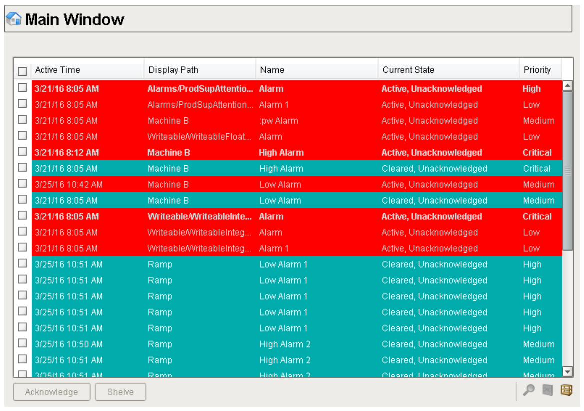
Next...

- Alarm Status - Row Styles

Alarm Status - Row Styles

You can customize the style of alarms in different states on the **Alarm Status Table** component. By default, the Alarm Status Table comes with a specific set of colors associated with each of the alarm states. You can change each of these colors for each of the states by going to the Alarms Row Styles Customizer that the component provides.

By default, the Active and Unacknowledged alarms are red and blinking, Cleared Unacknowledged alarms are teal, etc.



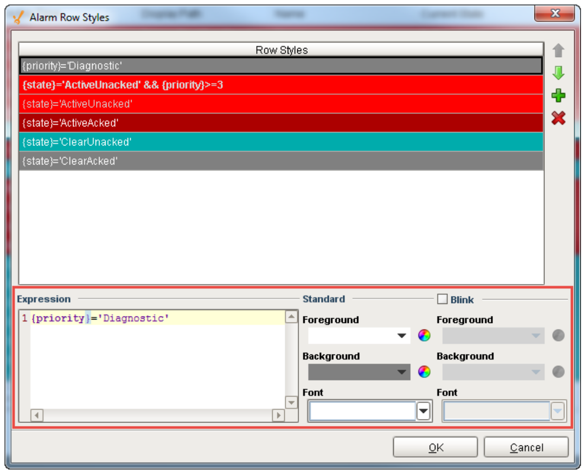
Main Window

Active Time	Display Path	Name	Current State	Priority
3/21/16 8:05 AM	**Alarms/ProdSupAttentio...**	**Alarm**	**Active, Unacknowledged**	**High**
3/21/16 8:05 AM	Alarms/ProdSupAttention...	Alarm 1	Active, Unacknowledged	Low
3/21/16 8:05 AM	Machine B	:pw Alarm	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Writeable/WriteableFloat...	Alarm	Active, Unacknowledged	Low
3/21/16 8:12 AM	**Machine B**	**High Alarm**	**Active, Unacknowledged**	**Critical**
3/21/16 8:05 AM	Machine B	High Alarm	Cleared, Unacknowledged	Critical
3/25/16 10:42 AM	Machine B	Low Alarm	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Machine B	Low Alarm	Cleared, Unacknowledged	Medium
3/21/16 8:05 AM	**Writeable/WriteableInte...**	**Alarm**	**Active, Unacknowledged**	**Critical**
3/21/16 8:05 AM	Writeable/WriteableInteg...	Alarm	Active, Unacknowledged	Low
3/21/16 8:05 AM	Writeable/WriteableInteg...	Alarm 1	Active, Unacknowledged	Low
3/25/16 10:51 AM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 10:51 AM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 10:51 AM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 10:51 AM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 10:51 AM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 10:50 AM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
3/25/16 10:51 AM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
3/25/16 10:51 AM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium

Right click on the **Alarm Status Table** component, go down to **Customizers** and select **Alarm Row Styles**.   Here is where you can modify an existing style, add more styles or delete a style. Each style has an expression, a color and if you want it to blink. The expression allows you to do any evaluation you want using any parts of the alarm: Priority, State, Display Path, Active time, Clear time, etc.

This is an ordered list of alarm row styles. The first style that returns **'True'** for a given alarm is the one that is going to be used so you want to make sure the order is correct in the Alarm Rows Styles Customizer.

In this example, you will notice that **{priority}='Diagnostic**' is the first style and it is using a gray color scheme. This means that any alarm with the 'Diagnostic' priority will be gray, regardless of it's state.



Alarm Row Styles

Row Styles
{priority}='Diagnostic'
{state}='ActiveUnacked' && {priority}>=3
{state}='ActiveUnacked'
{state}='ActiveAcked'
{state}='ClearUnacked'
{state}='ClearAcked'

Expression

```
{priority}='Diagnostic'
```

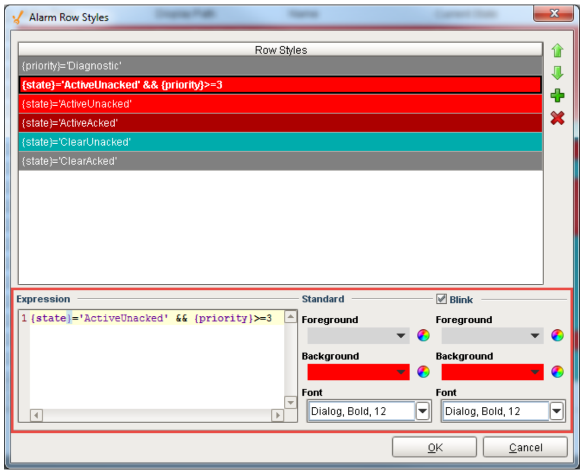
The second style is a **State 'ActiveUnacked'** alarm with a **priority of 3** or greater. If this is **'True,'** the red color scheme will be used and it will blink. IE: any alarm that is **Active**, **Unacknowledged**, and **High or Critical priority** will show as red and blinking.

Note: *Priorities are numbered behind the scenes, but named when you are creating them. 0 = Diagnostic, 1 = Low, 2 = Medium, 3 = High, and 4 = Critical. This allows you to use operations like '>=' (greater than or equal) when filtering on priority.*

INDUCTIVE UNIVERSITY

Alarm Status - Row Styles

Watch the Video



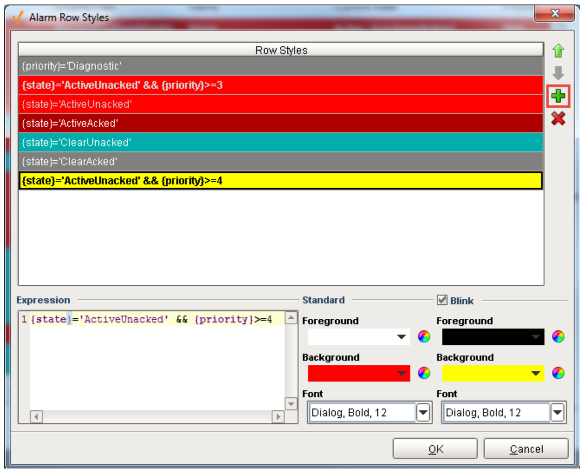
(priority)='Diagnostic'
{state}='ActiveUnacked' && {priority}>=3
{state}='ActiveUnacked'
{state}='ActiveAcked'
{state}='ClearUnacked'
{state}='ClearAcked'

Expression

```
{state}='ActiveUnacked' && {priority}>=3
```

Foreground / Background / Font: Dialog, Bold, 12 — Foreground / Background / Font: Dialog, Bold, 12

Let's add a new State alarm for **Active,Unacked** and **Critical** alarms. You can copy your new expression from the second style **'Active,Unacked'** and paste it into your new expression and change the **Priority** to **'4.'** Because this is a critical alarm, you want it to catch the operator's attention, so it's a good idea to make the style standout. Make the foreground color **'white'** and the background color **'red.'** Che ck the **blink box** and make the foreground color of **'black'** and background color of **'yellow**' so it is visible at the top of the Alarm Status Table.



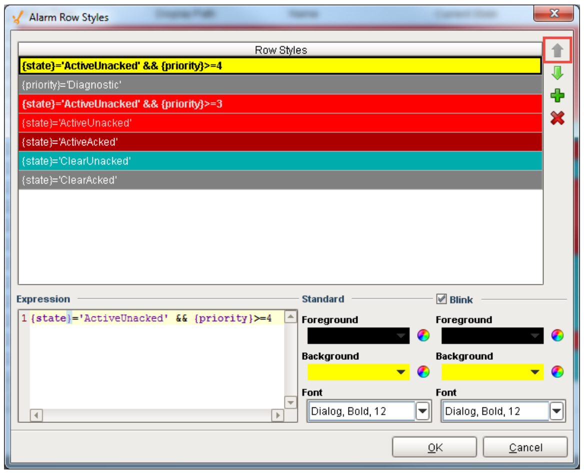
(priority)='Diagnostic'
{state}='ActiveUnacked' && {priority}>=3
{state}='ActiveUnacked'
{state}='ActiveAcked'
{state}='ClearUnacked'
{state}='ClearAcked'
{state}='ActiveUnacked' && {priority}>=4

Expression

```
{state}='ActiveUnacked' && {priority}>=4
```

Foreground / Background / Font: Dialog, Bold, 12 — Foreground / Background / Font: Dialog, Bold, 12

Remember what was stated earlier, that style order is important. So, if you keep this style at the bottom, the alarm will never be visible to the operator because the first state to return 'True' is **ActiveUnacked** wit h a **Priority >= 3**. You need to move **ActiveUnacked** with a **Priority >= 4** to the top, so it is evaluated before the others. **Press OK**.



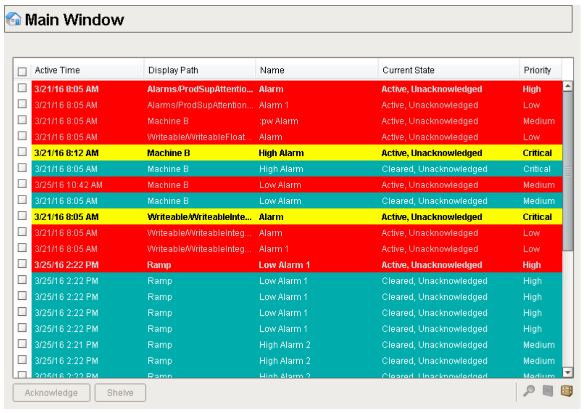
{state}='ActiveUnacked' && {priority}>=4
(priority)='Diagnostic'
{state}='ActiveUnacked' && {priority}>=3
{state}='ActiveUnacked'
{state}='ActiveAcked'
{state}='ClearUnacked'
{state}='ClearAcked'

Expression

```
{state}='ActiveUnacked' && {priority}>=4
```

Foreground / Background / Font: Dialog, Bold, 12 — Foreground / Background / Font: Dialog, Bold, 12

The critical alarms will now be blinking yellow and will definitely catch the operator's attention.


Main Window

Active Time	Display Path	Name	Current State	Priority
3/21/16 8:05 AM	**Alarms/ProdSupAttentio...**	**Alarm**	**Active, Unacknowledged**	**High**
3/21/16 8:05 AM	Alarms/ProdSupAttention...	Alarm 1	Active, Unacknowledged	Low
3/21/16 8:05 AM	Machine B	:pw Alarm	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Writeable/WriteableFloat...	Alarm	Active, Unacknowledged	Low
3/21/16 8:12 AM	**Machine B**	**High Alarm**	**Active, Unacknowledged**	**Critical**
3/21/16 8:05 AM	Machine B	High Alarm	Cleared, Unacknowledged	Critical
3/25/16 10:42 AM	Machine B	Low Alarm	Active, Unacknowledged	Medium
3/21/16 8:05 AM	Machine B	Low Alarm	Cleared, Unacknowledged	Medium
3/21/16 8:05 AM	**Writeable/WriteableInte...**	**Alarm**	**Active, Unacknowledged**	**Critical**
3/21/16 8:05 AM	Writeable/WriteableInteg...	Alarm	Active, Unacknowledged	Low
3/21/16 8:05 AM	Writeable/WriteableInteg...	Alarm 1	Active, Unacknowledged	Low
3/25/16 2:22 PM	**Ramp**	**Low Alarm 1**	**Active, Unacknowledged**	**High**
3/25/16 2:22 PM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 2:22 PM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 2:22 PM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 2:22 PM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
3/25/16 2:21 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
3/25/16 2:22 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
3/25/16 2:22 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium

To learn more about States and Priorities, including all the alarm configuration properties, refer to the sections on Alarm Properties.

Next...

- Alarm Status - Marquee Mode

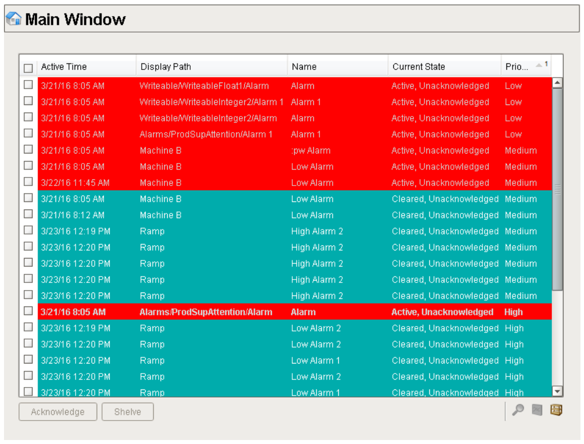
Alarm Status - Marquee Mode

The Alarm Status Table can take up a lot of space on the client window as shown in this example, but if a small **Alarm Status Table** is needed, you can turn on the **Marquee Mode** option.

INDUCTIVE UNIVERSITY

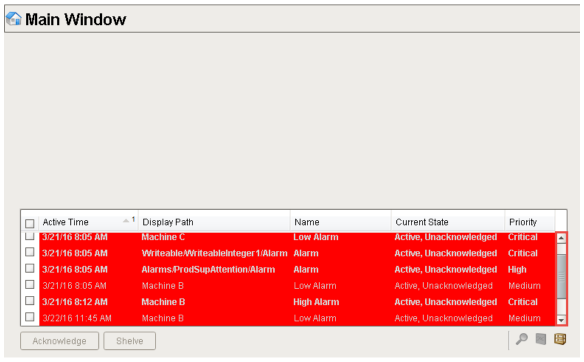
Alarm Status-Marquee Mode

Watch the Video



	Active Time	Display Path	Name	Current State	Prio...
☐	3/21/16 8:05 AM	Writeable/WriteableFloat1/Alarm	Alarm	Active, Unacknowledged	Low
☐	3/21/16 8:05 AM	Writeable/WriteableInteger2/Alarm 1	Alarm 1	Active, Unacknowledged	Low
☐	3/21/16 8:05 AM	Writeable/WriteableInteger2/Alarm	Alarm	Active, Unacknowledged	Low
☐	3/21/16 8:05 AM	Alarms/ProdSupAttention/Alarm 1	Alarm 1	Active, Unacknowledged	Low
☐	3/21/16 8:05 AM	Machine B	:pw Alarm	Active, Unacknowledged	Medium
☐	3/21/16 8:05 AM	Machine B	Low Alarm	Active, Unacknowledged	Medium
☐	3/22/16 11:45 AM	Machine B	Low Alarm	Active, Unacknowledged	Medium
☐	3/21/16 8:05 AM	Machine B	Low Alarm	Cleared, Unacknowledged	Medium
☐	3/21/16 8:12 AM	Machine B	Low Alarm	Cleared, Unacknowledged	Medium
☐	3/23/16 12:19 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
☐	3/23/16 12:20 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
☐	3/23/16 12:20 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
☐	3/23/16 12:20 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
☐	3/23/16 12:20 PM	Ramp	High Alarm 2	Cleared, Unacknowledged	Medium
☐	**3/21/16 8:05 AM**	**Alarms/ProdSupAttention/Alarm**	**Alarm**	**Active, Unacknowledged**	**High**
☐	3/23/16 12:19 PM	Ramp	Low Alarm 2	Cleared, Unacknowledged	High
☐	3/23/16 12:20 PM	Ramp	Low Alarm 2	Cleared, Unacknowledged	High
☐	3/23/16 12:20 PM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High
☐	3/23/16 12:20 PM	Ramp	Low Alarm 2	Cleared, Unacknowledged	High
☐	3/23/16 12:20 PM	Ramp	Low Alarm 1	Cleared, Unacknowledged	High

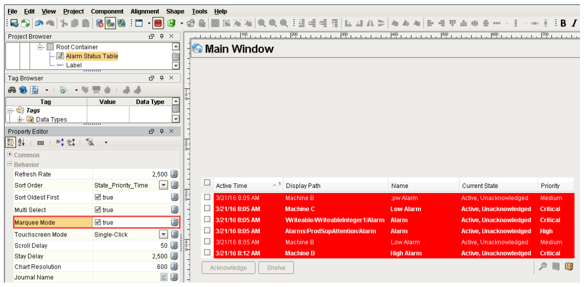
In many instances, a small Alarm Status Table is displayed at the very bottom of the Ignition client window taking up very little space. The problem is when an operator has quite a few Active alarms as shown in this example, they have to manually scroll through the list of alarms to see them all.   This can be a little tedious for the operator.



	Active Time	Display Path	Name	Current State	Priority
☐	**3/21/16 8:05 AM**	**Machine C**	**Low Alarm**	**Active, Unacknowledged**	**Critical**
☐	**3/21/16 8:05 AM**	**Writeable/WriteableInteger1/Alarm**	**Alarm**	**Active, Unacknowledged**	**Critical**
☐	**3/21/16 8:05 AM**	**Alarms/ProdSupAttention/Alarm**	**Alarm**	**Active, Unacknowledged**	**High**
☐	3/21/16 8:05 AM	Machine B	Low Alarm	Active, Unacknowledged	Medium
☐	**3/21/16 8:12 AM**	**Machine B**	**High Alarm**	**Active, Unacknowledged**	**Critical**
☐	3/22/16 11:45 AM	Machine B	Low Alarm	Active, Unacknowledged	Medium

The nice feature with Marquee Mode is it automatically scrolls through all the alarms when there is not enough vertical space on the client window to show them all at once.

In the **Property Editor** of the Alarm Status Table, there is a property called **Marquee Mode**. Set the property to **'True'** and the Alarm Status Table will automatically scroll through all alarms that are currently visible on the Alarm Status Table component.



Refresh Rate	2,500
Sort Order	State_Priority_Time
Sort Oldest First	☑ true
Multi Select	☑ true
Marquee Mode	☑ true
Touchscreen Mode	Single-Click
Scroll Delay	50
Stay Delay	2,500
Chart Resolution	600
Journal Name	

	Active Time	Display Path	Name	Current State	Priority
☐	3/21/16 8:05 AM	Machine B	:pw Alarm	Active, Unacknowledged	Medium
☐	**3/21/16 8:05 AM**	**Machine C**	**Low Alarm**	**Active, Unacknowledged**	**Critical**
☐	**3/21/16 8:05 AM**	**Writeable/WriteableInteger1/Alarm**	**Alarm**	**Active, Unacknowledged**	**Critical**
☐	**3/21/16 8:05 AM**	**Alarms/ProdSupAttention/Alarm**	**Alarm**	**Active, Unacknowledged**	**High**
☐	3/21/16 8:05 AM	Machine B	Low Alarm	Active, Unacknowledged	Medium
☐	**3/21/16 8:12 AM**	**Machine B**	**High Alarm**	**Active, Unacknowledged**	**Critical**

Marquee Mode makes it very easy for an operator to see what's going on without having to scroll through the entire alarm list to identify all the alarms that are Active or whatever state they may be looking for.

When using Marquee mode, it is even more important to pay attention to the filters on the Alarm Status Table since there is limited room. Make sure your operators know that it is a scrolling list.

Next...

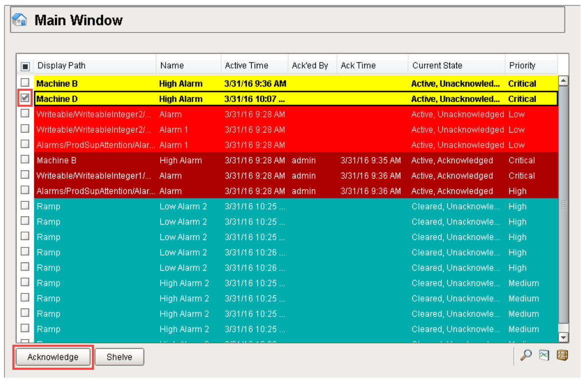
- Alarm Status - Acknowledgement

Alarm Status - Acknowledgement

Using Alarm Acknowledgement

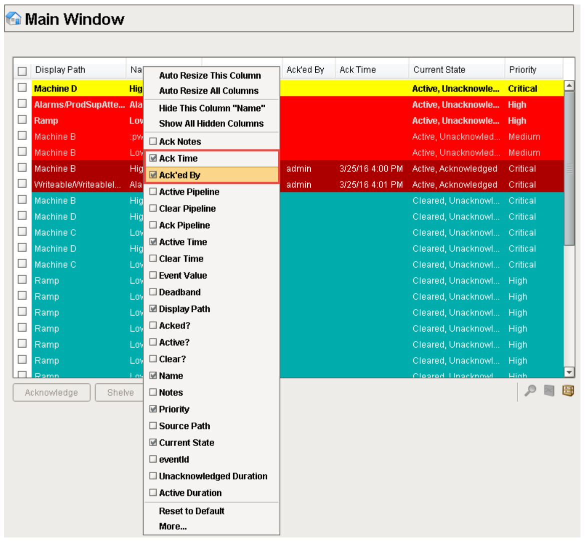
One of the most important things an operator is going to do in the runtime is acknowledge alarms using the **Alarm Status Table**. Alarm acknowledgement is built into the component and is very easy to use.

As soon as the operator selects and presses the Acknowledge button, the current State of the alarm will change, and the operator's credentials and the time the alarm was acknowledged will be recorded in the Alarm Status Table.



	Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
☐	Machine B	High Alarm	3/31/16 9:36 AM			Active, Unacknowled...	Critical
☑	Machine D	High Alarm	3/31/16 10:07 ...			Active, Unacknowled...	Critical
☐	Writeable/WriteableInteger2/...	Alarm	3/31/16 9:28 AM			Active, Unacknowledged	Low
☐	Writeable/WriteableInteger2/...	Alarm 1	3/31/16 9:28 AM			Active, Unacknowledged	Low
☐	Alarms/ProdSupAttention/Alar...	Alarm 1	3/31/16 9:28 AM			Active, Unacknowledged	Low
☐	Machine B	High Alarm	3/31/16 9:28 AM	admin	3/31/16 9:35 AM	Active, Acknowledged	Critical
☐	Writeable/WriteableInteger1/...	Alarm	3/31/16 9:28 AM	admin	3/31/16 9:36 AM	Active, Acknowledged	Critical
☐	Alarms/ProdSupAttention/Alar...	Alarm	3/31/16 9:28 AM	admin	3/31/16 9:36 AM	Active, Acknowledged	High
☐	Ramp	Low Alarm 2	3/31/16 10:25 ...			Cleared, Unacknowle...	High
☐	Ramp	Low Alarm 2	3/31/16 10:25 ...			Cleared, Unacknowle...	High
☐	Ramp	Low Alarm 2	3/31/16 10:25 ...			Cleared, Unacknowle...	High
☐	Ramp	Low Alarm 2	3/31/16 10:26 ...			Cleared, Unacknowle...	High
☐	Ramp	Low Alarm 2	3/31/16 10:26 ...			Cleared, Unacknowle...	High
☐	Ramp	High Alarm 2	3/31/16 10:25 ...			Cleared, Unacknowle...	Medium
☐	Ramp	High Alarm 2	3/31/16 10:25 ...			Cleared, Unacknowle...	Medium
☐	Ramp	High Alarm 2	3/31/16 10:25 ...			Cleared, Unacknowle...	Medium
☐	Ramp	High Alarm 2	3/31/16 10:25 ...			Cleared, Unacknowle...	Medium

If you don't have the Ack'ed By or Ack Time columns in the header of the Alarm Status Table, you can simply add them by right clicking on the header bar, and from the dropdown list, check the boxes for **Ack'ed By** and **Ack Time.**



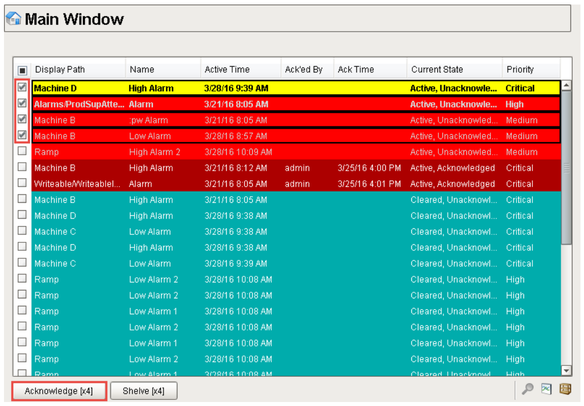
	Display Path	Na	Ack'ed By	Ack Time	Current State	Priority
☐	Machine D	Hig			Active, Unacknowle...	Critical
☐	Alarms/ProdSupAtte...	Ala			Active, Unacknowle...	High
☐	Ramp	Lov			Active, Unacknowle...	High
☐	Machine B	:pw			Active, Unacknowled...	Medium
☐	Machine B	Lov			Active, Unacknowled...	Medium
☐	Machine B	Hig	admin	3/25/16 4:00 PM	Active, Acknowledged	Critical
☐	Writeable/Writeablel...	Ala	admin	3/25/16 4:01 PM	Active, Acknowledged	Critical
☐	Machine B	Hig			Cleared, Unacknowl...	Critical
☐	Machine D	Hig			Cleared, Unacknowl...	Critical
☐	Machine C	Lov			Cleared, Unacknowl...	Critical
☐	Machine D	Hig			Cleared, Unacknowl...	Critical
☐	Machine C	Lov			Cleared, Unacknowl...	Critical
☐	Ramp	Lov			Cleared, Unacknowl...	High
☐	Ramp	Lov			Cleared, Unacknowl...	High
☐	Ramp	Lov			Cleared, Unacknowl...	High
☐	Ramp	Lov			Cleared, Unacknowl...	High
☐	Ramp	Lov			Cleared, Unacknowl...	High
☐	Ramp	Lov			Cleared, Unacknowl...	High

The Alarm Status Table component allows you to select an individual alarm, multiple alarms or use the select all checkbox in the header bar. Once you check all of the alarm(s) and press the Acknowledged button, all the alarms will be Acknowledged. In this example, 4 alarms were checked, and all 4 alarms were Acknowledged at the same time.

INDUCTIVE UNIVERSITY

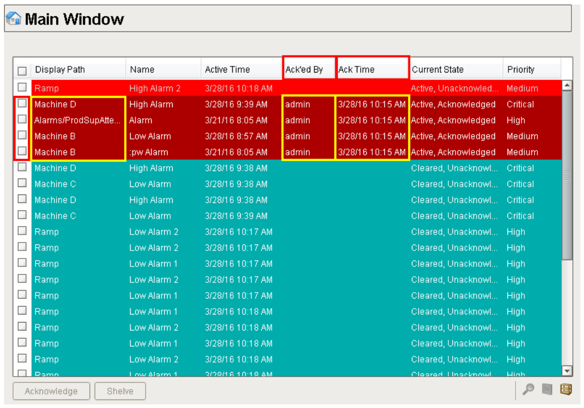
Alarm Status - Acknowledgement

Watch the Video



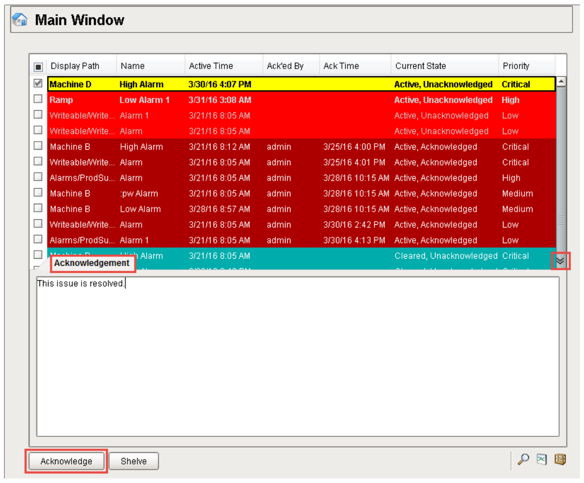
	Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
✓	Machine D	High Alarm	3/28/16 9:39 AM			Active, Unacknowle...	Critical
✓	Alarms/ProdSupAtte...	Alarm	3/21/16 8:05 AM			Active, Unacknowle...	High
✓	Machine B	:pw Alarm	3/21/16 8:05 AM			Active, Unacknowled...	Medium
✓	Machine B	Low Alarm	3/28/16 8:57 AM			Active, Unacknowled...	Medium
☐	Ramp	High Alarm 2	3/28/16 10:09 AM			Active, Unacknowled...	Medium
☐	Machine B	High Alarm	3/21/16 8:12 AM	admin	3/25/16 4:00 PM	Active, Acknowledged	Critical
☐	Writeable/Writeablel...	Alarm	3/21/16 8:05 AM	admin	3/25/16 4:01 PM	Active, Acknowledged	Critical
☐	Machine B	High Alarm	3/21/16 8:05 AM			Cleared, Unacknowl...	Critical
☐	Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
☐	Machine C	Low Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
☐	Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
☐	Machine C	Low Alarm	3/28/16 9:39 AM			Cleared, Unacknowl...	Critical
☐	Ramp	Low Alarm 2	3/28/16 10:08 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 2	3/28/16 10:08 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 1	3/28/16 10:08 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 2	3/28/16 10:08 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 1	3/28/16 10:08 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 2	3/28/16 10:08 AM			Cleared, Unacknowl...	High

This example shows all 4 alarms were Acknowledged, who acknowledged them and the time they were acknowledged.



	Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
☐	Ramp	High Alarm 2	3/28/16 10:18 AM			Active, Unacknowled...	Medium
☐	Machine D	High Alarm	3/28/16 9:39 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	Critical
☐	Alarms/ProdSupAtte...	Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	High
☐	Machine B	Low Alarm	3/28/16 8:57 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	Medium
☐	Machine B	:pw Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	Medium
☐	Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
☐	Machine C	Low Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
☐	Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
☐	Machine C	Low Alarm	3/28/16 9:39 AM			Cleared, Unacknowl...	Critical
☐	Ramp	Low Alarm 2	3/28/16 10:17 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 2	3/28/16 10:17 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 1	3/28/16 10:17 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 2	3/28/16 10:17 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 1	3/28/16 10:17 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 1	3/28/16 10:18 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 2	3/28/16 10:18 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 1	3/28/16 10:18 AM			Cleared, Unacknowl...	High
☐	Ramp	Low Alarm 2	3/28/16 10:18 AM			Cleared, Unacknowl...	High

If any of the selected alarms require **Acknowledgement Notes**, a small window is displayed in which the operator must add notes, otherwise, the alarm cannot be acknowledged. If the operator wants to cancel the Acknowledgement Notes and close the window, use the **Expand button** to close the Acknowledgement Notes window.



	Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
✓	Machine D	High Alarm	3/30/16 4:07 PM			Active, Unacknowledged	Critical
☐	Ramp	Low Alarm 1	3/31/16 3:08 AM			Active, Unacknowledged	High
☐	Writeable/Write...	Alarm 1	3/21/16 8:05 AM			Active, Unacknowledged	Low
☐	Writeable/Write...	Alarm	3/21/16 8:05 AM			Active, Unacknowledged	Low
☐	Machine B	High Alarm	3/21/16 8:12 AM	admin	3/25/16 4:00 PM	Active, Acknowledged	Critical
☐	Writeable/Write...	Alarm	3/21/16 8:05 AM	admin	3/25/16 4:01 PM	Active, Acknowledged	Critical
☐	Alarms/ProdSu...	Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	High
☐	Machine B	:pw Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	Medium
☐	Machine B	Low Alarm	3/28/16 8:57 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	Medium
☐	Writeable/Write...	Alarm	3/21/16 8:05 AM	admin	3/30/16 2:42 PM	Active, Acknowledged	Low
☐	Alarms/ProdSu...	Alarm 1	3/21/16 8:05 AM	admin	3/30/16 4:13 PM	Active, Acknowledged	Low
☐	[illegible]	High Alarm	3/21/16 8:05 AM			Cleared, Unacknowledged	Critical

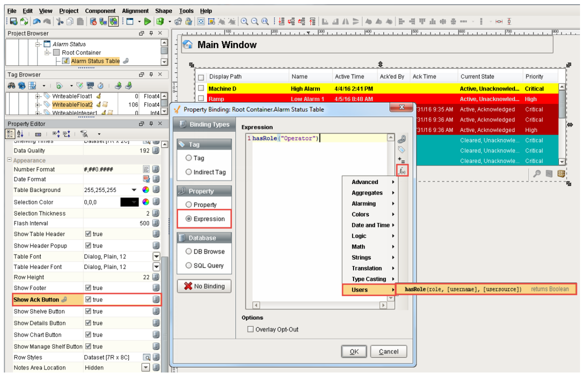
This issue is resolved.

Security for Alarm Acknowledgement

If you want to restrict who can use the Acknowledge button, there is the **Show Ack Button** in the Property Editor that can be set to **'False.'** By setting the **Show Ack Button** to '**False**,' this hides the Acknowledge button on the Alarm Status Table.

In order for operators to acknowledge alarms, the correct permission must be assigned. Permissions can be assigned using the Show Ack Button. Click the **Show Ack Button binding icon** to open the property binding window. This example shows how to set permissions to acknowledge alarms for people in the **Operator Role**.

1. Under **Property Binding Type**, select **Expression**.
2. Click the **Function icon.**
3. Scroll down to **Users,** and select '**hasRole.'** This enters the function name.
4. Edit the expression to read: **hasRole("Operator")**
5. Click **OK**.
   If you currently have the "Operator" role you will notice that the **Show Ack Button** property is now true, otherwise it will be False.


Property Binding: Root Container.Alarm Status Table

Expression

```
hasRole("Operator")
```

```
hasRole(role, [username], [usersource])    returns Boolean
```

Data Quality	192
Number Format	#,##0.####
Date Format	
Table Background	255,255,255
Selection Color	0,0,0
Selection Thickness	2
Flash Interval	500
Show Table Header	✓ true
Show Header Popup	✓ true
Table Font	Dialog, Plain, 12
Table Header Font	Dialog, Plain, 12
Row Height	22
Show Footer	✓ true
Show Ack Button	✓ true
Show Shelve Button	✓ true
Show Details Button	✓ true
Show Chart Button	✓ true
Show Manage Shelf Button	✓ true
Row Styles	Dataset [7R x 8C]
Notes Area Location	Hidden

Next...

- Alarm Status - Shelving

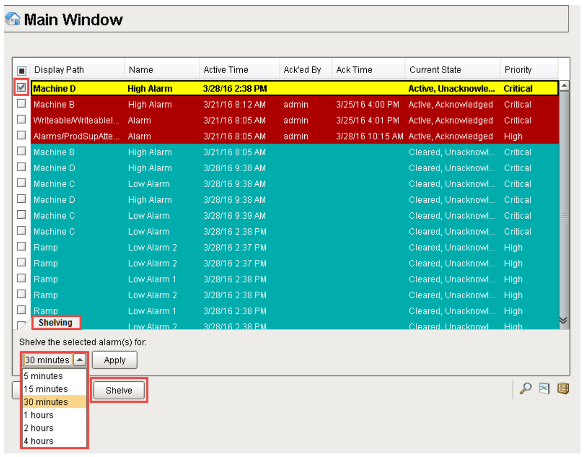
Alarm Status - Shelving

Another important feature of the Alarm Status Table is the capability to shelve alarms. Shelving alarms allows you to temporarily silence an alarm for a fixed period of time. This feature is extremely handy when an alarm is already active and you want to temporarily silence the alarm while you're working on the issue. The Alarm Status Table component will not send any notifications while the alarm is shelved, and will be temporarily dropped from the Alarm Status list so operators don't get confused and think it's active. When the shelved time is up and if the alarm is still active, it will return into the Alarm Status list.

Alarm Status - Shelving

Watch the Video

In this example, you can see one critical alarm which is Active and Unacknowledged. You can select one alarm or multiple alarms, then press the **Shelve button**. The **Shelving tab** will automatically open so you can set a duration from 5 minutes to 4 hours in which to shelve selected alarm(s). The shelving time is also customizable. Select **30 minutes** and press **Apply**. The selected alarm(s) will be removed from the Alarm Status list.



Main Window

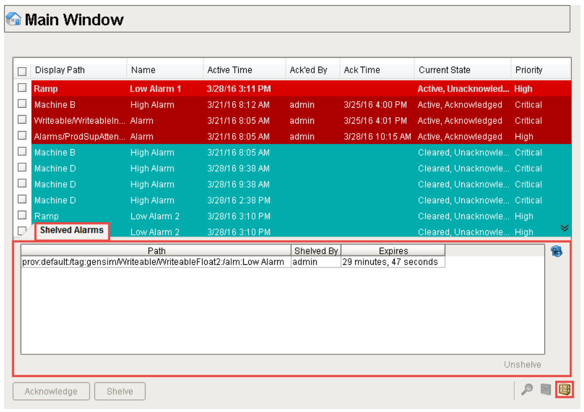
Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
Machine D	High Alarm	3/28/16 2:38 PM			Active, Unacknowle...	Critical
Machine B	High Alarm	3/21/16 8:12 AM	admin	3/25/16 4:00 PM	Active, Acknowledged	Critical
Writeable/Writeablel...	Alarm	3/21/16 8:05 AM	admin	3/25/16 4:01 PM	Active, Acknowledged	Critical
Alarms/ProdSupAtte...	Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	High
Machine B	High Alarm	3/21/16 8:05 AM			Cleared, Unacknowl...	Critical
Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
Machine C	Low Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
Machine C	Low Alarm	3/28/16 9:39 AM			Cleared, Unacknowl...	Critical
Machine C	Low Alarm	3/28/16 2:38 PM			Cleared, Unacknowl...	Critical
Ramp	Low Alarm 2	3/28/16 2:37 PM			Cleared, Unacknowl...	High
Ramp	Low Alarm 2	3/28/16 2:37 PM			Cleared, Unacknowl...	High
Ramp	Low Alarm 1	3/28/16 2:38 PM			Cleared, Unacknowl...	High
Ramp	Low Alarm 2	3/28/16 2:38 PM			Cleared, Unacknowl...	High
Ramp	Low Alarm 1	3/28/16 2:38 PM			Cleared, Unacknowl...	High
	Low Alarm 2	3/28/16 2:38 PM			Cleared, Unacknowl...	High

Shelving

Shelve the selected alarm(s) for:

30 minutes

To view all the alarms that are currently shelved, go to the bottom right of the Alarm Status Table and click on the **Shelved Alarms icon.** A Shelved Alarm tab will open and you can see all the alarms that are shelved and when the shelved time on the alarm will expire. Click the **Shelved icon** again, and the Shelved Alarm tab will disappear.



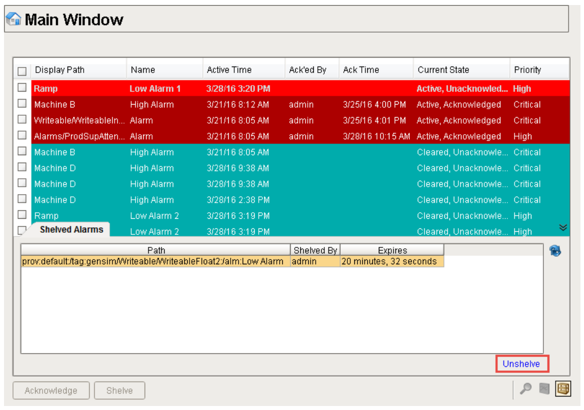
Main Window

Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
Ramp	Low Alarm 1	3/28/16 3:11 PM			Active, Unacknowled...	High
Machine B	High Alarm	3/21/16 8:12 AM	admin	3/25/16 4:00 PM	Active, Acknowledged	Critical
Writeable/WriteableIn...	Alarm	3/21/16 8:05 AM	admin	3/25/16 4:01 PM	Active, Acknowledged	Critical
Alarms/ProdSupAtten...	Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	High
Machine B	High Alarm	3/21/16 8:05 AM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	3/28/16 2:38 PM			Cleared, Unacknowle...	Critical
Ramp	Low Alarm 2	3/28/16 3:10 PM			Cleared, Unacknowle...	High
	Low Alarm 2	3/28/16 3:10 PM			Cleared, Unacknowle...	High

Shelved Alarms

Path	Shelved By	Expires
prov:default:/tag:gensim/Writeable/WriteableFloat2:/alm:Low Alarm	admin	29 minutes, 47 seconds

If you want to Unshelve the alarm, **select the alarm** under the **Shelved Alarms tab**, click the **Unshelved link,** and the alarm will return back to the Alarm Status list. After the amount of time expires on a shelved alarm, it will be evaluated, and if it is still active, it will automatically return to the Alarm Status list.


Main Window

Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
Ramp	Low Alarm 1	3/28/16 3:20 PM			Active, Unacknowled...	High
Machine B	High Alarm	3/21/16 8:12 AM	admin	3/25/16 4:00 PM	Active, Acknowledged	Critical
Writeable/WriteableIn...	Alarm	3/21/16 8:05 AM	admin	3/25/16 4:01 PM	Active, Acknowledged	Critical
Alarms/ProdSupAtten...	Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	High
Machine B	High Alarm	3/21/16 8:05 AM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowle...	Critical
Machine D	High Alarm	3/28/16 2:38 PM			Cleared, Unacknowle...	Critical
Ramp	Low Alarm 2	3/28/16 3:19 PM			Cleared, Unacknowle...	High
	Low Alarm 2	3/28/16 3:19 PM			Cleared, Unacknowle...	High

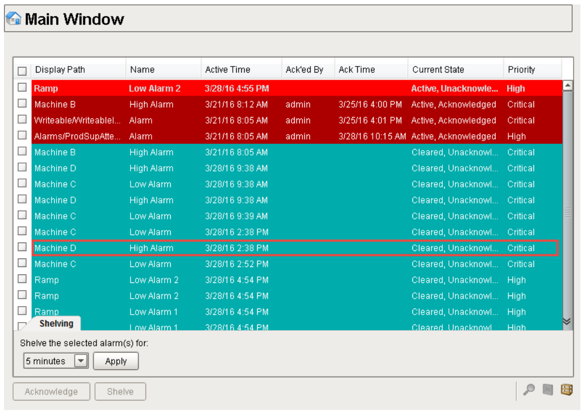
Shelved Alarms

Path	Shelved By	Expires
prov:default:/tag:gensim/Writeable/WriteableFloat2:/alm:Low Alarm	admin	20 minutes, 32 seconds

Unshelve

Acknowledge Shelve

If the alarm is fixed during this time period (i.e., tank water level is low and when the water level returns to the setpoint), the alarm will show up as Cleared in the Alarm Status list making it no longer Active as shown in this example.


Main Window

Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
Ramp	Low Alarm 2	3/28/16 4:55 PM			Active, Unacknowle...	High
Machine B	High Alarm	3/21/16 8:12 AM	admin	3/25/16 4:00 PM	Active, Acknowledged	Critical
Writeable/Writeablel...	Alarm	3/21/16 8:05 AM	admin	3/25/16 4:01 PM	Active, Acknowledged	Critical
Alarms/ProdSupAtte...	Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	High
Machine B	High Alarm	3/21/16 8:05 AM			Cleared, Unacknowl...	Critical
Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
Machine C	Low Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
Machine D	High Alarm	3/28/16 9:38 AM			Cleared, Unacknowl...	Critical
Machine C	Low Alarm	3/28/16 9:39 AM			Cleared, Unacknowl...	Critical
Machine C	Low Alarm	3/28/16 2:38 PM			Cleared, Unacknowl...	Critical
Machine D	High Alarm	3/28/16 2:38 PM			Cleared, Unacknowl...	Critical
Machine C	Low Alarm	3/28/16 2:52 PM			Cleared, Unacknowl...	Critical
Ramp	Low Alarm 2	3/28/16 4:54 PM			Cleared, Unacknowl...	High
Ramp	Low Alarm 2	3/28/16 4:54 PM			Cleared, Unacknowl...	High
Ramp	Low Alarm 1	3/28/16 4:54 PM			Cleared, Unacknowl...	High
	Low Alarm 1	3/28/16 4:54 PM			Cleared, Unacknowl...	High

Shelving

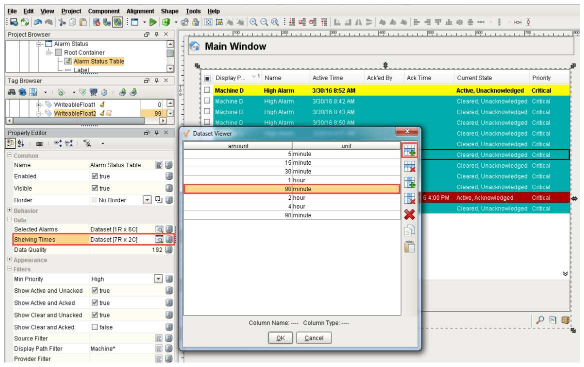
Shelve the selected alarm(s) for:

5 minutes Apply

Acknowledge Shelve

Another nice feature of the Alarm Status Table is the shelve time is totally customizable. By default, the component has several shelving preset times, but you can add or remove times based on your requirements. One thing to note with modifying the shelving times, is that there is no option to set a shelving time indefinitely. Shelving times are meant to be temporary, not long term.

In the **Property Editor**, there is a property called **Shelving Times.** Click on the **Dataset Viewer** icon to open the Dataset Viewer. This is where you can add, remove or change the shelving times for alarms. Once you edited the shelving times, press **OK**. You can see in this example that **90 minutes** was added to the list of shelving times.



Property Editor

Common	
Name	Alarm Status Table
Enabled	true
Visible	true
Border	No Border
Behavior	
Data	
Selected Alarms	Dataset [1R x 6C]
Shelving Times	Dataset [7R x 2C]
Data Quality	192
Appearance	
Filters	
Min Priority	High
Show Active and Unacked	true
Show Active and Acked	true
Show Clear and Unacked	true
Show Clear and Acked	false
Source Filter	
Display Path Filter	Machine*
Provider Filter	

Dataset Viewer

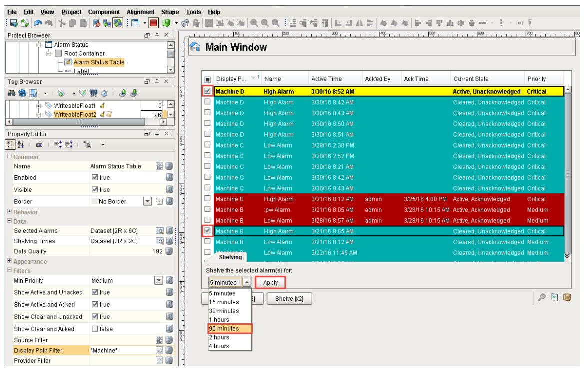
amount	unit
5	minute
15	minute
30	minute
1	hour
90	minute
2	hour
4	hour
90	minute

Column Name: ---- Column Type: ----

OK Cancel

Display P...	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
Machine D	High Alarm	3/30/16 8:52 AM			Active, Unacknowledged	Critical
Machine D	High Alarm	3/30/16 8:42 AM			Cleared, Unacknowledged	Critical
Machine D	High Alarm	3/30/16 8:43 AM			Cleared, Unacknowledged	Critical
Machine D	High Alarm	3/30/16 8:50 AM			Cleared, Unacknowledged	Critical

To verify your new time was entered correctly, select the alarm(s) you want to shelve and click the Shelve button. Choose the shelve time from the dropdown list and press **Apply** to temporarily silence your alarm(s). You can see how easy it is to modify your shelve times.



	Display P...	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
✓	Machine D	High Alarm	3/30/16 8:52 AM			Active, Unacknowledged	Critical
	Machine D	High Alarm	3/30/16 8:42 AM			Cleared, Unacknowledged	Critical
	Machine D	High Alarm	3/30/16 8:43 AM			Cleared, Unacknowledged	Critical
	Machine D	High Alarm	3/30/16 8:50 AM			Cleared, Unacknowledged	Critical
	Machine D	High Alarm	3/30/16 8:51 AM			Cleared, Unacknowledged	Critical
	Machine C	Low Alarm	3/28/16 2:38 PM			Cleared, Unacknowledged	Critical
	Machine C	Low Alarm	3/28/16 2:52 PM			Cleared, Unacknowledged	Critical
	Machine C	Low Alarm	3/30/16 8:21 AM			Cleared, Unacknowledged	Critical
	Machine C	Low Alarm	3/30/16 8:42 AM			Cleared, Unacknowledged	Critical
	Machine C	Low Alarm	3/30/16 8:43 AM			Cleared, Unacknowledged	Critical
	Machine B	High Alarm	3/21/16 8:12 AM	admin	3/25/16 4:00 PM	Active, Acknowledged	Critical
	Machine B	:pw Alarm	3/21/16 8:05 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	Medium
	Machine B	Low Alarm	3/28/16 8:57 AM	admin	3/28/16 10:15 AM	Active, Acknowledged	Medium
✓	Machine B	High Alarm	3/21/16 8:05 AM			Cleared, Unacknowledged	Critical
	Machine B	Low Alarm	3/21/16 8:12 AM			Cleared, Unacknowledged	Medium
		Low Alarm	3/22/16 11:45 AM			Cleared, Unacknowledged	Medium

Shelve the selected alarm(s) for: 5 minutes [Apply]

- 5 minutes
- 15 minutes
- 30 minutes
- 1 hours
- 90 minutes
- 2 hours
- 4 hours

Common	
Name	Alarm Status Table
Enabled	✓ true
Visible	✓ true
Border	No Border
Behavior	
Data	
Selected Alarms	Dataset [2R x 6C]
Shelving Times	Dataset [7R x 2C]
Data Quality	192
Appearance	
Filters	
Min Priority	Medium
Show Active and Unacked	✓ true
Show Active and Acked	✓ true
Show Clear and Unacked	✓ true
Show Clear and Acked	false
Source Filter	
Display Path Filter	*Machine*
Provider Filter	

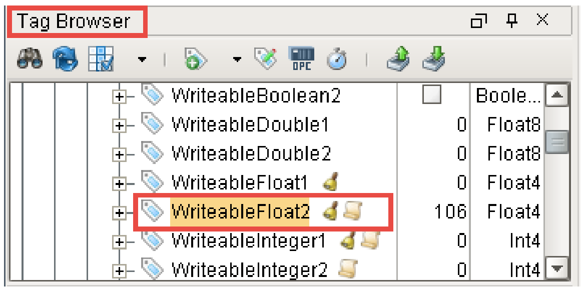
Next...

- Alarm Status - Tag History and Alarm History

Alarm Status - Tag History and Alarm History

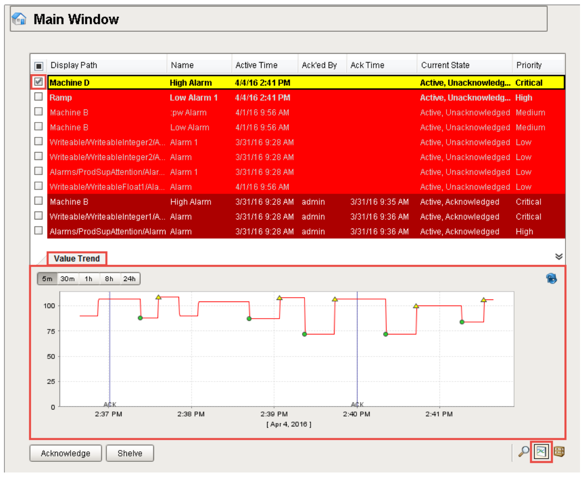
The **Alarm Status Table** component has a unique feature built into it that lets you view the actual trend of a tag's value and an alarm's history for any alarm that is currently active. It shows you exactly where the alarms occurred, when it was cleared, and when it was acknowledged.

For the trend feature to work, you must have your tag history setup so it’s logging to the Tag Historian as well as an Alarm Journal Profile configured so that the log history gets logged to a database. As long as both the Tag Historian and Alarm Journal are configured, you will see the trend for any given alarm that is currently active or in the Alarm Status Table list.



Tag Browser		
WriteableBoolean2	☐	Boole...
WriteableDouble1	0	Float8
WriteableDouble2	0	Float8
WriteableFloat1	0	Float4
WriteableFloat2	106	Float4
WriteableInteger1	0	Int4
WriteableInteger2	0	Int4

In the Alarm Status Table, select any **alarm** and click on the **Trend icon** located on the bottom right of the table. The trend chart shows you the trend of the selected alarm's value over time.   You can look at the last 5 minutes, 30 minutes, 1 hour, 8 hours or up to 24 hours to view the exact history of that tag's value, as well as all the alarms that occurred over time. The yellow triangles denote when the alarm was active, the green circles identify when the alarm was cleared, and a vertical black bar shows when the alarm was acknowledged.   It’s all built into the Alarm Status Table component and you don’t have to go to a separate window to view an alarm trend or alarm history.



Main Window

	Display Path	Name	Active Time	Ack'ed By	Ack Time	Current State	Priority
☑	Machine D	High Alarm	4/4/16 2:41 PM			Active, Unacknowledg...	Critical
☐	Ramp	Low Alarm 1	4/4/16 2:41 PM			Active, Unacknowledg...	High
☐	Machine B	:pw Alarm	4/1/16 9:56 AM			Active, Unacknowledged	Medium
☐	Machine B	Low Alarm	4/1/16 9:56 AM			Active, Unacknowledged	Medium
☐	Writeable/WriteableInteger2/A...	Alarm 1	3/31/16 9:28 AM			Active, Unacknowledged	Low
☐	Writeable/WriteableInteger2/A...	Alarm	3/31/16 9:28 AM			Active, Unacknowledged	Low
☐	Alarms/ProdSupAttention/Alar...	Alarm 1	3/31/16 9:28 AM			Active, Unacknowledged	Low
☐	Writeable/WriteableFloat1/Ala...	Alarm	4/1/16 9:56 AM			Active, Unacknowledged	Low
☐	Machine B	High Alarm	3/31/16 9:28 AM	admin	3/31/16 9:35 AM	Active, Acknowledged	Critical
☐	Writeable/WriteableInteger1/A...	Alarm	3/31/16 9:28 AM	admin	3/31/16 9:36 AM	Active, Acknowledged	Critical
☐	Alarms/ProdSupAttention/Alarm	Alarm	3/31/16 9:28 AM	admin	3/31/16 9:36 AM	Active, Acknowledged	High

Next...

- Alarm Status Tags

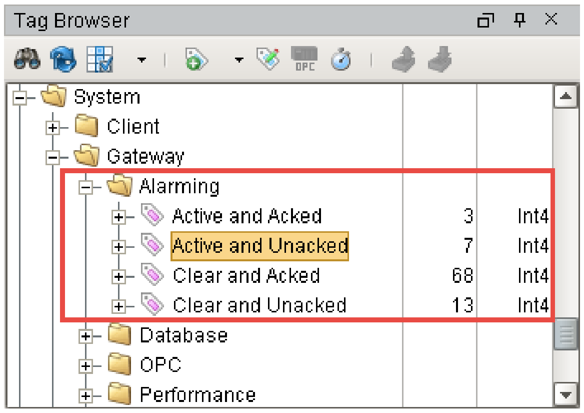
Alarm Status - Tag History and Alarm History

Watch the Video

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 on the Navigation window. The four system tag states are: Active and Unacknowledged, Active and Unacknowledged, Clear and Acknowledged, and Clear and Unacknowledged.

In the **Tag Browser** of the Designer, scroll down to the **System > Gateway > Alarming** folder. You can see all 4 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**.

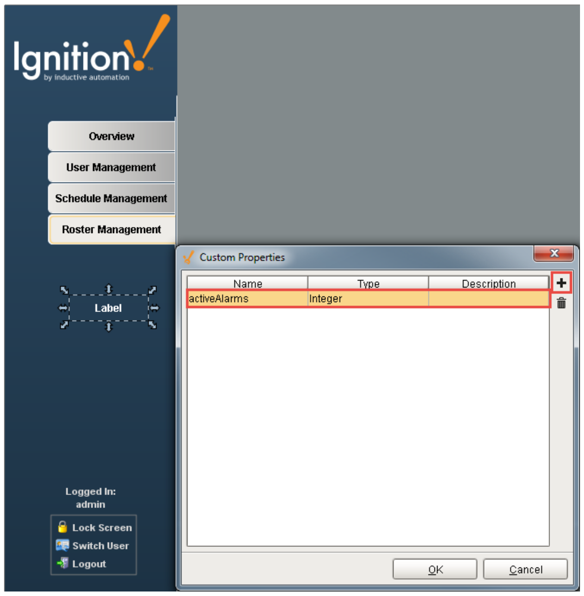


Active and Acked	3	Int4
Active and Unacked	7	Int4
Clear and Acked	68	Int4
Clear and Unacked	13	Int4

The **Active and Unacknowledged** tag is a key tag for operators so they know how many alarms are currently active that they have to respond too.

Let's setup an Active Alarm on the Navigation window using a Label component for the **Active and UnAcked** tag showing how many alarms are currently active and unacknowledged.

1. Open the **Navigation window**, drag in a **Label component** and place it under the tab strip.
2. Right click on **Customizers > Custom Properties**.
3. Create a new property and name it **'activeAlarms'** with an **'Integer'** data type.
4. Click **OK.**



Name	Type	Description
activeAlarms	Integer	

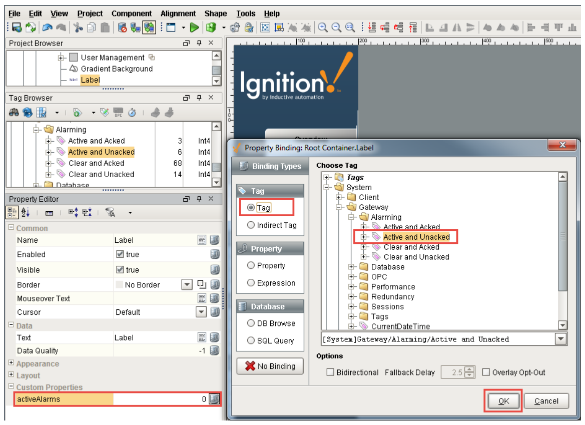
5. Bind the new custom property **'activeAlarm'** to the following tag by navigating the **Tag Browser**.

   "[System]Gateway/Alarming/Active and Unacked"

6. Click **OK**.

Alarm Status Tags

Watch the Video

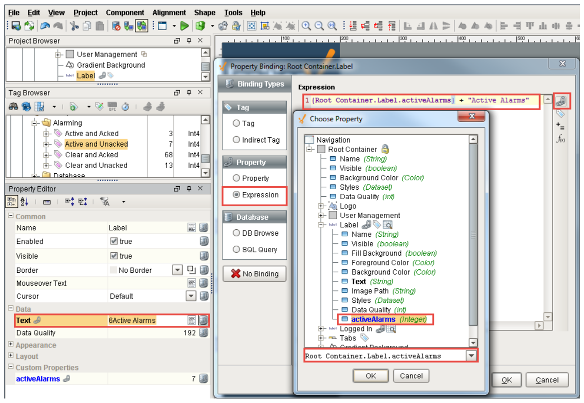


[System]Gateway/Alarming/Active and Unacked

7. Bind the **Text property** of the **Label** to the **Expression** by navigating to the path using the **Link icon** in the upper right corner of the Expression window. Combine the expression with the literal string value **"Active Alarms."**

Combine Expression with Literal String Value

```
{Root Container.Label.activeAlarms} + " Active Alarms"
```



{Root Container.Label.activeAlarms} + "Active Alarms"

Root Container.Label.activeAlarms

8. You want to make sure the Active Alarms label stands out on the Navigation window, so let's change the color to something bright and make it blink. Right click on the **Label** and select **Customizers > Style Customizer.**

9. For the **Driving Property**, select **activeAlarm,** and for the **Styled Properties**, select **Background Color**, **Foreground Color**, **Border** and **Visible**, and click on the green arrow pointing to the right and add them to the **Used Properties** list.

10. Under the **Styles area**, click on the green arrow at the bottom 2 times to add 2 **Values**. Click the **Expand buttons** to open and customize the style for the Active Alarm Label. Set the following values for **Styles**:

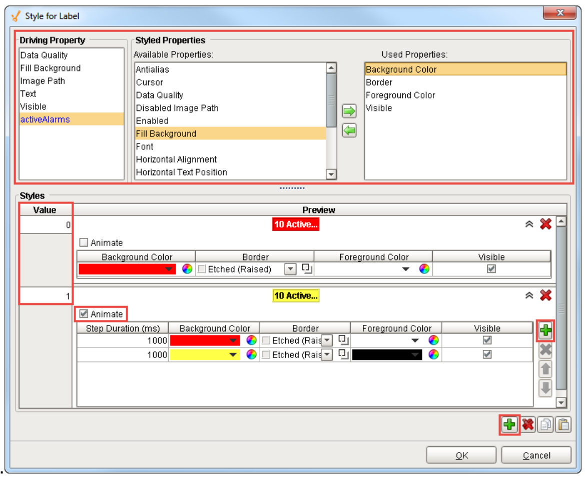
    Value 0
    Background Color: **Red**
    Border: **Etched (Raised)**
    Foreground Color: **White**
    Visible: **check the box**

    Value 1
    Click the **Animate** box. Use the green plus icon to add 2 **animation steps**. Change the values to

    Step 1 Duration: **1000**
    Background Color: **Red**
    Border: **Etched (Raised)**
    Foreground Color: **White**

Visible: **check the box**

Step 2 Duration: **1000**
Background Color: **Red**
Border: **Etched (Raised)**
Foreground Color: **White**
Visible: **check the box**


Style for Label

Driving Property: Data Quality, Fill Background, Image Path, Text, Visible, activeAlarms

Styled Properties — Available Properties: Antialias, Cursor, Data Quality, Disabled Image Path, Enabled, Fill Background, Font, Horizontal Alignment, Horizontal Text Position

Used Properties: Background Color, Border, Foreground Color, Visible

Styles

Value	Preview
0	10 Active...
1	10 Active...

Value 0 — Animate: unchecked

Background Color	Border	Foreground Color	Visible
🟥	Etched (Raised)		✓

Value 1 — Animate: ✓

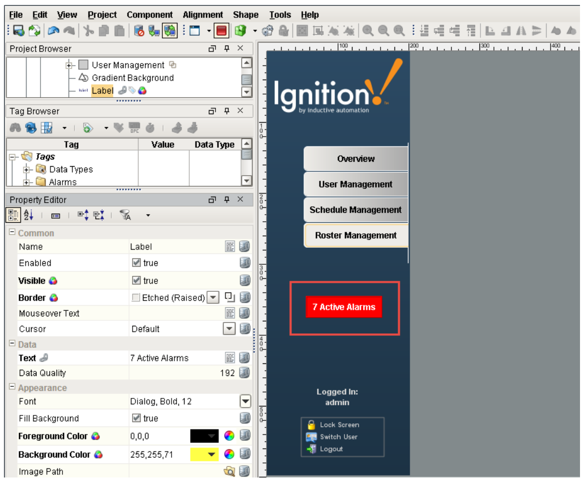
Step Duration (ms)	Background Color	Border	Foreground Color	Visible
1000	🟥	Etched (Rais		✓
1000	🟨	Etched (Rais	⬛	✓

OK Cancel

.

Click **OK**.

Now you have a label on the Navigation window that shows the number of Active Alarms. This example shows 7 Active Alarms. The Active Alarm label is easily visible to the operators and easy to create.



Property Editor

Common	
Name	Label
Enabled	✓ true
Visible	✓ true
Border	Etched (Raised)
Mouseover Text	
Cursor	Default
Data	
Text	7 Active Alarms
Data Quality	192
Appearance	
Font	Dialog, Bold, 12
Fill Background	✓ true
Foreground Color	0,0,0
Background Color	255,255,71
Image Path	

Next...

Alarm Journal

An alarm journal stores historical information about alarms in a database. It can store basic data about alarms that have occurred, such as their source and timestamp, along with 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 table and different alarms in a different table.

In this section ...

Create 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 their source and timestamp, along with 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 that can be accessed through scripting functions and direct database queries.

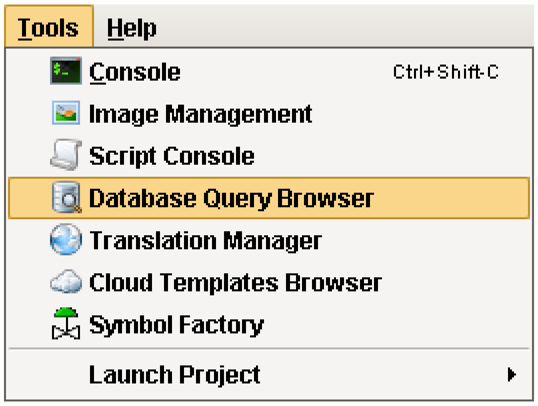
Create Alarm Journal Profile

Watch the Video

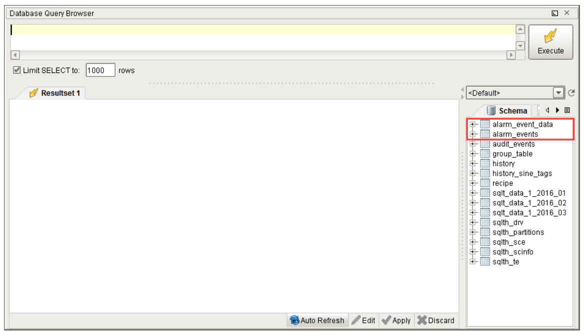
To create a new Alarm Journal profile

1. Go to the **Configure** section of the Gateway.
2. Choose **Alarming > Journal** from the menu on the left.
   The Alarm Journal Profiles page is displayed.
3. Look for the orange arrow and click on **Create New Alarm Journal Profile...**.
   The New Alarm Journal Profiles page is displayed.
4. Fill in the fields and click the **Create Journal** button at the bottom of the page.
   Once completed, the tables will be created for you once an alarm event occurs. It's a good idea to trigger a test alarm and verify that Ignition automatically created the tables in the database after your Alarm Journal Profile was created.

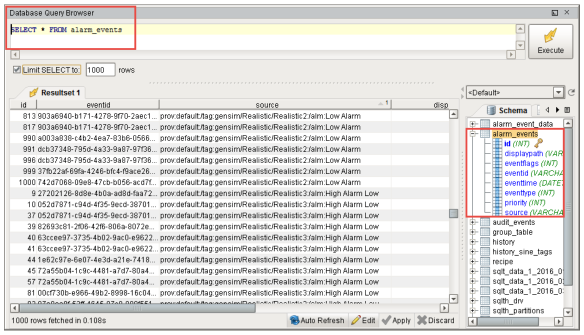
   You can easily verify the Alarm Journal tables from the Ignition **Designer**. Open it and go to the menu bar and select **Tools > Database Query Browser.**



The 2 tables that Ignition automatically creates are the alarm_events and alarm_event_data. Alarm events consist of 2 main types of data: the primary information about the alarm, such as transition state, time, etc., and the event data. If you don't see these 2 tables in your database, you need to make an alarm occur after the Alarm Journal Profile is created. Make the alarm Active, take it back to Clear, Acknowledge it, and you will start seeing information come into these tables.



Double click on the **alarm_events**, to see all the events that occurred when the alarms are active, when they're cleared, and when they are acknowledged.


Database Query Browser

```
SELECT * FROM alarm_events
```

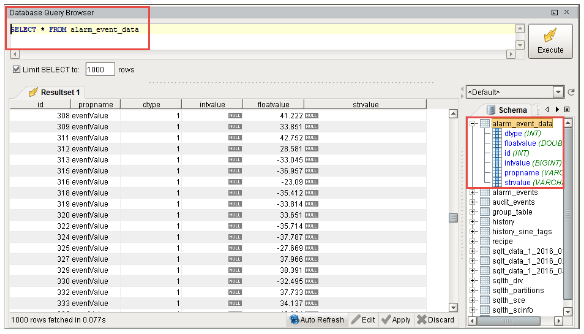
Limit SELECT to: 1000 rows

Resultset 1

id	eventid	source	disp
813	903a6940-b171-4278-9f70-2aec1...	prov:default:/tag:gensim/Realistic/Realistic2:/alm:Low Alarm	
817	903a6940-b171-4278-9f70-2aec1...	prov:default:/tag:gensim/Realistic/Realistic2:/alm:Low Alarm	
990	a003a838-c4b2-4ea7-83b6-0566...	prov:default:/tag:gensim/Realistic/Realistic2:/alm:Low Alarm	
991	dcb37348-795d-4a33-9a87-97f36...	prov:default:/tag:gensim/Realistic/Realistic2:/alm:Low Alarm	
996	dcb37348-795d-4a33-9a87-97f36...	prov:default:/tag:gensim/Realistic/Realistic2:/alm:Low Alarm	
999	37fb22af-69fa-4246-bfc4-f9ace26...	prov:default:/tag:gensim/Realistic/Realistic2:/alm:Low Alarm	
1000	742d7068-09e8-47cb-b056-acd7f...	prov:default:/tag:gensim/Realistic/Realistic2:/alm:Low Alarm	
9	27202126-8d8e-4b0a-ad8d-faa72...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
10	052d7871-c94d-4f35-9ecd-38701...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
37	052d7871-c94d-4f35-9ecd-38701...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
39	82693c81-2f06-42f6-806a-8072e...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
40	63ccee97-3735-4b02-9ac0-e9622...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
41	63ccee97-3735-4b02-9ac0-e9622...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
44	1e62c97e-6e07-4e3d-a21e-7418...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
45	72a55b04-1c9c-4481-a7d7-80a4...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
57	72a55b04-1c9c-4481-a7d7-80a4...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	
81	00cf730b-e966-49b2-8998-16c04...	prov:default:/tag:gensim/Realistic/Realistic3:/alm:High Alarm Low	

1000 rows fetched in 0.108s

The **alarm_event_data** is all the associated data that is associated with each alarm.


Database Query Browser

```
SELECT * FROM alarm_event_data
```

Limit SELECT to: 1000 rows

Resultset 1

id	propname	dtype	intvalue	floatvalue	strvalue
308	eventValue	1	NULL	41.222	NULL
309	eventValue	1	NULL	33.851	NULL
311	eventValue	1	NULL	42.752	NULL
312	eventValue	1	NULL	28.581	NULL
313	eventValue	1	NULL	-33.045	NULL
315	eventValue	1	NULL	-36.957	NULL
316	eventValue	1	NULL	-23.09	NULL
318	eventValue	1	NULL	-35.412	NULL
319	eventValue	1	NULL	-33.814	NULL
320	eventValue	1	NULL	33.651	NULL
322	eventValue	1	NULL	-35.714	NULL
324	eventValue	1	NULL	-37.787	NULL
325	eventValue	1	NULL	-27.669	NULL
327	eventValue	1	NULL	37.966	NULL
329	eventValue	1	NULL	38.391	NULL
330	eventValue	1	NULL	-32.495	NULL
332	eventValue	1	NULL	37.733	NULL
333	eventValue	1	NULL	34.137	NULL

1000 rows fetched in 0.077s

You will want to look into the Database Query Browser to simply verify that the information is there. Once the information is there, close the Database Query Browser.

Note: if the tables were not created, check the Gateway console page for any errors.

Journal Options

Main	
Name	The default name is Journal.
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	If selected, will store the values of dynamically bound alarm configuration properties. The value of these properties can change at any

Config	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	
Filtering Settings	The minimum priority and store shelved events options filter what time of alarm events are stored in the journal. "Shelved" events are alarms that have been temporarily silenced by operators. Though they are not displayed to users, these events continue to be generated, and can be stored if the journal settings permit it. When stored, they will have a special flag indicating that they were shelved at the time.
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	Default is not enabled. If selected, data will be deleted after the specified time period as set by the Prune Age and Units below. 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 default is 1.
Prune Age Units	The default is YEAR. You can choose the unit to be MS, SEC, MIN, HOUR, DAY, WEEK, MONTH, and YEAR.
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 default is `alarm_events`. The table name for the core event table.
Event Data Table Name	The default is `alarm_event_data`. The table name for event data associated with alarms.

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 that this individual event is related to. Each `alarm event` (one particular `active/clear/ack` cycle of a defined alarm) receives a unique `id` in order to distinguish it from other events from the same source.
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 - See below for more information 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.
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.

About System Events

System events are stored in the journal to record actions that aren't directly related to a particular alarm.

Currently the following events are stored in the journal:

- System Startup
- System Shutdown
- Alarms Shelved
- Alarms Un-shelved

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.

Next...

- Alarm Journal Component

Alarm Journal Component

The **Alarm Journal Table** component provides a built-in view to explore alarm history that has been stored in an Alarm Journal.

Before viewing alarm history, you must first tell Ignition to log alarms in your database by creating your Alarm Journal Profile.

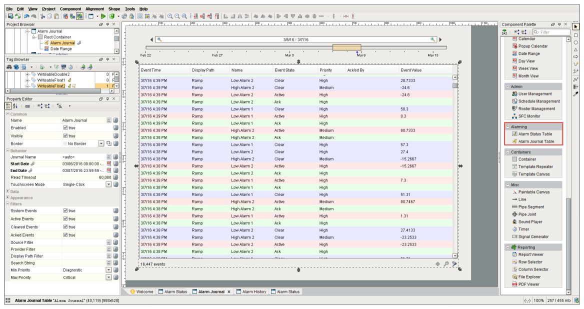
Additional information on the Alarm Journal Table component can be found on the Alarm Journal Table page in the appendix.

Alarm Journal Component

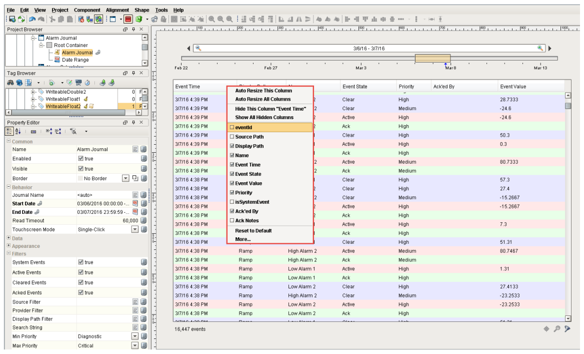
Watch the Video

Alarm Journal Component

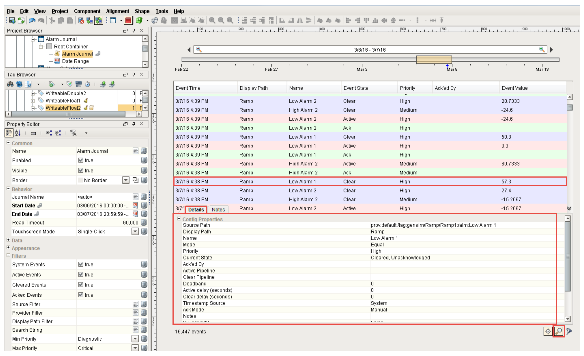
There is a pre-built component in Ignition called the **Alarm Journal Table** that allows the database to see this alarm history. In the **Component Palette**, under Alarming, drag the **Alarm Journal Table** component to your window and you'll see the history of all the alarms that occurred.



By default, the Alarm Journal Table does not filter out any of the alarms. The Alarm Journal Table shows you all the alarms stored in the default Alarm Journal. There is no date filtering and no filtering on which profile that you want to pull it from. The alarms are color coded so you know what each of the states represent. You can right click in the header to hide or show columns. You can also move columns around by clicking them and dragging. You can sort by a column by left-clicking it, or sort by multiple columns by holding the **Ctrl** key while clicking the column headers. You will see small numbers to indicate the sort order and direction (ascending vs descending).



You can click on individual alarms, then click the buttons in the lower right corner of the component to view the **Details** as well as the **Notes** to see all the information that is associated with the individual alarm.



Common	
Name	Alarm Journal
Enabled	☑ true
Visible	☑ true
Border	No Border
Behavior	
Journal Name	<auto>
Start Date	03/06/2016 00:00:00 -...
End Date	03/07/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click
Data	
Appearance	
Filters	
System Events	☑ true
Active Events	☑ true
Cleared Events	☑ true
Acked Events	☑ true
Source Filter	
Provider Filter	
Display Path Filter	
Search String	
Min Priority	Diagnostic
Max Priority	Critical

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/7/16 4:39 PM	Ramp	Low Alarm 2	Clear	High		28.7333
3/7/16 4:39 PM	Ramp	High Alarm 2	Clear	Medium		-24.6
3/7/16 4:39 PM	Ramp	Low Alarm 2	Active	High		-24.6
3/7/16 4:39 PM	Ramp	Low Alarm 2	Ack	High		
3/7/16 4:39 PM	Ramp	Low Alarm 1	Clear	High		50.3
3/7/16 4:39 PM	Ramp	Low Alarm 1	Active	High		0.3
3/7/16 4:39 PM	Ramp	Low Alarm 1	Ack	High		
3/7/16 4:38 PM	Ramp	High Alarm 2	Active	Medium		80.7333
3/7/16 4:38 PM	Ramp	High Alarm 2	Ack	Medium		
3/7/16 4:38 PM	Ramp	Low Alarm 1	Clear	High		57.3
3/7/16 4:38 PM	Ramp	Low Alarm 2	Clear	High		27.4
3/7/16 4:38 PM	Ramp	High Alarm 2	Clear	Medium		-15.2667
3/7'	Ramp	Low Alarm 2	Active	High		-15.2667

Config Properties	
Source Path	prov:default:/tag:gensim/Ramp/Ramp1:/alm:Low Alarm 1
Display Path	Ramp
Name	Low Alarm 1
Mode	Equal
Priority	High
Current State	Cleared, Unacknowledged
Ack'ed By	
Active Pipeline	
Clear Pipeline	
Deadband	0
Active delay (seconds)	0
Clear delay (seconds)	0
Timestamp Source	System
Ack Mode	Manual
Notes	

16,447 events

Next...

- Alarm Journal - Filter on Date Range
- Alarm Journal

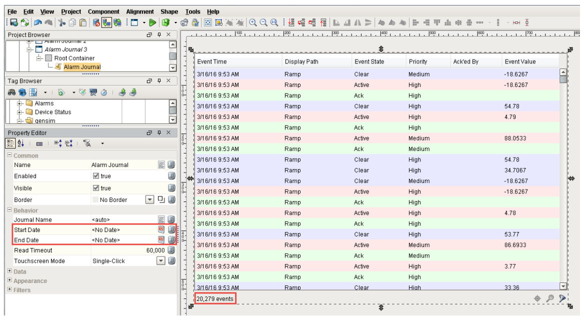
Alarm Journal - Filter on Date Range

You can filter the alarm history data stored in **Alarm Journal Table** component for all alarms within a specific start date and end date.

When you first put the Alarm Journal Table component on a window, Ignition shows you the last 8 hours of alarm events that are in the history table of the database. By default, there is no filtering of alarms. You typically want to filter for alarm events in a specific time period rather than scroll through a list of events. You don't want operators to see everything, but to choose a start date and end date so they can see the alarm history for a specific time period. This is very easy to setup on the Date Range component.

Alarm Journal - Filter on Date Range

Watch the Video

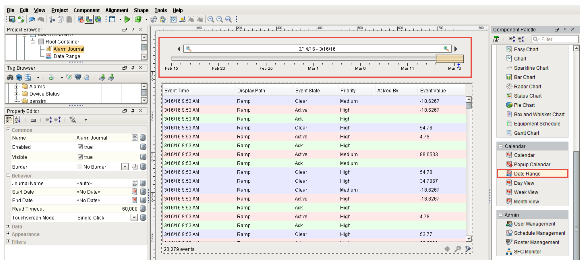


Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/16/16 9:53 AM	Ramp	Clear	Medium		-18.6267
3/16/16 9:53 AM	Ramp	Active	High		-18.6267
3/16/16 9:53 AM	Ramp	Ack	High		
3/16/16 9:53 AM	Ramp	Clear	High		54.78
3/16/16 9:53 AM	Ramp	Active	High		4.79
3/16/16 9:53 AM	Ramp	Ack	High		
3/16/16 9:53 AM	Ramp	Active	Medium		88.0533
3/16/16 9:53 AM	Ramp	Ack	Medium		
3/16/16 9:53 AM	Ramp	Clear	High		54.78
3/16/16 9:53 AM	Ramp	Clear	High		34.7067
3/16/16 9:53 AM	Ramp	Clear	Medium		-18.6267
3/16/16 9:53 AM	Ramp	Active	High		-18.6267
3/16/16 9:53 AM	Ramp	Ack	High		
3/16/16 9:53 AM	Ramp	Active	High		4.78
3/16/16 9:53 AM	Ramp	Ack	High		
3/16/16 9:53 AM	Ramp	Clear	High		53.77
3/16/16 9:53 AM	Ramp	Active	Medium		86.6933
3/16/16 9:53 AM	Ramp	Ack	Medium		
3/16/16 9:53 AM	Ramp	Active	High		3.77
3/16/16 9:53 AM	Ramp	Ack	High		
3/16/16 9:53 AM	Ramp	Clear	High		33.36

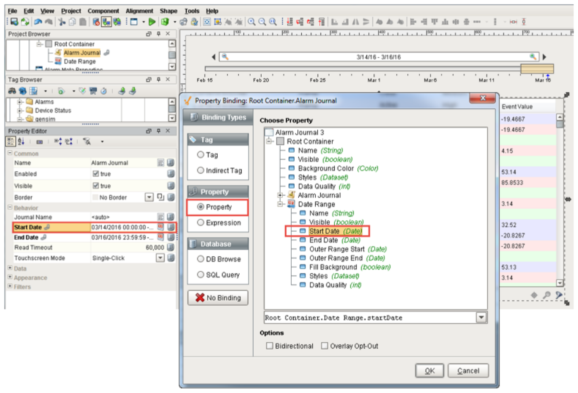
20,279 events

Name: Alarm Journal
Enabled: true
Visible: true
Border: No Border
Journal Name: <auto>
Start Date: <No Date>
End Date: <No Date>
Read Timeout: 60,000
Touchscreen Mode: Single-Click

1. Ignition offers several types of date pickers under the Calendar tab of the Component Palette. This example uses the **Date Range component**. The Start Date and End Date are all built into this one component so you can select the Start Date and End Date all in one place. Drag the **Date Range component** from the Calendar tab to your window above the Alarm Journal Table.



2. To link the Alarm Journal Table component to the Date Range component, first select the **Alarm Journal Table component**, click the **Start Date binding icon**, bind it to a **Property,** and select the **Date Range's Start Date**. Click **OK**.

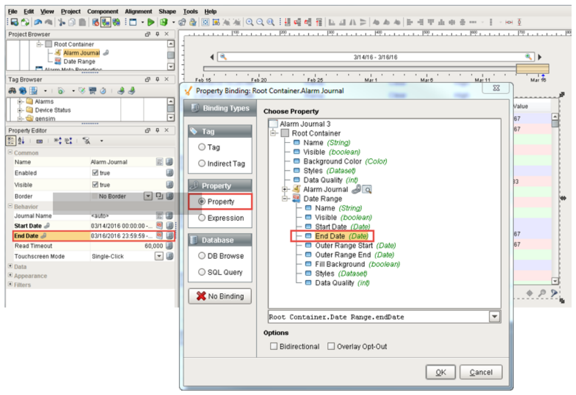


Property Binding: Root Container.Alarm Journal

Start Date: 03/14/2016 00:00:00
End Date: 03/16/2016 23:59:59

```
Root Container.Date Range.startDate
```

3. Now, do the same for the **Alarm Journal Table's End Date.** Click the **End Date binding icon**, bind it to a **Property**, and select the **Date Range's End Date**. Click. **OK**.


Property Binding: Root Container.Alarm Journal

Binding Types: Tag, Indirect Tag, Property (selected), Expression, DB Browse, SQL Query, No Binding

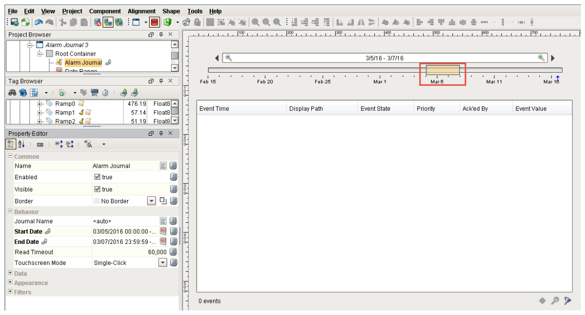
Choose Property: Alarm Journal 3 > Root Container > Date Range > End Date (Date)

```
Root Container.Date Range.endDate
```

Options: Bidirectional, Overlay Opt-Out

4. By linking the Alarm Journal Table and the Date Range components together, as soon as the operator changes the Start Date and End Date on the Date Range, Ignition automatically updates the Alarm Journal Table and all the alarm events within that time period will be displayed.

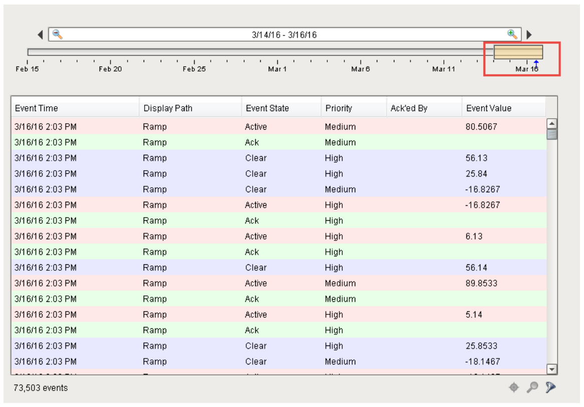
5. Click on **Preview Mode** and move the slider to the left.   You will notice there are no events in that time period.


3/5/16 - 3/7/16

Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value

0 events

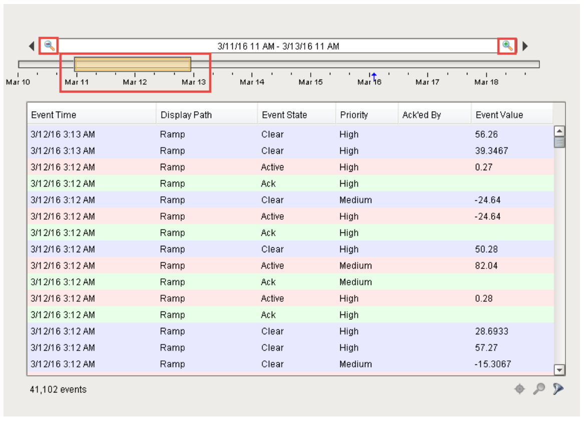
Now, move the slider to the right, and you will see all the alarm events within that time period.


3/14/16 - 3/16/16

Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/16/16 2:03 PM	Ramp	Active	Medium		80.5067
3/16/16 2:03 PM	Ramp	Ack	Medium		
3/16/16 2:03 PM	Ramp	Clear	High		56.13
3/16/16 2:03 PM	Ramp	Clear	High		25.84
3/16/16 2:03 PM	Ramp	Clear	Medium		-16.8267
3/16/16 2:03 PM	Ramp	Active	High		-16.8267
3/16/16 2:03 PM	Ramp	Ack	High		
3/16/16 2:03 PM	Ramp	Active	High		6.13
3/16/16 2:03 PM	Ramp	Ack	High		
3/16/16 2:03 PM	Ramp	Clear	High		56.14
3/16/16 2:03 PM	Ramp	Active	Medium		89.8533
3/16/16 2:03 PM	Ramp	Ack	Medium		
3/16/16 2:03 PM	Ramp	Active	High		5.14
3/16/16 2:03 PM	Ramp	Ack	High		
3/16/16 2:03 PM	Ramp	Clear	High		25.8533
3/16/16 2:03 PM	Ramp	Clear	Medium		-18.1467

73,503 events

6. You can also zoom in or out on specific time periods to see the alarm events that exist in the history table.



Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/12/16 3:13 AM	Ramp	Clear	High		56.26
3/12/16 3:13 AM	Ramp	Clear	High		39.3467
3/12/16 3:12 AM	Ramp	Active	High		0.27
3/12/16 3:12 AM	Ramp	Ack	High		
3/12/16 3:12 AM	Ramp	Clear	Medium		-24.64
3/12/16 3:12 AM	Ramp	Active	High		-24.64
3/12/16 3:12 AM	Ramp	Ack	High		
3/12/16 3:12 AM	Ramp	Clear	High		50.28
3/12/16 3:12 AM	Ramp	Active	Medium		82.04
3/12/16 3:12 AM	Ramp	Ack	Medium		
3/12/16 3:12 AM	Ramp	Active	High		0.28
3/12/16 3:12 AM	Ramp	Ack	High		
3/12/16 3:12 AM	Ramp	Clear	High		28.6933
3/12/16 3:12 AM	Ramp	Clear	High		57.27
3/12/16 3:12 AM	Ramp	Clear	Medium		-15.3067

41,102 events

It is very easy to filter alarm events based on time. It's important to note that the Alarm Journal Table does not pull data unless the start or end date properties are changed and when the window is first opened. So if you want to see new events come in to your table, you have to change the Date Range. Use the slider to move back and forward to start seeing new events come into the table.

Use the Date Range slider and mouse to see new alarm events

To see new alarm events, move the slider back and release the mouse, then drag the slider forward to the current date/time and release the mouse.

Next...

- Alarm Journal - General Filtering

Alarm Journal - General Filtering

Alarm Journal Profile

To view alarm history, an Alarm Journal Profile must be created first before logging alarms to the database.

Alarm Journal - General Filtering

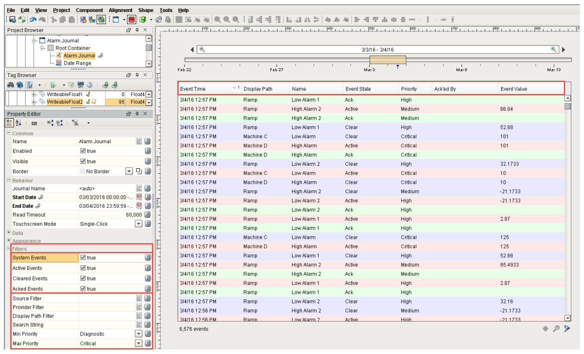
Watch the Video

Ignition provides a host of filters to view alarm history data. Not only can you filter the Alarm Journal Table on a date range to see alarm history data within a specific time period, but there are also a number of other filters you can use as well.

In the Alarm Journal Table component you can filter on alarms in 2 different areas on the table; filter properties in the Property Editor and the filter icons located at the bottom of the Alarm Journal Table component. In this section, we will focus on filtering alarm data using the filter properties in the Property Editor.

In the Property Editor, you will notice that you have 4 checked boxes for different type of alarm events that you can filter on: **System Events**, **Active Events**, **Cleared Events** and **Acked Events**. For example, if you uncheck the Cleared Events and Acked Events, you will only show the Active and the System events. If you want to show only Active events, check Active Events and leave the other events unchecked. It's easy to configure the event filters on the type of alarms you want to see. You can also filter on **Source Path**, **Priority**, **Display Path** and even create your own **Search String**.

This example shows a basic filter displaying all the alarm events. Each color represents a different alarm state. You can also right click in the column header to show or hide columns, reorganize the columns, and resize the columns.

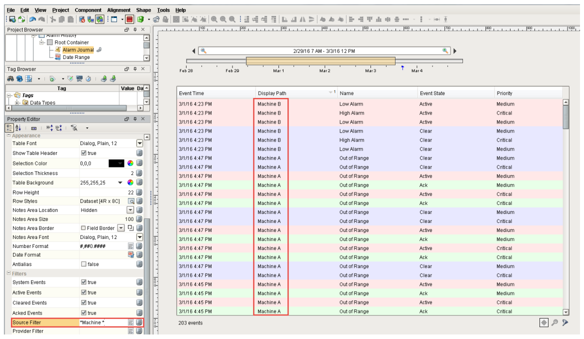


Common	
Name	Alarm Journal
Enabled	☑ true
Visible	☑ true
Border	No Border
Behavior	
Journal Name	<auto>
Start Date	03/03/2016 00:00:00 -...
End Date	03/04/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click
Data	
Appearance	
Filters	
System Events	☑ true
Active Events	☑ true
Cleared Events	☑ true
Acked Events	☑ true
Source Filter	
Provider Filter	
Display Path Filter	
Search String	
Min Priority	Diagnostic
Max Priority	Critical

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/4/16 12:57 PM	Ramp	Low Alarm 1	Ack	High		
3/4/16 12:57 PM	Ramp	High Alarm 2	Active	Medium		86.84
3/4/16 12:57 PM	Ramp	High Alarm 2	Ack	Medium		
3/4/16 12:57 PM	Ramp	Low Alarm 1	Clear	High		52.88
3/4/16 12:57 PM	Machine C	Low Alarm	Clear	Critical		101
3/4/16 12:57 PM	Machine D	High Alarm	Active	Critical		101
3/4/16 12:57 PM	Machine D	High Alarm	Ack	Critical		
3/4/16 12:57 PM	Ramp	Low Alarm 2	Clear	High		32.1733
3/4/16 12:57 PM	Machine C	Low Alarm	Active	Critical		10
3/4/16 12:57 PM	Machine D	High Alarm	Clear	Critical		10
3/4/16 12:57 PM	Ramp	High Alarm 2	Clear	Medium		-21.1733
3/4/16 12:57 PM	Ramp	Low Alarm 2	Active	High		-21.1733
3/4/16 12:57 PM	Ramp	Low Alarm 2	Ack	High		
3/4/16 12:57 PM	Ramp	Low Alarm 1	Active	High		2.87
3/4/16 12:57 PM	Ramp	Low Alarm 1	Ack	High		
3/4/16 12:57 PM	Machine C	Low Alarm	Clear	Critical		125
3/4/16 12:57 PM	Machine D	High Alarm	Active	Critical		125
3/4/16 12:57 PM	Ramp	Low Alarm 1	Clear	High		52.88
3/4/16 12:57 PM	Ramp	High Alarm 2	Active	Medium		85.4933
3/4/16 12:57 PM	Ramp	High Alarm 2	Ack	Medium		
3/4/16 12:57 PM	Ramp	Low Alarm 1	Active	High		2.87
3/4/16 12:57 PM	Ramp	Low Alarm 1	Ack	High		
3/4/16 12:57 PM	Ramp	Low Alarm 2	Clear	High		32.16
3/4/16 12:56 PM	Ramp	High Alarm 2	Clear	Medium		-21.1733
3/4/16 12:56 PM	Ramp	Low Alarm 2	Active	High		-21.1733

6,576 events

There is also a way to filter alarms based on a tag path, tag provider and alarm name. If you configure alarms in particular folders in Ignition, you can use the **Source Filter** to find alarms by tag path. In this example, ***Machine*** was entered using wildcards in the Source Path field to find all the Machine alarms in all the Machine folders.

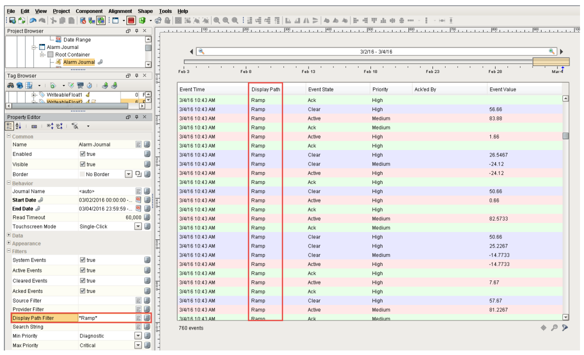


Appearance	
Table Font	Dialog, Plain, 12
Show Table Header	☑ true
Selection Color	0,0,0
Selection Thickness	2
Table Background	255,255,25
Row Height	22
Row Styles	Dataset [4R x 8C]
Notes Area Location	Hidden
Notes Area Size	100
Notes Area Border	☐ Field Border
Notes Area Font	Dialog, Plain, 12
Number Format	#,##0.####
Date Format	
Antialias	☐ false
Filters	
System Events	☑ true
Active Events	☑ true
Cleared Events	☑ true
Acked Events	☑ true
Source Filter	*Machine*
Provider Filter	

Event Time	Display Path	Name	Event State	Priority
3/1/16 4:23 PM	Machine B	Low Alarm	Active	Medium
3/1/16 4:23 PM	Machine B	High Alarm	Active	Critical
3/1/16 4:23 PM	Machine B	Low Alarm	Active	Medium
3/1/16 4:23 PM	Machine B	Low Alarm	Clear	Medium
3/1/16 4:23 PM	Machine B	High Alarm	Clear	Critical
3/1/16 4:23 PM	Machine B	Low Alarm	Clear	Medium
3/1/16 4:47 PM	Machine A	Out of Range	Clear	Medium
3/1/16 4:47 PM	Machine A	Out of Range	Clear	Critical
3/1/16 4:47 PM	Machine A	Out of Range	Active	Medium
3/1/16 4:47 PM	Machine A	Out of Range	Ack	Medium
3/1/16 4:47 PM	Machine A	Out of Range	Active	Critical
3/1/16 4:47 PM	Machine A	Out of Range	Ack	Critical
3/1/16 4:47 PM	Machine A	Out of Range	Clear	Medium
3/1/16 4:47 PM	Machine A	Out of Range	Clear	Critical
3/1/16 4:47 PM	Machine A	Out of Range	Active	Medium
3/1/16 4:47 PM	Machine A	Out of Range	Ack	Medium
3/1/16 4:47 PM	Machine A	Out of Range	Active	Critical
3/1/16 4:47 PM	Machine A	Out of Range	Ack	Critical
3/1/16 4:47 PM	Machine A	Out of Range	Clear	Medium
3/1/16 4:47 PM	Machine A	Out of Range	Clear	Critical
3/1/16 4:45 PM	Machine A	Out of Range	Active	Medium
3/1/16 4:45 PM	Machine A	Out of Range	Ack	Medium
3/1/16 4:45 PM	Machine A	Out of Range	Active	Critical
3/1/16 4:45 PM	Machine A	Out of Range	Ack	Critical

203 events

You can also filter on alarms by name using the **Display Path Filter** which was setup in the alarm configuration. In this example, ***Ramp*** was entered in the Display Path field to find all the alarms in the system with a 'Ramp' name.



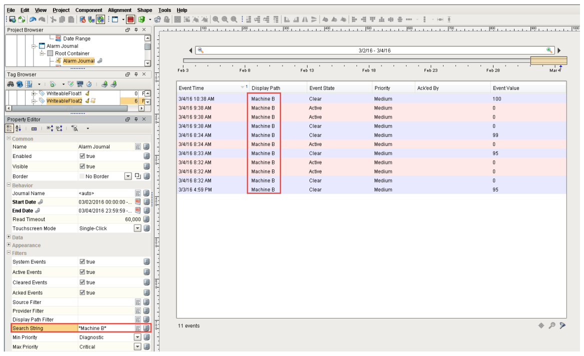
Name	Alarm Journal
Enabled	true
Visible	true
Border	No Border
Journal Name	<auto>
Start Date	03/02/2016 00:00:00 -...
End Date	03/04/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click
System Events	true
Active Events	true
Cleared Events	true
Acked Events	true
Source Filter	
Provider Filter	
Display Path Filter	*Ramp*
Search String	
Min Priority	Diagnostic
Max Priority	Critical

3/2/16 - 3/4/16

Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/4/16 10:43 AM	Ramp	Ack	High		
3/4/16 10:43 AM	Ramp	Clear	High		56.66
3/4/16 10:43 AM	Ramp	Active	Medium		83.88
3/4/16 10:43 AM	Ramp	Ack	Medium		
3/4/16 10:43 AM	Ramp	Active	High		1.66
3/4/16 10:43 AM	Ramp	Ack	High		
3/4/16 10:43 AM	Ramp	Clear	High		26.5467
3/4/16 10:43 AM	Ramp	Clear	Medium		-24.12
3/4/16 10:43 AM	Ramp	Active	High		-24.12
3/4/16 10:43 AM	Ramp	Ack	High		
3/4/16 10:43 AM	Ramp	Clear	High		50.66
3/4/16 10:43 AM	Ramp	Active	High		0.66
3/4/16 10:43 AM	Ramp	Ack	High		
3/4/16 10:43 AM	Ramp	Active	Medium		82.5733
3/4/16 10:43 AM	Ramp	Ack	Medium		
3/4/16 10:43 AM	Ramp	Clear	High		50.66
3/4/16 10:43 AM	Ramp	Clear	High		25.2267
3/4/16 10:43 AM	Ramp	Clear	Medium		-14.7733
3/4/16 10:43 AM	Ramp	Active	High		-14.7733
3/4/16 10:43 AM	Ramp	Ack	High		
3/4/16 10:43 AM	Ramp	Active	High		7.67
3/4/16 10:43 AM	Ramp	Ack	High		
3/4/16 10:43 AM	Ramp	Clear	High		57.67
3/4/16 10:43 AM	Ramp	Active	Medium		81.2267
3/4/16 10:43 AM	Ramp	Ack	Medium		

760 events

You can also create your own **Search String** to find specific alarms, events and conditions. In this example, ***Machine B*** was entered in the Search String field to find all Machine B alarms in the system.



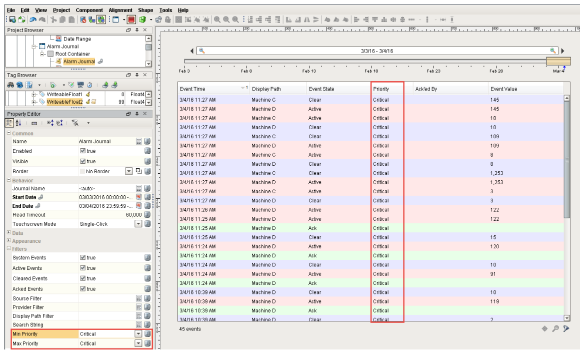
Name	Alarm Journal
Enabled	true
Visible	true
Border	No Border
Journal Name	<auto>
Start Date	03/02/2016 00:00:00 -...
End Date	03/04/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click
System Events	true
Active Events	true
Cleared Events	true
Acked Events	true
Source Filter	
Provider Filter	
Display Path Filter	
Search String	*Machine B*
Min Priority	Diagnostic
Max Priority	Critical

3/2/16 - 3/4/16

Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/4/16 10:38 AM	Machine B	Clear	Medium		100
3/4/16 9:38 AM	Machine B	Active	Medium		0
3/4/16 9:38 AM	Machine B	Active	Medium		0
3/4/16 9:38 AM	Machine B	Clear	Medium		0
3/4/16 8:34 AM	Machine B	Clear	Medium		99
3/4/16 8:34 AM	Machine B	Active	Medium		0
3/4/16 8:33 AM	Machine B	Clear	Medium		95
3/4/16 8:32 AM	Machine B	Active	Medium		0
3/4/16 8:32 AM	Machine B	Active	Medium		0
3/4/16 8:32 AM	Machine B	Clear	Medium		0
3/3/16 4:59 PM	Machine B	Clear	Medium		95

11 events

Lastly, you can also filter alarms based on **Priority**.   Typically, when you open the Alarm Journal Table, it displays every single alarm on all the priorities from Diagnostic to Critical. If all you want to do is display Critical alarms, select 'Critical' from the **Min Priority** field and 'Critical' from the **Max Priority** field.   Now, only the Critical alarms will be displayed in your table.



Name	Alarm Journal
Enabled	true
Visible	true
Border	No Border
Journal Name	<auto>
Start Date	03/03/2016 00:00:00 -...
End Date	03/04/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click
System Events	true
Active Events	true
Cleared Events	true
Acked Events	true
Source Filter	
Provider Filter	
Display Path Filter	
Search String	
Min Priority	Critical
Max Priority	Critical

3/3/16 - 3/4/16

Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/4/16 11:27 AM	Machine C	Clear	Critical		145
3/4/16 11:27 AM	Machine D	Active	Critical		145
3/4/16 11:27 AM	Machine C	Active	Critical		10
3/4/16 11:27 AM	Machine D	Clear	Critical		10
3/4/16 11:27 AM	Machine C	Clear	Critical		109
3/4/16 11:27 AM	Machine D	Active	Critical		109
3/4/16 11:27 AM	Machine C	Active	Critical		8
3/4/16 11:27 AM	Machine D	Clear	Critical		8
3/4/16 11:27 AM	Machine C	Clear	Critical		1,253
3/4/16 11:27 AM	Machine D	Active	Critical		1,253
3/4/16 11:27 AM	Machine C	Active	Critical		3
3/4/16 11:27 AM	Machine D	Clear	Critical		3
3/4/16 11:26 AM	Machine D	Active	Critical		122
3/4/16 11:25 AM	Machine D	Active	Critical		122
3/4/16 11:25 AM	Machine D	Ack	Critical		
3/4/16 11:25 AM	Machine D	Clear	Critical		15
3/4/16 11:24 AM	Machine D	Active	Critical		120
3/4/16 11:24 AM	Machine D	Ack	Critical		
3/4/16 11:24 AM	Machine D	Clear	Critical		10
3/4/16 11:24 AM	Machine D	Active	Critical		91
3/4/16 11:24 AM	Machine D	Ack	Critical		
3/4/16 10:39 AM	Machine D	Clear	Critical		10
3/4/16 10:39 AM	Machine D	Active	Critical		119
3/4/16 10:39 AM	Machine D	Ack	Critical		
3/4/16 10:39 AM	Machine D	Clear	Critical		2

45 events

All of the Alarm Journal Filter properties are bindable so you can design your windows and let the operator choose and filter the alarm data based on how they want to display it.

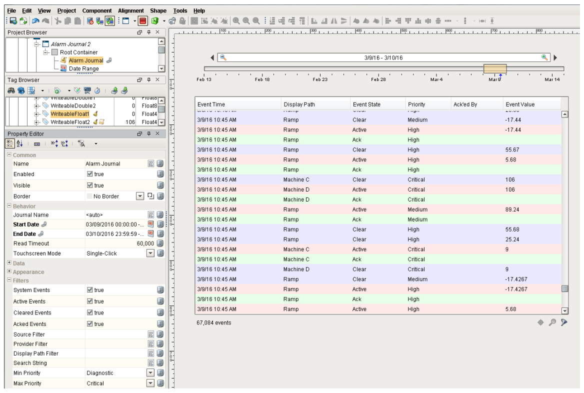
Next...

- Alarm Journal - Filter on Associated Data

Alarm Journal - Filter on Associated Data

You can filter the **Alarm Journal Table** component on an alarm's associated data to define, search, and display the exact data you are interested in from the alarm history data.

Once associated data is added to an alarm, you can easily filter for specific alarms in the Alarm Journal Component, as well as the Alarm Status Component and Alarm Pipelines. Typically, the Alarm Journal Table shows you the complete history of all your alarms within a specific time period using the Date Range component.   In this section, you will see how you can filter the Alarm Journal Table to find particular alarms based on associated data configured in an alarm.

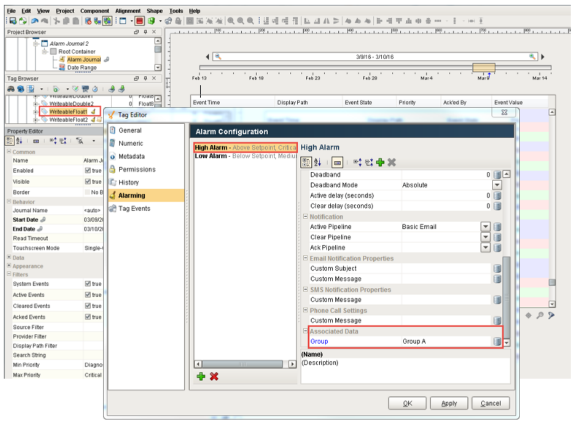


3/9/16 - 3/10/16

Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:45 AM	Ramp	Clear	Medium		-17.44
3/9/16 10:45 AM	Ramp	Active	High		-17.44
3/9/16 10:45 AM	Ramp	Ack	High		
3/9/16 10:45 AM	Ramp	Clear	High		55.67
3/9/16 10:45 AM	Ramp	Active	High		5.68
3/9/16 10:45 AM	Ramp	Ack	High		
3/9/16 10:45 AM	Machine C	Clear	Critical		106
3/9/16 10:45 AM	Machine D	Active	Critical		106
3/9/16 10:45 AM	Machine D	Ack	Critical		
3/9/16 10:45 AM	Ramp	Active	Medium		89.24
3/9/16 10:45 AM	Ramp	Ack	Medium		
3/9/16 10:45 AM	Ramp	Clear	High		55.68
3/9/16 10:45 AM	Ramp	Clear	High		25.24
3/9/16 10:45 AM	Machine C	Active	Critical		9
3/9/16 10:45 AM	Machine C	Ack	Critical		
3/9/16 10:45 AM	Machine D	Clear	Critical		9
3/9/16 10:45 AM	Ramp	Clear	Medium		-17.4267
3/9/16 10:45 AM	Ramp	Active	High		-17.4267
3/9/16 10:45 AM	Ramp	Ack	High		
3/9/16 10:45 AM	Ramp	Active	High		5.68

67,084 events

In this example, a **'WriteableFloat1**' tag was used and added to an alarm.   You can see that 2 alarms were configured for 'WriteableFloat1,' but only one has associated data. Associated data was added to the **High Alarm** called '**Group'** and was given a static value of **'Group A.'**   You can filter on what is shown in the alarm list based on the associated data. You can search for all the Group A's or whatever associated data that was added to your alarm.



Alarm Configuration

High Alarm

Deadband	0
Deadband Mode	Absolute
Active delay (seconds)	0
Clear delay (seconds)	0
Active Pipeline	Basic Email
Group	Group A

Once your associated data is configured in your alarm, the next thing you need to do is setup your script to filter for all the alarms in your history that have the associated data named 'Group' with a value of 'Group A'. Instead of using the properties in the Property Editor, go to the **Scripting area** and use an **Ext ension Function**.

- Right click on your Alarm Journal Table component and select **Scripting**.
- Under the **Extension Functions** folder, click **'filterAlarm.'**
- Check **Enabled** and enter the following code at the bottom of the 'filterAlarm' script.

'filterAlarm' on associated data

```
group = alarmEvent.get("Group")
if group == "Group A":
        return True
return False
```

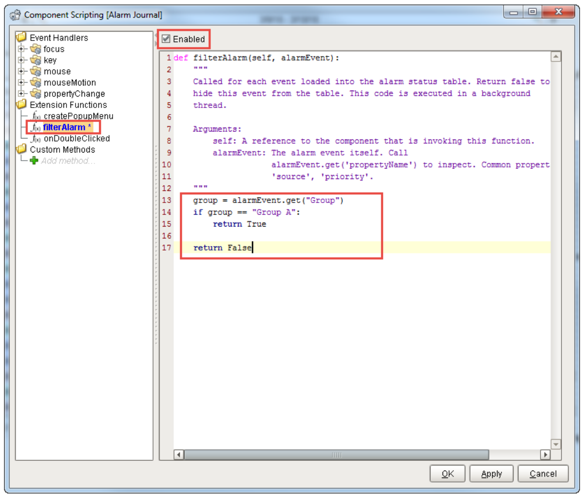
Alarm Journal - Filter on Associated Data

Watch the Video

What the ‘filterAlarm’ script is doing, is for every alarm matching ‘Group A,’ it will return ‘True,’ and show the alarm in the table. If the associated data does not match ‘Group A’, it will return ‘False,’ and the alarm will not show up in the table.

Scripting allows you to use the 'filterAlarm' function for associated data.   Essentially, this script is only going to show you the alarms matching the associated data.   You can add other filters as well depending on your requirements.

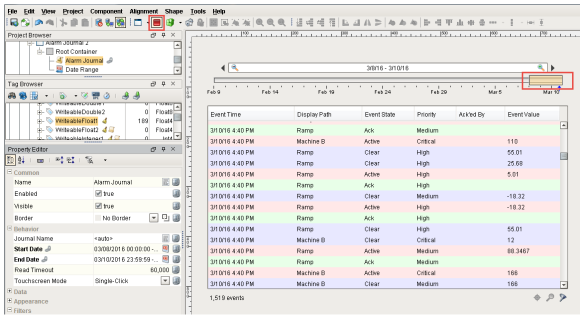
This example shows the updated ‘filterAlarm’ script.



```
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 propert
                    'source', 'priority'.
    """
    group = alarmEvent.get("Group")
    if group == "Group A":
        return True

    return False
```

Press **OK**. By default, the script is not going to automatically use your new filter until the data is updated. This happens automatically when a window is opened, but to trigger the associated data filter now, you can just change the **Start Date** or **End Date.** The easiest way is by moving the Date Range slider if you have one already bound.

Click **Preview Mode,** and move the **Date Range** slider to initiate the associated data filter.

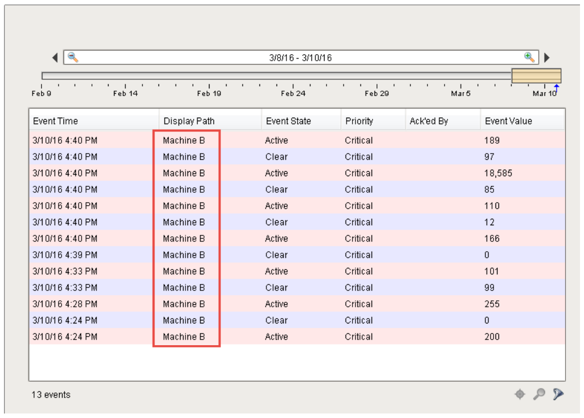


Name	Alarm Journal
Enabled	✓ true
Visible	✓ true
Border	No Border
Journal Name	<auto>
Start Date	03/08/2016 00:00:00 -...
End Date	03/10/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click

Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/10/16 4:40 PM	Ramp	Ack	Medium		
3/10/16 4:40 PM	Machine B	Active	Critical		110
3/10/16 4:40 PM	Ramp	Clear	High		55.01
3/10/16 4:40 PM	Ramp	Clear	High		25.68
3/10/16 4:40 PM	Ramp	Active	High		5.01
3/10/16 4:40 PM	Ramp	Ack	High		
3/10/16 4:40 PM	Ramp	Clear	Medium		-18.32
3/10/16 4:40 PM	Ramp	Active	High		-18.32
3/10/16 4:40 PM	Ramp	Ack	High		
3/10/16 4:40 PM	Ramp	Clear	High		55.01
3/10/16 4:40 PM	Machine B	Clear	Critical		12
3/10/16 4:40 PM	Ramp	Active	Medium		88.3467
3/10/16 4:40 PM	Ramp	Ack	Medium		
3/10/16 4:40 PM	Machine B	Active	Critical		166
3/10/16 4:40 PM	Machine B	Clear	Medium		166

1,519 events

Now, the Alarm Journal Table will show only alarms for **'Group A.'** So it is very easy to filter on associated data. The associated data is outside the folder structure and outside the Display Path. It allows you to filter any way you want so you can look at alarms in a particular area of the plant or for a certain set of alarms.



Event Time	Display Path	Event State	Priority	Ack'd By	Event Value
3/10/16 4:40 PM	Machine B	Active	Critical		189
3/10/16 4:40 PM	Machine B	Clear	Critical		97
3/10/16 4:40 PM	Machine B	Active	Critical		18,585
3/10/16 4:40 PM	Machine B	Clear	Critical		85
3/10/16 4:40 PM	Machine B	Active	Critical		110
3/10/16 4:40 PM	Machine B	Clear	Critical		12
3/10/16 4:40 PM	Machine B	Active	Critical		166
3/10/16 4:39 PM	Machine B	Clear	Critical		0
3/10/16 4:33 PM	Machine B	Active	Critical		101
3/10/16 4:33 PM	Machine B	Clear	Critical		99
3/10/16 4:28 PM	Machine B	Active	Critical		255
3/10/16 4:24 PM	Machine B	Clear	Critical		0
3/10/16 4:24 PM	Machine B	Active	Critical		200

13 events

Next...

- Alarm Journal - Row Styles

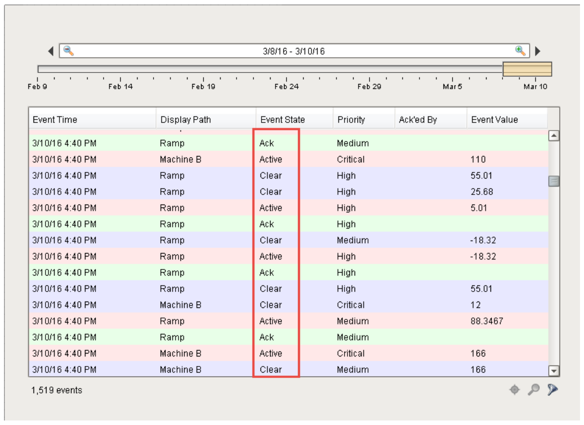
Alarm Journal - Row Styles

Alarm Journal - Row Styles

Watch the Video

You can customize the row styles for different states of alarm history on the **Alarm Journal Table** component.

Just like the Alarm Status Table, the Alarm Journal Table comes with a particular set of colors associated with each of the alarm states as shown in this example. The Alarm Journal Table has a built-in feature to change the colors of each individual row based on the state of the alarm. You can change each of these colors for each of the alarm states in the **Alarm Rows Styles Customizer**.

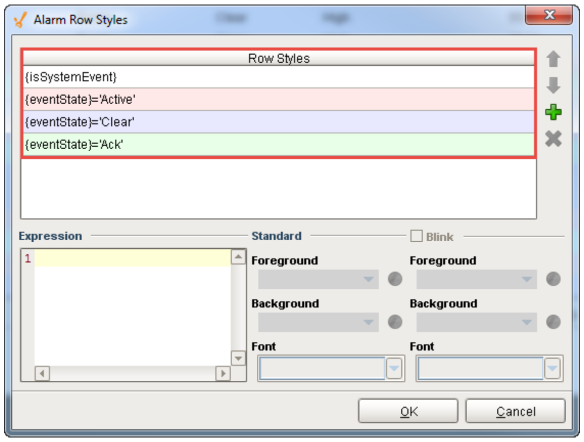


Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/10/16 4:40 PM	Ramp	Ack	Medium		
3/10/16 4:40 PM	Machine B	Active	Critical		110
3/10/16 4:40 PM	Ramp	Clear	High		55.01
3/10/16 4:40 PM	Ramp	Clear	High		25.68
3/10/16 4:40 PM	Ramp	Active	High		5.01
3/10/16 4:40 PM	Ramp	Ack	High		
3/10/16 4:40 PM	Ramp	Clear	Medium		-18.32
3/10/16 4:40 PM	Ramp	Active	High		-18.32
3/10/16 4:40 PM	Ramp	Ack	High		
3/10/16 4:40 PM	Ramp	Clear	High		55.01
3/10/16 4:40 PM	Machine B	Clear	Critical		12
3/10/16 4:40 PM	Ramp	Active	Medium		88.3467
3/10/16 4:40 PM	Ramp	Ack	Medium		
3/10/16 4:40 PM	Machine B	Active	Critical		166
3/10/16 4:40 PM	Machine B	Clear	Medium		166

1,519 events

Right click on the **Alarm Journal Table Component** and go down to **Customizers** and select **Alarm Row Styles.** The Alarm Row Styles window will open like the one in this example.

You can see the default styles, colors and states. You can add as many styles as you want. Each style has an expression, a color, and if you want it to blink or not. In this example, the SystemEvent is white, the Active is red, Clear is blue, and Ack is green.

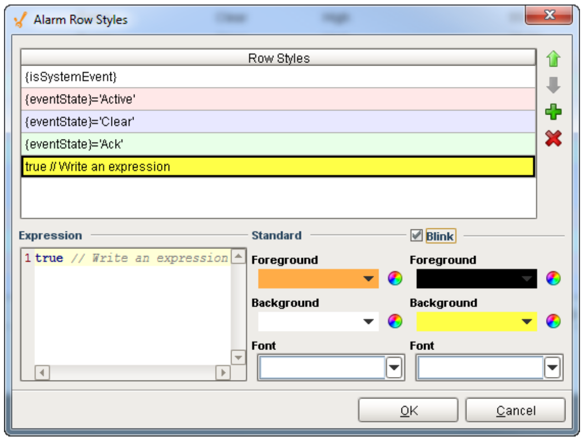


Row Styles
{isSystemEvent}
{eventState}='Active'
{eventState}='Clear'
{eventState}='Ack'

You can add more styles or modify the expressions that currently exist. Click on the green plus icon ➕ to add a new row.

In this example, Row Styles gives you a head start for building your new expression. You can use priority, state, display path, active time, cleared time, etc., in your expression. You can do any evaluations you want and use any parts of the alarm for building your expression.

Alarms are evaluated in sequential order, from top to bottom. The first alarm that returns **'True'** is the style that will be used. If the alarm returns **'False,'** that style will not be used.



Alarm Row Styles

Row Styles
{isSystemEvent}
{eventState}='Active'
{eventState}='Clear'
{eventState}='Ack'
true // Write an expression

Expression

```
true // Write an expression
```

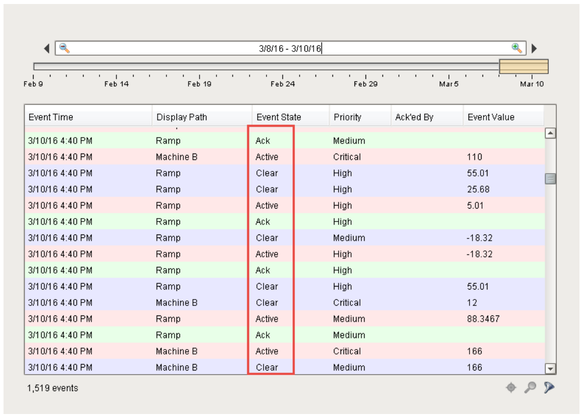
Standard
Foreground
Background
Font

Blink ✓
Foreground
Background
Font

OK Cancel

Once the changes are made to row styles, you can see that all the red alarms are the Active alarms, all the Cleared are blue, and all the Ack are the green alarms. So you can see how easy it is to add more alarm states, and change the colors of each of the individual rows in Alarm Journal Row Styles.

Refer to the Alarm Properties section to learn more about alarm properties and how they are used in the Alarm Row Style Customizer.



3/8/16 - 3/10/16

Event Time	Display Path	Event State	Priority	Ack'ed By	Event Value
3/10/16 4:40 PM	Ramp	Ack	Medium		
3/10/16 4:40 PM	Machine B	Active	Critical		110
3/10/16 4:40 PM	Ramp	Clear	High		55.01
3/10/16 4:40 PM	Ramp	Clear	High		25.68
3/10/16 4:40 PM	Ramp	Active	High		5.01
3/10/16 4:40 PM	Ramp	Ack	High		
3/10/16 4:40 PM	Ramp	Clear	Medium		-18.32
3/10/16 4:40 PM	Ramp	Active	High		-18.32
3/10/16 4:40 PM	Ramp	Ack	High		
3/10/16 4:40 PM	Ramp	Clear	High		55.01
3/10/16 4:40 PM	Machine B	Clear	Critical		12
3/10/16 4:40 PM	Ramp	Active	Medium		88.3467
3/10/16 4:40 PM	Ramp	Ack	Medium		
3/10/16 4:40 PM	Machine B	Active	Critical		166
3/10/16 4:40 PM	Machine B	Clear	Medium		166

1,519 events

Next...

- Alarm Journal - Focusing on Alarms

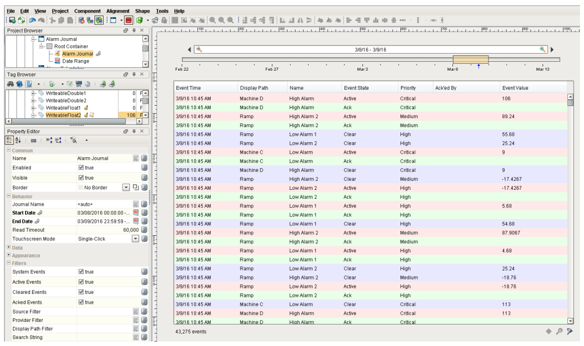
Alarm Journal - Focusing on Alarms

Alarm Journal - Focusing on Alarms

Watch the Video

Focusing on alarms is a really nice feature built into the Alarm Journal Table component that allows you to focus on a specific alarm in the alarm event list. Typically, the Alarm Journal Table shows you the history of all your alarms within a specific time period that you selected using the Data Range component. The **Focus feature** allows you to focus in on one specific alarm to view the history for that alarm.

In this example, you can see a lot of Machine C and Machine D alarms including quite a few Ramp alarms. As you can see, the Alarm Journal Table is displaying a complete list of all alarms.

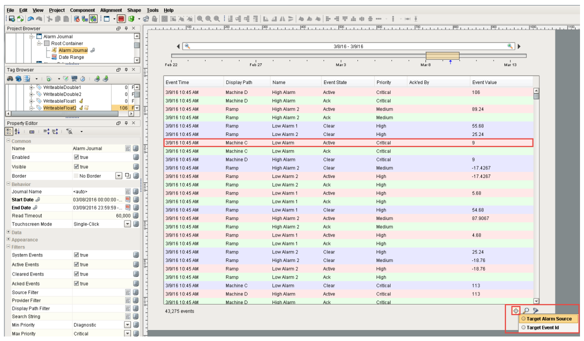


Name	Alarm Journal
Enabled	true
Visible	true
Border	No Border
Journal Name	<auto>
Start Date	03/08/2016 00:00:00 -...
End Date	03/09/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click
System Events	true
Active Events	true
Cleared Events	true
Acked Events	true
Source Filter	
Provider Filter	
Display Path Filter	
Search String	

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		106
3/9/16 10:45 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:45 AM	Ramp	High Alarm 2	Active	Medium		89.24
3/9/16 10:45 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		55.68
3/9/16 10:45 AM	Ramp	Low Alarm 2	Clear	High		25.24
3/9/16 10:45 AM	Machine C	Low Alarm	Active	Critical		9
3/9/16 10:45 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:45 AM	Machine D	High Alarm	Clear	Critical		9
3/9/16 10:45 AM	Ramp	High Alarm 2	Clear	Medium		-17.4267
3/9/16 10:45 AM	Ramp	Low Alarm 2	Active	High		-17.4267
3/9/16 10:45 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Active	High		5.68
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		54.68
3/9/16 10:45 AM	Ramp	High Alarm 2	Active	Medium		87.9067
3/9/16 10:45 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Active	High		4.68
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 2	Clear	High		25.24
3/9/16 10:45 AM	Ramp	High Alarm 2	Clear	Medium		-18.76
3/9/16 10:45 AM	Ramp	Low Alarm 2	Active	High		-18.76
3/9/16 10:45 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:45 AM	Machine C	Low Alarm	Clear	Critical		113
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		113
3/9/16 10:45 AM	Machine D	High Alarm	Ack	Critical		

43,275 events

If you want to see just all Machine C alarms, click on any **Machine C** alarm event. Go to the bottom of the Alarm Journal Table and click on the **Focus icon** which looks like a little target symbol. It allows you to focus on the Alarm Source or the Event Id. Select the **Target Alarm Source** option. The Target Alarm Source is simply the name, Machine C.

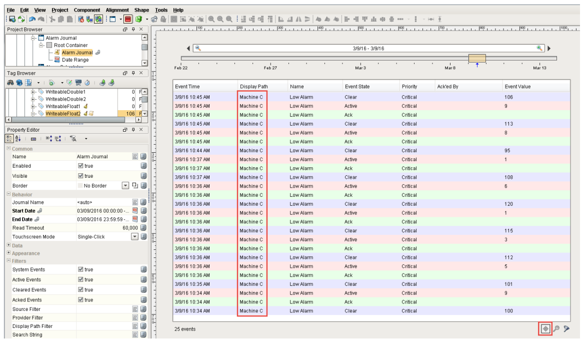


Name	Alarm Journal
Enabled	true
Visible	true
Border	No Border
Journal Name	<auto>
Start Date	03/08/2016 00:00:00 -...
End Date	03/09/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click
System Events	true
Active Events	true
Cleared Events	true
Acked Events	true
Source Filter	
Provider Filter	
Display Path Filter	
Search String	
Min Priority	Diagnostic
Max Priority	Critical

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		106
3/9/16 10:45 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:45 AM	Ramp	High Alarm 2	Active	Medium		89.24
3/9/16 10:45 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		55.68
3/9/16 10:45 AM	Ramp	Low Alarm 2	Clear	High		25.24
3/9/16 10:45 AM	Machine C	Low Alarm	Active	Critical		9
3/9/16 10:45 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:45 AM	Machine D	High Alarm	Clear	Critical		9
3/9/16 10:45 AM	Ramp	High Alarm 2	Clear	Medium		-17.4267
3/9/16 10:45 AM	Ramp	Low Alarm 2	Active	High		-17.4267
3/9/16 10:45 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Active	High		5.68
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		54.68
3/9/16 10:45 AM	Ramp	High Alarm 2	Active	Medium		87.9067
3/9/16 10:45 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Active	High		4.68
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 2	Clear	High		25.24
3/9/16 10:45 AM	Ramp	High Alarm 2	Clear	Medium		-18.76
3/9/16 10:45 AM	Ramp	Low Alarm 2	Active	High		-18.76
3/9/16 10:45 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:45 AM	Machine C	Low Alarm	Clear	Critical		113
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		113
3/9/16 10:45 AM	Machine D	High Alarm	Ack	Critical		

43,275 events

Once you click on the Target Alarm Source you will see all the Machine C alarms. You can see all the alarm event activity that occurred for Machine C including the changes to the Active and Cleared states. You can then remove the focus from Machine C and go back to see the complete list of all alarms by clicking on the **Focus icon** again.

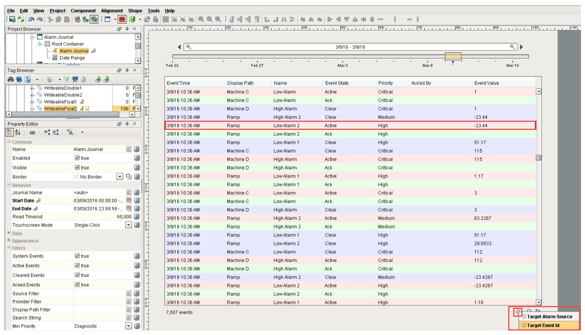


Name	Alarm Journal
Enabled	true
Visible	true
Border	No Border
Journal Name	<auto>
Start Date	03/09/2016 00:00:00 -...
End Date	03/09/2016 23:59:59 -...
Read Timeout	60,000
Touchscreen Mode	Single-Click
System Events	true
Active Events	true
Cleared Events	true
Acked Events	true
Source Filter	
Provider Filter	
Display Path Filter	
Search String	

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:45 AM	Machine C	Low Alarm	Clear	Critical		106
3/9/16 10:45 AM	Machine C	Low Alarm	Active	Critical		9
3/9/16 10:45 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:45 AM	Machine C	Low Alarm	Clear	Critical		113
3/9/16 10:45 AM	Machine C	Low Alarm	Active	Critical		8
3/9/16 10:45 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:44 AM	Machine C	Low Alarm	Clear	Critical		95
3/9/16 10:37 AM	Machine C	Low Alarm	Active	Critical		1
3/9/16 10:37 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:37 AM	Machine C	Low Alarm	Clear	Critical		108
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		6
3/9/16 10:36 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:36 AM	Machine C	Low Alarm	Clear	Critical		120
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		1
3/9/16 10:36 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:36 AM	Machine C	Low Alarm	Clear	Critical		115
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		3
3/9/16 10:36 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:36 AM	Machine C	Low Alarm	Clear	Critical		112
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		5
3/9/16 10:36 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:35 AM	Machine C	Low Alarm	Clear	Critical		101
3/9/16 10:34 AM	Machine C	Low Alarm	Active	Critical		9
3/9/16 10:34 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:34 AM	Machine C	Low Alarm	Clear	Critical		100

25 events

You can also focus on a specific alarm Id; one specific event that already happened. In this example, **Ramp 3/9/16 at 10:36 AM with an Active state** was selected, and the **Target Event Id** was clicked. Focusing on the Target Event Id will show you all the alarms occurring for that specific alarm event.

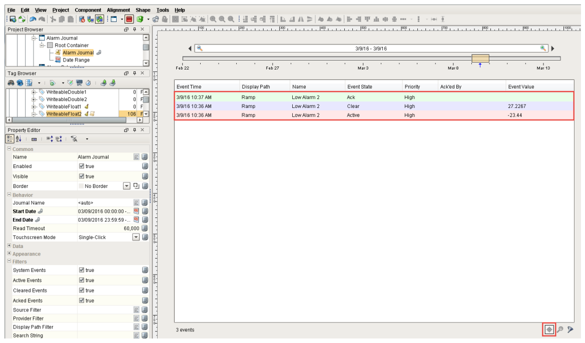


3/9/16 - 3/9/16

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		1
3/9/16 10:36 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:36 AM	Machine D	High Alarm	Clear	Critical		1
3/9/16 10:36 AM	Ramp	High Alarm 2	Clear	Medium		-23.44
3/9/16 10:36 AM	Ramp	Low Alarm 2	Active	High		-23.44
3/9/16 10:36 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:36 AM	Ramp	Low Alarm 1	Clear	High		51.17
3/9/16 10:36 AM	Machine C	Low Alarm	Clear	Critical		115
3/9/16 10:36 AM	Machine D	High Alarm	Active	Critical		115
3/9/16 10:36 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:36 AM	Ramp	Low Alarm 1	Active	High		1.17
3/9/16 10:36 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		3
3/9/16 10:36 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:36 AM	Machine D	High Alarm	Clear	Critical		3
3/9/16 10:36 AM	Ramp	High Alarm 2	Active	Medium		83.2267
3/9/16 10:36 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:36 AM	Ramp	Low Alarm 1	Clear	High		51.17
3/9/16 10:36 AM	Ramp	Low Alarm 2	Clear	High		29.8933
3/9/16 10:36 AM	Machine C	Low Alarm	Clear	Critical		112
3/9/16 10:36 AM	Machine D	High Alarm	Active	Critical		112
3/9/16 10:36 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:36 AM	Ramp	High Alarm 2	Clear	Medium		-23.4267
3/9/16 10:36 AM	Ramp	Low Alarm 2	Active	High		-23.4267
3/9/16 10:36 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:36 AM	Ramp	Low Alarm 1	Active	High		1.18

7,507 events

You can see when an alarm went Active, when it was Cleared, and when it was Acknowledged. Click the **Focus icon** to remove the focus and return to the complete list of all alarms.

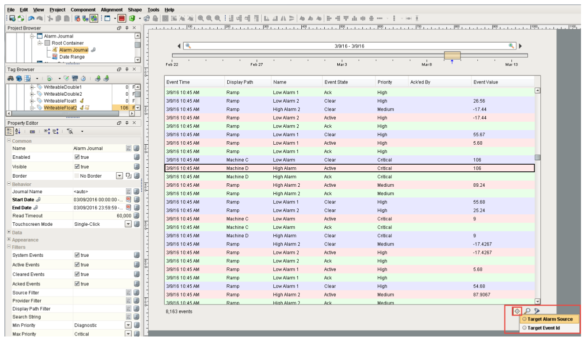


3/9/16 - 3/9/16

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:37 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:36 AM	Ramp	Low Alarm 2	Clear	High		27.2267
3/9/16 10:36 AM	Ramp	Low Alarm 2	Active	High		-23.44

3 events

When you have a big list of all alarm events, the Active, the Cleared and the Acknowledged states are not going to be in sequential order depending on when the event times actually happened so it helps to use the Focus feature to to filter on alarms.



3/9/16 - 3/9/16

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 2	Clear	High		26.56
3/9/16 10:45 AM	Ramp	High Alarm 2	Clear	Medium		-17.44
3/9/16 10:45 AM	Ramp	Low Alarm 2	Active	High		-17.44
3/9/16 10:45 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		55.67
3/9/16 10:45 AM	Ramp	Low Alarm 1	Active	High		5.68
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Machine C	Low Alarm	Clear	Critical		106
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		106
3/9/16 10:45 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:45 AM	Ramp	High Alarm 2	Active	Medium		89.24
3/9/16 10:45 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		55.68
3/9/16 10:45 AM	Ramp	Low Alarm 2	Clear	High		25.24
3/9/16 10:45 AM	Machine C	Low Alarm	Active	Critical		9
3/9/16 10:45 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:45 AM	Machine D	High Alarm	Clear	Critical		9
3/9/16 10:45 AM	Ramp	High Alarm 2	Clear	Medium		-17.4267
3/9/16 10:45 AM	Ramp	Low Alarm 2	Active	High		-17.4267
3/9/16 10:45 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Active	High		5.68
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		54.68
3/9/16 10:45 AM	Ramp	High Alarm 2	Active	Medium		87.9067
3/9/16 10:45 AM	Ramp	High Alarm 2	Ack	Medium		

8,163 events

Using the Focus feature, you can actually focus on one Target Alarm Source or the Target Event Id to look at when alarms occurred. It's a nice way of filtering through the Alarm Journal Table without having to scroll through the entire list of alarms to match up the alarm events, times and states.

Next...

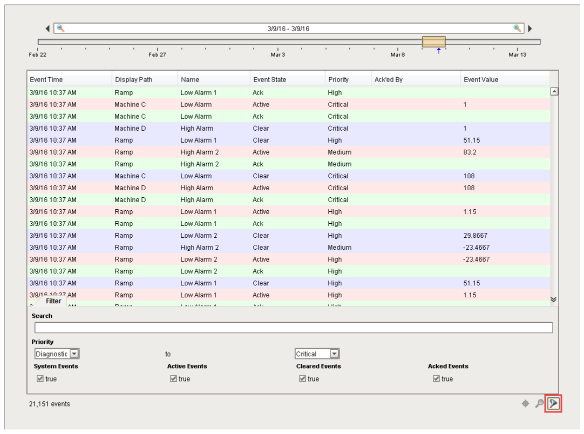
- Alarm Journal - Searching

Alarm Journal - Searching

Another nice feature of the **Alarm Journal Table** component is the ability to search through the history of alarms and filter for particular alarms with any criteria. The Search feature is built into the Alarm Journal Table component. There are no extra components outside of the table to bind these properties too. On the bottom right side of the table, you can see the little Filter icon. Click on the Filter icon to start your search.

Alarm Journal - Searching

Watch the Video


3/9/16 - 3/9/16

Feb 22 · Feb 27 · Mar 3 · Mar 8 · Mar 13

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:37 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:37 AM	Machine C	Low Alarm	Active	Critical		1
3/9/16 10:37 AM	Machine C	Low Alarm	Ack	Critical		
3/9/16 10:37 AM	Machine D	High Alarm	Clear	Critical		1
3/9/16 10:37 AM	Ramp	Low Alarm 1	Clear	High		51.15
3/9/16 10:37 AM	Ramp	High Alarm 2	Active	Medium		83.2
3/9/16 10:37 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:37 AM	Machine C	Low Alarm	Clear	Critical		108
3/9/16 10:37 AM	Machine D	High Alarm	Active	Critical		108
3/9/16 10:37 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:37 AM	Ramp	Low Alarm 1	Active	High		1.15
3/9/16 10:37 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:37 AM	Ramp	Low Alarm 2	Clear	High		29.8667
3/9/16 10:37 AM	Ramp	High Alarm 2	Clear	Medium		-23.4667
3/9/16 10:37 AM	Ramp	Low Alarm 2	Active	High		-23.4667
3/9/16 10:37 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:37 AM	Ramp	Low Alarm 1	Clear	High		51.15
3/9/1 AM	Ramp	Low Alarm 1	Active	High		1.15

Filter

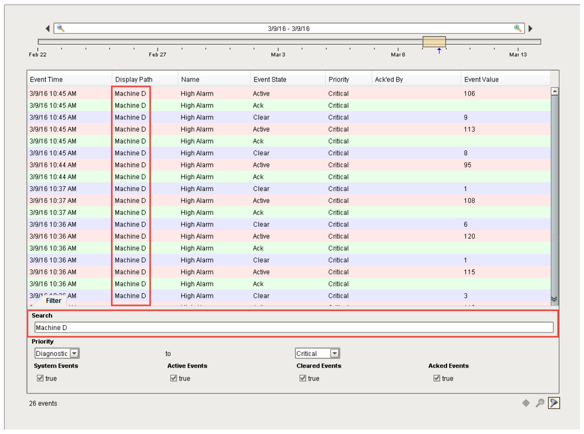
Search:

Priority: Diagnostic to Critical

System Events ☑ true · Active Events ☑ true · Cleared Events ☑ true · Acked Events ☑ true

21,151 events

You can type in any search criteria you desire as well as filter on Priority and Event types. This example shows how to find all Machine D alarms. Enter **'Machine D'** in the Search bar. The **Alarm Journal Table** will display only the results for Machine D, essentially, letting the operator enter their own criteria and perform the search without having to include other components or bindings.


3/9/16 - 3/9/16

Feb 22 · Feb 27 · Mar 3 · Mar 8 · Mar 13

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		106
3/9/16 10:45 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:45 AM	Machine D	High Alarm	Clear	Critical		9
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		113
3/9/16 10:45 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:45 AM	Machine D	High Alarm	Clear	Critical		8
3/9/16 10:44 AM	Machine D	High Alarm	Active	Critical		95
3/9/16 10:44 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:37 AM	Machine D	High Alarm	Clear	Critical		1
3/9/16 10:37 AM	Machine D	High Alarm	Active	Critical		108
3/9/16 10:37 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:36 AM	Machine D	High Alarm	Clear	Critical		6
3/9/16 10:36 AM	Machine D	High Alarm	Active	Critical		120
3/9/16 10:36 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:36 AM	Machine D	High Alarm	Clear	Critical		1
3/9/16 10:36 AM	Machine D	High Alarm	Active	Critical		115
3/9/16 10:36 AM	Machine D	High Alarm	Ack	Critical		
3/9/1 AM	Machine D	High Alarm	Clear	Critical		3

Filter

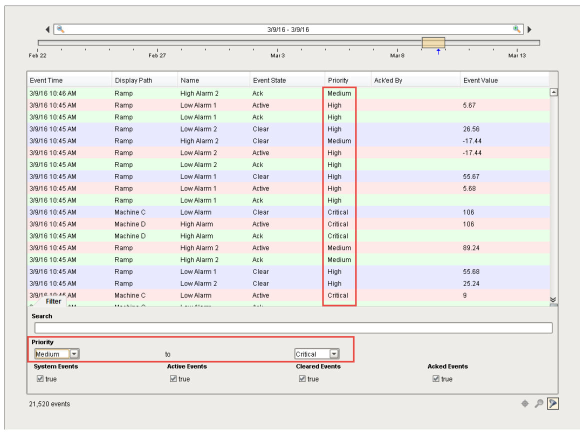
Search: Machine D

Priority: Diagnostic to Critical

System Events ☑ true · Active Events ☑ true · Cleared Events ☑ true · Acked Events ☑ true

26 events

You can also filter on **Priority**. This example shows all the alarms with a priority from Medium to Critical.


3/9/16 - 3/9/16

Feb 22 · Feb 27 · Mar 3 · Mar 8 · Mar 13

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:46 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Active	High		5.67
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 2	Clear	High		26.56
3/9/16 10:45 AM	Ramp	High Alarm 2	Clear	Medium		-17.44
3/9/16 10:45 AM	Ramp	Low Alarm 2	Active	High		-17.44
3/9/16 10:45 AM	Ramp	Low Alarm 2	Ack	High		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		55.67
3/9/16 10:45 AM	Ramp	Low Alarm 1	Active	High		5.68
3/9/16 10:45 AM	Ramp	Low Alarm 1	Ack	High		
3/9/16 10:45 AM	Machine C	Low Alarm	Clear	Critical		106
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		106
3/9/16 10:45 AM	Machine D	High Alarm	Ack	Critical		
3/9/16 10:45 AM	Ramp	High Alarm 2	Active	Medium		89.24
3/9/16 10:45 AM	Ramp	High Alarm 2	Ack	Medium		
3/9/16 10:45 AM	Ramp	Low Alarm 1	Clear	High		55.68
3/9/16 10:45 AM	Ramp	Low Alarm 2	Clear	High		25.24
3/9/1 AM	Machine C	Low Alarm	Active	Critical		9

Filter

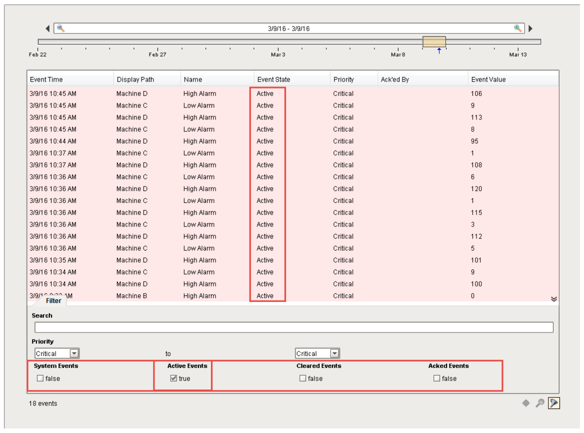
Search:

Priority: Medium to Critical

System Events ☑ true · Active Events ☑ true · Cleared Events ☑ true · Acked Events ☑ true

21,520 events

You can also filter based on the type of events you want to see. If you only want to look for just the Active alarms, you can uncheck everything else and simply leave the **Active Events** checked.


3/9/16 - 3/9/16

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		106
3/9/16 10:45 AM	Machine C	Low Alarm	Active	Critical		9
3/9/16 10:45 AM	Machine D	High Alarm	Active	Critical		113
3/9/16 10:45 AM	Machine C	Low Alarm	Active	Critical		8
3/9/16 10:44 AM	Machine D	High Alarm	Active	Critical		95
3/9/16 10:37 AM	Machine C	Low Alarm	Active	Critical		1
3/9/16 10:37 AM	Machine D	High Alarm	Active	Critical		108
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		6
3/9/16 10:36 AM	Machine D	High Alarm	Active	Critical		120
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		1
3/9/16 10:36 AM	Machine D	High Alarm	Active	Critical		115
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		3
3/9/16 10:36 AM	Machine D	High Alarm	Active	Critical		112
3/9/16 10:36 AM	Machine C	Low Alarm	Active	Critical		5
3/9/16 10:35 AM	Machine D	High Alarm	Active	Critical		101
3/9/16 10:34 AM	Machine C	Low Alarm	Active	Critical		9
3/9/16 10:34 AM	Machine D	High Alarm	Active	Critical		100
3/9/[illegible] AM	Machine B	High Alarm	Active	Critical		0

Filter

Search

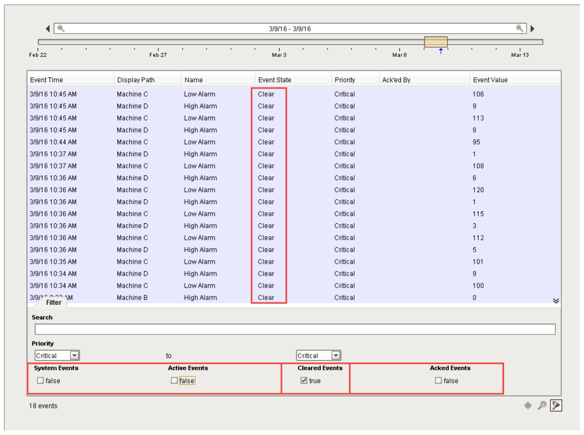
Priority

Critical to Critical

System Events	Active Events	Cleared Events	Acked Events
☐ false	☑ true	☐ false	☐ false

18 events

If you only want to display the Cleared alarms, you can uncheck all the other events and leave the **Cleare d Events** checked.


3/9/16 - 3/9/16

Event Time	Display Path	Name	Event State	Priority	Ack'ed By	Event Value
3/9/16 10:45 AM	Machine C	Low Alarm	Clear	Critical		106
3/9/16 10:45 AM	Machine D	High Alarm	Clear	Critical		9
3/9/16 10:45 AM	Machine C	Low Alarm	Clear	Critical		113
3/9/16 10:45 AM	Machine D	High Alarm	Clear	Critical		8
3/9/16 10:44 AM	Machine C	Low Alarm	Clear	Critical		95
3/9/16 10:37 AM	Machine D	High Alarm	Clear	Critical		1
3/9/16 10:37 AM	Machine C	Low Alarm	Clear	Critical		108
3/9/16 10:36 AM	Machine D	High Alarm	Clear	Critical		6
3/9/16 10:36 AM	Machine C	Low Alarm	Clear	Critical		120
3/9/16 10:36 AM	Machine D	High Alarm	Clear	Critical		1
3/9/16 10:36 AM	Machine C	Low Alarm	Clear	Critical		115
3/9/16 10:36 AM	Machine D	High Alarm	Clear	Critical		3
3/9/16 10:36 AM	Machine C	Low Alarm	Clear	Critical		112
3/9/16 10:36 AM	Machine D	High Alarm	Clear	Critical		5
3/9/16 10:35 AM	Machine C	Low Alarm	Clear	Critical		101
3/9/16 10:34 AM	Machine D	High Alarm	Clear	Critical		9
3/9/16 10:34 AM	Machine C	Low Alarm	Clear	Critical		100
3/9/[illegible] AM	Machine B	High Alarm	Clear	Critical		0

Filter

Search

Priority

Critical to Critical

System Events	Active Events	Cleared Events	Acked Events
☐ false	☐ false	☑ true	☐ false

18 events

Next...

Alarm Notification

Voice and SMS Notification may require additional hardware purchase such as a cellular modem or the subscription to a SMS texting service.

Alarm Notification Profiles

A notification profile defines a delivery channel by which an alarm notification message is sent to a user. There are three types of delivery channels: Email, Phone, and SMS.

- Email
  The Alarm Notification Module comes with one-way and two-way email notification profiles. The one-way email notification profile simply sends an email containing the alarm notification message. The two-way email notification profile sends an email that contains links that allow that user to acknowledge the alarm(s).
- Voice
  The Voice Notification Module adds the ability to notify users by calling them on the phone.
- SMS
  The SMS Notification Module adds the ability to send text messages to users to notify them of alarms.

In this section ...

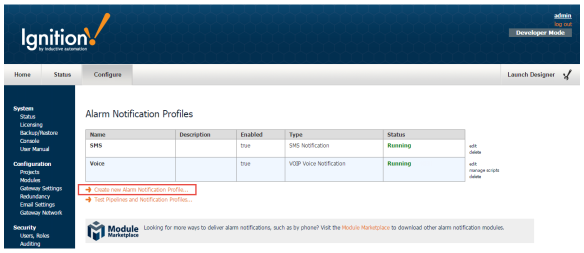
Email Notification Profile

Alarm notification is the act of sending a message to a group of people when an alarm becomes active or cleared. In Ignition, this functionality is enabled by having the Alarm Notification Module installed which provides alarm pipelines and email notification.

Once you have your Email Notification Profile created in Ignition, you can notify people when alarms occur via Email. If the two-way setting is enabled, operators can also acknowledge the alarm by clicking the link in the body of the email.

To create an Email Notification Profile

1. Go to the **Configure** section of the Gateway.
2. Choose **Alarming > Notification** from the menu on the left side. The Alarm Notification Profile page will be displayed.
3. Click on the **Create new Alarm Notification Profile** link.



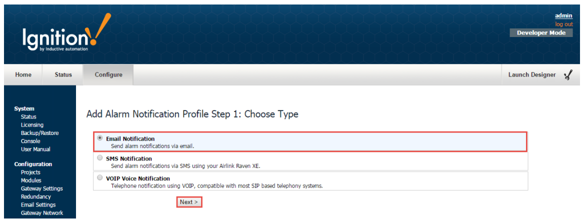
Alarm Notification Profiles

Name	Description	Enabled	Type	Status	
SMS		true	SMS Notification	Running	edit delete
Voice		true	VOIP Voice Notification	Running	edit manage scripts delete

Create new Alarm Notification Profile...
Test Pipelines and Notification Profiles...

Looking for more ways to deliver alarm notifications, such as by phone? Visit the Module Marketplace to download other alarm notification modules.

4. Select the **Email Notification Profile** and click **Next**.



Add Alarm Notification Profile Step 1: Choose Type

[✓ **Email Notification**] Send alarm notifications via email.
SMS Notification Send alarm notifications via SMS using your Airlink Raven XE.
VOIP Voice Notification Telephone notification using VOIP, compatible with most SIP based telephony systems.

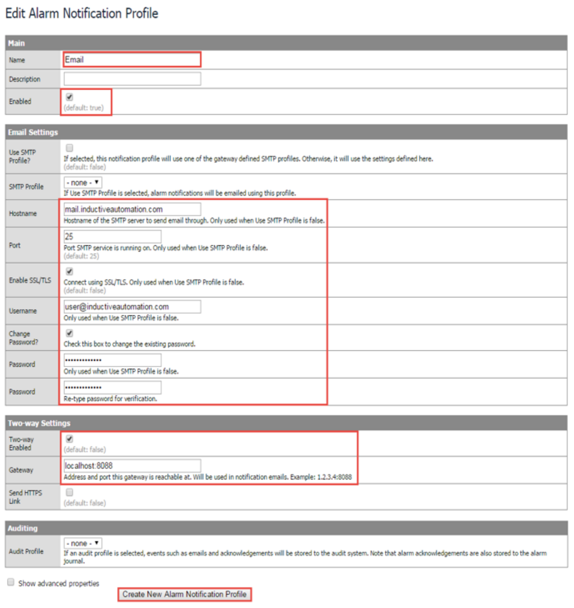
5. Configure the following settings in the Email alarm Notification Profile:

- Enter a **Name**: Email
- Click the **Enabled** box to enable Email notifications.
- Enter the **Hostname**: You can use your company's SMTP server, Gmail, Yahoo, etc.
- Enter the **Port**. The default is **Port 25**, but your mail server may be different.
- Enable **SSL/TLS** if you are using encryption.
- Enter your **Username** and **Password** if they are required.
- Click the **Two-way Enable** box if you want to make your Email Notification Profile two-way so that the operator can acknowledge the email. There is a link to acknowledge the alarm in the email. **Note:** You must have a connection back to the Ignition Gateway. So, if you are working from home and not on a VPN, you will either have to get on a VPN or expose your Ignition server on the internet. You can do this with port forwarding, talk to your IT department about it.
- **Auditing**: You have the option of attaching emails to an Audit profile so that Ignition can store all the events associated with the Email Notification Profile. (I.e,, who is receiving alarm and who is acknowledging an alarm).
- Once you have all the settings correct, such as the SMTP server, the Two-way Enabled, and Auditing, you can click the **Create New Alarm Notification Profile** button.

INDUCTIVE UNIVERSITY

Email Notification Profile

Watch the Video


Edit Alarm Notification Profile

Main	
Name	Email
Description	
Enabled	✓ (default: true)

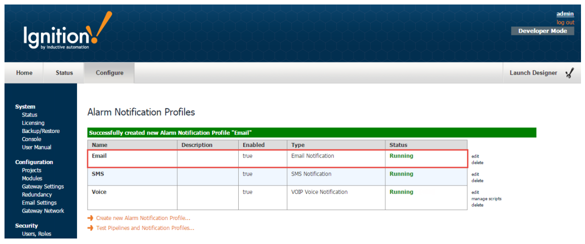
Email Settings	
Use SMTP Profile?	☐ If selected, this notification profile will use one of the gateway defined SMTP profiles. Otherwise, it will use the settings defined here. (default: false)
SMTP Profile	- none - ▼ If Use SMTP Profile is selected, alarm notifications will be emailed using this profile.
Hostname	mail.inductiveautomation.com Hostname of the SMTP server to send email through. Only used when Use SMTP Profile is false.
Port	25 Port SMTP service is running on. Only used when Use SMTP Profile is false. (default: 25)
Enable SSL/TLS	✓ Connect using SSL/TLS. Only used when Use SMTP Profile is false. (default: false)
Username	user@inductiveautomation.com Only used when Use SMTP Profile is false.
Change Password?	✓ Check this box to change the existing password.
Password	•••••••••••••• Only used when Use SMTP Profile is false.
Password	•••••••••••••• Re-type password for verification.

Two-way Settings	
Two-way Enabled	✓ (default: false)
Gateway	localhost:8088 Address and port this gateway is reachable at. Will be used in notification emails. Example: 1.2.3.4:8088
Send HTTPS Link	☐ (default: false)

Auditing	
Audit Profile	- none - ▼ 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

6. Once you created your Email Notification Profile, you will see that it exists in the list of Alarm Notification Profiles, and it will be running. Now, it's available to choose when you want to notify certain people when alarms occur.



Alarm Notification Profiles

Successfully created new Alarm Notification Profile "Email"

Name	Description	Enabled	Type	Status	
Email		true	Email Notification	Running	edit delete
SMS		true	SMS Notification	Running	edit delete
Voice		true	VOIP Voice Notification	Running	edit manage scripts delete

Create new Alarm Notification Profile...
Test Pipelines and Notification Profiles...

Next...

- SMS Notification Profile

SMS Notification Profile

The **SMS Notification Module** allows you to deliver SMS alarm notifications via a cellular modem configured with a SIM card belonging to an active cellular account. If enabled, recipients of these messages can reply with a special code in order to acknowledge the alarm.

To create an SMS notification profile

1. Go to **Configure** section of the Gateway.
2. Choose **Alarming > Notification** from the menu on the left.
   The **Alarm Notification Profiles** page is displayed.
3. Find the orange arrow and click on **Create new Alarm Notification Profile**.
4. Select the **SMS Notification** profile type and click **Next**.
5. Set the following settings concerning your device.

   Airlink Host Address: The IP address of the airlink modem.
   Send Port: Configured in the device, the default port is 17341.
   Receive Port: The port used by Ignition when two-way messaging is enabled. Therefore, the port must not already be used by the host system, and must not be blocked by a firewall. The default port is 17342.
   Two-way Enabled: Default is not enabled. If enabled, the message recipients can acknowledge alarms by replying to the SMS messages received. This is communicated to Ignition via UDP data sent from the modem. Therefore, the Airlink modem must be configured with the IP address of the system.
   Ack Code - The default is not enabled. Ack code must be enabled to acknowledge an SMS message.

6. At the bottom of the page click on **Create new Alarm Notification Profile** to save the profile.

SMS Notification Profile

Watch the Video

AirLink LS300 Setup Guide

Link to Knowledge Base Article

Device Configuration

You can do the basic configuration for the Airlink modem by importing a template settings file provided by Inductive Automation. Please reference the Knowledge Base article link on this page.

Multiple Systems with One Modem

You can use one SMS modem with multiple Ignition systems for one-way messaging. Only one system can receive responses for two-way messaging.

Twilio Module

SMS notifications can also be sent with the Twilio module. This does not require a cellular modem, but the gateway must have Internet access, and a Twilio account must be created (www.twilio.com). Once created, the account must be defined in Ignition.

To define a Twilio account

1. Go to **Configure** section of the Gateway.
2. Choose **Twilio > Accounts** from the menu on the left.
3. Click the **Create new Twilio Account** link.
4. Provide a **Name** for the account as well as the **Twilio Account Sid,** and **Twilio Auth Token**. Both the Account SID and Auth Token can be retrieved from the Account Settings menu in Twilio.
5. If desired, provide the **Public Hostname**. This is the hostname or IP address that inbound requests will be forwarded. If the users should be able to acknowledge alarms by responding to the SMS messages, then this property should list a publicly reachable hostname or IP address.
6. Click the **Create New Twilio Account** button.

Once the account has been defined, a Twilio SMS notification profile must also be created.

To create a Twilio SMS notification profile

1. Go to **Configure** section of the Gateway.
2. Choose **Alarming > Notification** from the menu on the left.
   The **Alarm Notification Profiles** page is displayed.
3. Find the orange arrow and click on **Create new Alarm Notification Profile**.
4. Select the **Twilio SMS Notification** profile type and click **Next**.

5. Provide a name for the profile, and select a **Twilio Account** from the dropdown. **Acknowledgement Allowed** enables users to respond to notifications directly from the SMS message. if enabled, the **Public Hostname** property in the Twilio Account must be configured (see step 5 in "To define a Twilio account" above.
6. Click **Create New Alarm Notification Profile**

Next...

- Voice Notification Profile

Voice Notification Profile

For the **Voice Notification Module** to work, a compatible Voice Module must also be installed.

The Voice Notification Module adds the ability to deliver alarm notifications to users via telephone, using most SIP compatible phone system, excluding those requiring Cnonce. Messages are constructed in text and are delivered through a high quality text-to-speech engine. The engine supports multiple voices and languages.

INDUCTIVE UNIVERSITY

Voice Notification Profile

Watch the Video

Core Features

Deliver voice calls through any SIP compatible phone system. No dedicated hardware required. Messages generated by high-quality text to speech, and not a canned set of prerecorded files.

- Supports multiple languages concurrently, based on user preference.
- Allows users to acknowledge events.
- Supports requiring a personal identification number for additional security.
- Ties into the audit log to audit call events, successful message delivery, and user acknowledgements.
- Supports message consolidation.

About SIP and VOIP

The Session Initiation Protocol, is a highly popular specification for implementing Voice Over IP (VOIP) based phone systems. The protocol itself, as the name suggests, is responsible for initiating communication sessions, and then other protocols such as SDP and RTP are used to actually transfer voice data. Ignition has these protocols built in, SIP is a peer-to-peer protocol, where one side talks directly to the other. However, it is possible to have gateways that repeat and route data between the two parties. Sometimes phone calls on VOIP networks stay purely in software, but often a gateway will transition the call to a traditional phone line.

By leveraging SIP, Ignition can call physical phones, soft phones, be worked into more complex PBX schemes, while avoiding the high cost of traditional, dedicated voice cards. To get started, though, you'll need some sort of SIP gateway. Asterisk is a popular, open source, system that is used by thousands of companies worldwide. If you simply want to connect to a phone line, the Atcom IP01 FXO box is a low cost device that runs Asterisk.

Gateway Configuration

To get started with voice notification, add a new profile by going to Configure -> Alarming -> Notification in the Ignition gateway. You are only required to specify the host address of the SIP gateway, though depending on the gateway, you may be required to enter a username and password.

There are additional settings that dictate how calls are managed, such as timeouts for answering, and the maximum amount of time that a call can take. Additionally, you can choose to link the notification profile to an audit profile in order to record important call lifecycle events.

After saving the profile, you should see the status update from "Unknown" to "Registered", indicating that the gateway has successfully registered with the SIP gateway. If you see an error, verify that the settings are correct, and that the username and password are correct. The system log console can also be useful in determining what is wrong.

Note: If you receive errors indicating that an "invalid parameter" has been used, try setting the local and public bind interface settings under the advanced options. These should be set to the IP address of the network card that is being used to communicate with the SIP gateway. On some systems, especially Linux hosts, the default empty values result in this error.

Using Skype

If you don't have an existing VOIP system in place, using a hosted service is the quickest way to get going with the voice notification module. Skype offers SIP based service through their Skype Connect product. To get started, you must have a business account, which provides you access to the Skype Manager. From there, you can create a new Skype Connect channel. For more information, visit http://manager.skype.com

Once you have created a Skype Connect channel, configuration in the gateway is similar to that of any other PBX system. The host will be "sip.skype.com", and the user name and password will be those provided by skype during the registration process. Note: while skype allows to you specify how many "channels" may be used, Ignition currently only supports one channel at a time.

Configuring Messages

The message played to the user during the phone calls is defined in the call script. The script dictates the overall structure of the call, defining the phrases and options, and the possible responses. The messages for each alarm are built off of a message template that can reference properties in the alarm. The script can be edited by selecting "manage scripts" next to the voice notification profile. Note: Although the link appears next to a particular profile, the scripts are shared across all voice notification profiles.

The role of each phrase in the script is explained on the settings page. Some parts of the script, such as the phrase requesting the user's PIN number, will only be used if certain settings are configured on the notification block in the pipeline (in this case, the setting to require a PIN). As previously mentioned, the alarm message (for both active and clear) can reference any property of the alarm. The default message looks like this:

At {eventTime|hh:mm:ss}, alarm named {name} became {eventState} with a value of {eventValue}

In this case, the message refers to the alarm name, the eventState and eventValue (note: eventState is different than state. Event state is just the transition that triggered the alarm, such as "active", whereas "state" is the full current state, such as "active, unacknowledged"), and the eventTime. Notice that the event time is formatted to only use the time, and not include the current date.

Scripts can be created for different languages. When the system attempts to deliver a notification, it will look to see if the target user has a preferred language. If so, and the language has both a script defined, and a compatible voice installed, the user will be notified in that language.

Note: Alarms also allow you to define a custom message, relevant to that particular state. If a custom message is defined, it will be used instead of the default message in the script.

The Call Lifecycle

A call is initiated when an alarm event enters the notification profile block in a pipeline. When this occurs, the target users are collected, based on the defined call roster, and the current schedule. Only users who have phone contact details defined will be selected for phone notification.

The voice system can only call one number at a time, and so it takes the first contact off of the queue and initiates the call. The user is given up to the "answer timeout" to answer. After picking up, the user will be asked to enter their pin number, or press any number to continue. The call is not considered "answered" until this action occurs, so the message will be repeated until the answer timeout expires. By acting in this way, the system is able to confirm that a human has actually answered, and the call will then be audited as successful.

Once past the initial challenge, the user will then hear the alarm messages. After each message, they will be asked to either acknowledge, ignore, or repeat the message. Selecting "acknowledge" will cause the alarm to be acknowledged in the alarming system, likely causing it to drop out of the alarm pipeline (dependant on pipeline settings). Ignoring the alarm indicates that the user has heard the message, but cannot or does not want to acknowledge the alarm.

Once the call has completed, the notification system will check the alarm events against the pipeline fallout conditions, and move on to the next call. The system will cycle through all alarms and all contacts (and all phone numbers for each contact) until everyone has been notified, or the pipeline settings have caused all events to drop out.

Pipeline Configuration

The voice notification system is accessed through the Notification Block in the Designer's pipeline configuration, like all other notification methods. As with other methods, you select a call roster, and can optionally turn on consolidation. There are only two options which can be set on the notification block:

Require PIN: If selected, the user will be required to enter their personal identification number in order to hear the message. The user's PIN number is specified on their profile in the user management system. If false, or if the user does not have a PIN specified, the user will only be required to press any key to continue.

Allow Acknowledgement: If true, the user will be given the opportunity to acknowledge the alarm. If false, they will only be able to hear the alarm and continue.

Next...

- Notification Users

Notification Users

You can add users to Ignition who can receive alarms. Each user must have a name and contact information in order for Ignition to contact them. Users can receive alarm notifications in SMS, phone (voice dialout), and email.

To add a user

1. Go to the **Configure** section of the Gateway.
2. Choose **Security > Users, Roles** from the menu on the left.
   The **User Sources** page is displayed.
3. Select **manage users** from the appropriate user source.
4. Look for the orange arrow and select the **Add User** link to add a user.
   The **Manage Users and Roles for Profile** page is displayed.
5. Complete the fields that correspond to the user's information.
6. In the **Contact Info** section, choose the delivery method, and the appropriate value such as an email address.
7. Cick **Add User**.

INDUCTIVE UNIVERSITY

Notification Users

Watch the Video

Next...

- User Schedules

User Schedules

Schedules define the times of 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.

INDUCTIVE UNIVERSITY

User Schedules

Watch the Video

To define a new schedule

1. Go to the **Configure** 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.

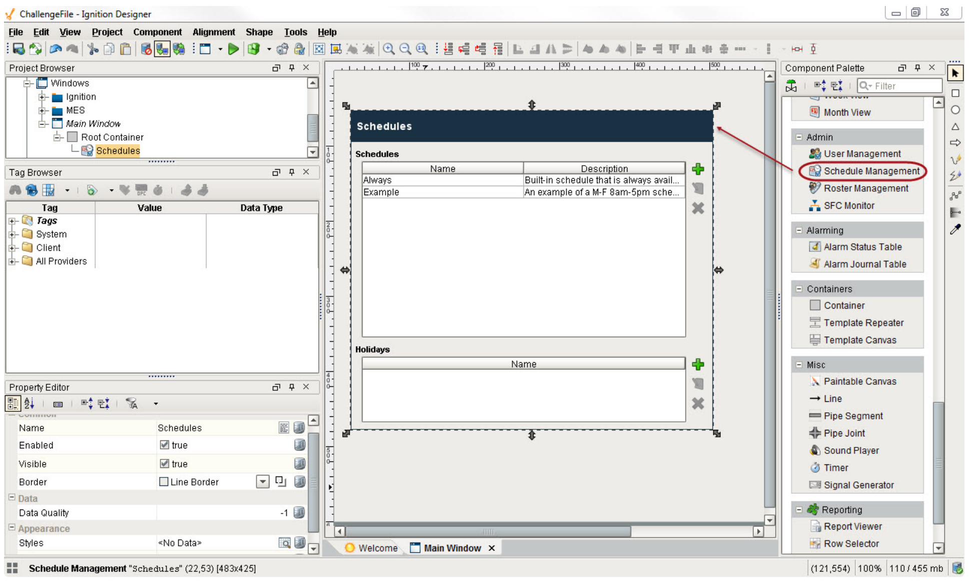
5. On the **New Schedule** page, enter a **Name**, the **Schedule** to be notified, and any **Repeating** options, then click **Add Schedule**.
   Each schedule is defined by which days of the week it is active for, and during what times. Additionally, you can define repeating patterns that repeat either weekly or daily. 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.

To manage schedules from the Client

You can manage schedules from a Vision Client.

1. Open your design project in the **Designer**.

2. From the Component Palette, drag and drop a **Schedule Management** component in your design.



Schedules

Schedules

Name	Description
Always	Built-in schedule that is always avail...
Example	An example of a M-F 8am-5pm sche...

Holidays

Name

Name	Schedules
Enabled	☑ true
Visible	☑ true
Border	☐ Line Border
Data	
Data Quality	-1
Appearance	
Styles	<No Data>

Schedule Management "Schedules" (22,53) [483x425] (121,554) 100% 110 / 455 mb

3. Save your project.
   Users can now manage the schedule from the Client.

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.

Next...

- On-Call Rosters

On-Call Rosters

The On-Call Roster lets you create user groups to be notified when an alarm occurs. Each group includes a list of users in a specific order. Alarm pipeline's notification blocks must choose an on-call roster which defines the users to notify for that notification block. Depending on the alarm notification profile used, the users can be notified one at a time (sequential), or all at once (parallel). It is important to remember that when an on-call roster is used for alarm notification, only those users on the roster whose schedules are active will be notified.

On-Call Rosters

Watch the Video

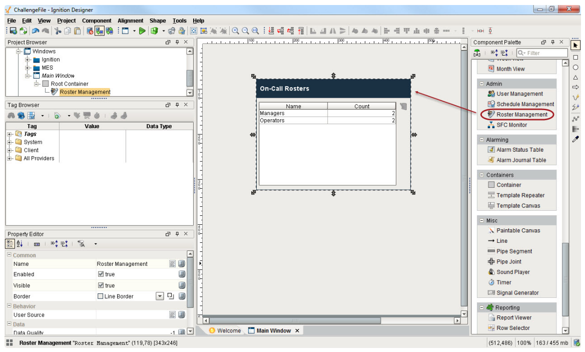
To create and manage the on-call roster

1. Go to the **Configure** section of the Gateway.
2. Choose **Alarming > On-Call Rosters** from the menu on the left.
   The On-Call Rosters page is displayed.
3. Click on **Create New On-Call Roster**.
   You can create as many rosters as you wish.
4. On the **Create New On-Call Roster**, enter a **Name**, and click **Create New On-Call Roster**.
5. Click on **manage** to the right of the roster name you want to add the user list to.
   The **Manage Roster** page is displayed and from here you can choose one or all users from a source list, drag them into the roster list, and put them in any order you wish.

To manage the roster from the Client

Roster management can take place from a Vision Client.

1. Open your design project in the **Designer**.
2. From the Component Palette, drag and drop a **Roster Management** component in your design window.



3. In the **Property Editor**, if you have more than one user source, enter the one you want to be used by the roster in the **User Source** property.
4. Save your project.
   Client users can now manage the roster by re-arranging and adding/removing names to the roster.

Next...

- User, Schedule, and Roster Management

User, Schedule, and Roster Management

You can manage users, schedules, and rosters from the Ignition Client using components to affect the User Sources in the Gateway.

INDUCTIVE UNIVERSITY

User, Schedule, and Roster Management

Watch the Video

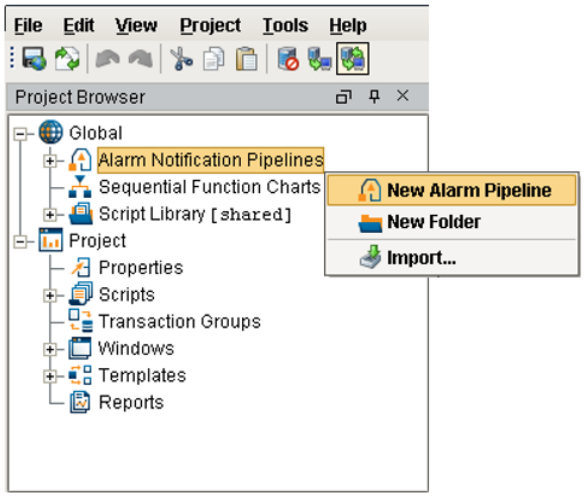
Next...

Alarm Notification Pipelines

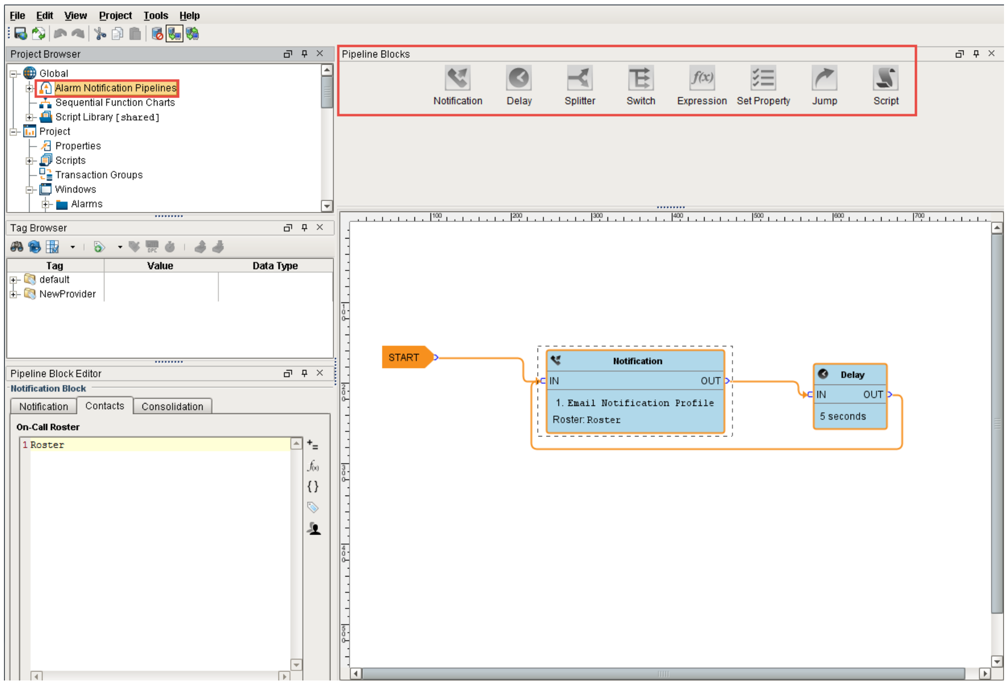
Alarm notification pipelines are logic diagrams that control how alarm notifications are sent out, including who they are sent to and when. They can be used to achieve many advanced alarming scenarios, such as delays, escalation, and parallel delivery. Pipelines provide an easy to use drag and drop mechanism for creating complex notification scenarios.

Creating Pipelines

Before you create a pipeline in the Designer, you need to configure at least one alarm notification profile and at least one roster in the Gateway. Then in the Designer, you create the Pipelines under the **Global** node of the **Project Browser**. Alarm Pipelines, unlike other types of resources such as windows and Transaction Groups, are global resources, and are not part of a project. A defined pipeline runs in the Gateway, and you can see the same pipelines available to edit regardless of the project that is currently open.



When you create a new pipeline, or select one to edit, the Designer Switches to the Pipeline Workspace. The workspace is basically a blank canvas, on which you arrange and connect together various Pipeline Blocks. Each pipeline block has an input, and potentially has outputs. Using the mouse, you can draw connectors between the output of one block into the input of another. The START block, which you will see when you first create the pipeline, cannot be moved or deleted and is the only fixed part of the workspace. It represents the entry point for alarm events into the pipeline, and is treated like an output block.



Multiple outputs can be connected to a single input. Also, it is perfectly acceptable to write the output of a downstream block back to the input of an earlier block, creating a loop. Of course, some care should be taken when doing this, as it can easily result in unexpected behavior, like sending out many emails for one event!

Pipeline Blocks

Pipeline Blocks are the building blocks of the Alarm Pipelines. Each block, depending on the type, will perform some action for the pipeline, such as sending a notification, setting a property, or evaluating an Expression. Refer to the Pipeline Blocks secton for the different pipeline blocks that you can use to build your pipeline.

Pipeline Properties

Pipelines have only a few property settings, but they are important.

To display and modify the properties

Click on the gray area of the Pipeline workspace anywhere there isn't a block.
In the lower-left corner of the Designer, the Pipeline Properties are displayed. They are as follows:

- **Dropout Conditions**
  These dictate when alarms fall out of the pipeline. There are three possible conditions: **Acknowledged**, **Cleared**, and **Shelved**. If any of the selected conditions become true for an event in a pipeline, it will "drop out" entirely, no matter where it is at. No further action will be taken for that event.
- **Pipeline Enabled**
  Indicates whether or not the pipeline is allowed to receive events.

Event Flow

Understanding how events flow through pipelines is crucial to leveraging them fully. Alarms are configured on individual tags, and each alarm can specify a target pipeline for active and clear conditions. When the condition for that alarm is met a new alarm event is generated and sent to the entry point of the associated pipeline. The alarm event then moves through the different logical elements of the pipeline until it finally reaches a pipeline endpoint, or until the alarm matches the pipeline's dropout condition, upon which it then exits the pipeline. There can be multiple alarm events in a pipeline at any given point in time and you can view the current status of the pipeline from the Alarm Pipelines status screen in the Ignition Gateway. It is possible to forward events from one pipeline to another, making it possible to create multipurpose and reusable pipelines.

Each alarm event progresses sequentially through the blocks of a pipeline. However, multiple alarm events can exist in parallel in the pipeline, and new events may enter the pipeline at any time. Some block settings may result in alarm events being held up, such as consolidated notification, but one alarm event cannot affect other events in the pipeline. In this particular case, when multiple events will be collected together for notification, the events that don't drop out of the pipeline after notification are individually forwarded to the output of the block.

At every transition, and occasionally inside of the block execution as well, the dropout conditions will be evaluated. If the event no longer meets the conditions, it will drop out of the pipeline. It is crucial to understand how this works in order to understand pipeline execution. For example, a common scenario is to place a Delay block directly in front of a Notification block. If the delay is 5 minutes, and "Clear" is one of the pipeline drop out conditions, it effectively means that only events that stay active for longer than 5 minutes will actually result in notifications. If acknowledge is one of the dropout conditions, it would mean that operators (or anyone viewing alarm status) would have 5 minutes to acknowledge them, and only then would notifications be sent.

Event Instances and Runtime Properties

As mentioned, a new "alarm event" is generated each time an alarm transitions to active. This event consists of core data, but also has arbitrary properties available on it. The alarm configuration properties are available, as well as associated data, but in addition there are "runtime properties", that only exist while the alarm event is in memory, and are only used by pipelines. These properties can be accessed by several block types, and runtime properties can be created and updated with the "Set Property" block type.

During an alarm event's lifetime, its properties may be updated at any time. When an alarm goes to clear, for example, the system sets information about that state on the alarm event. In the pipeline, if a block accessed the "state" property, it would see a different value than if it had checked just moments earlier.

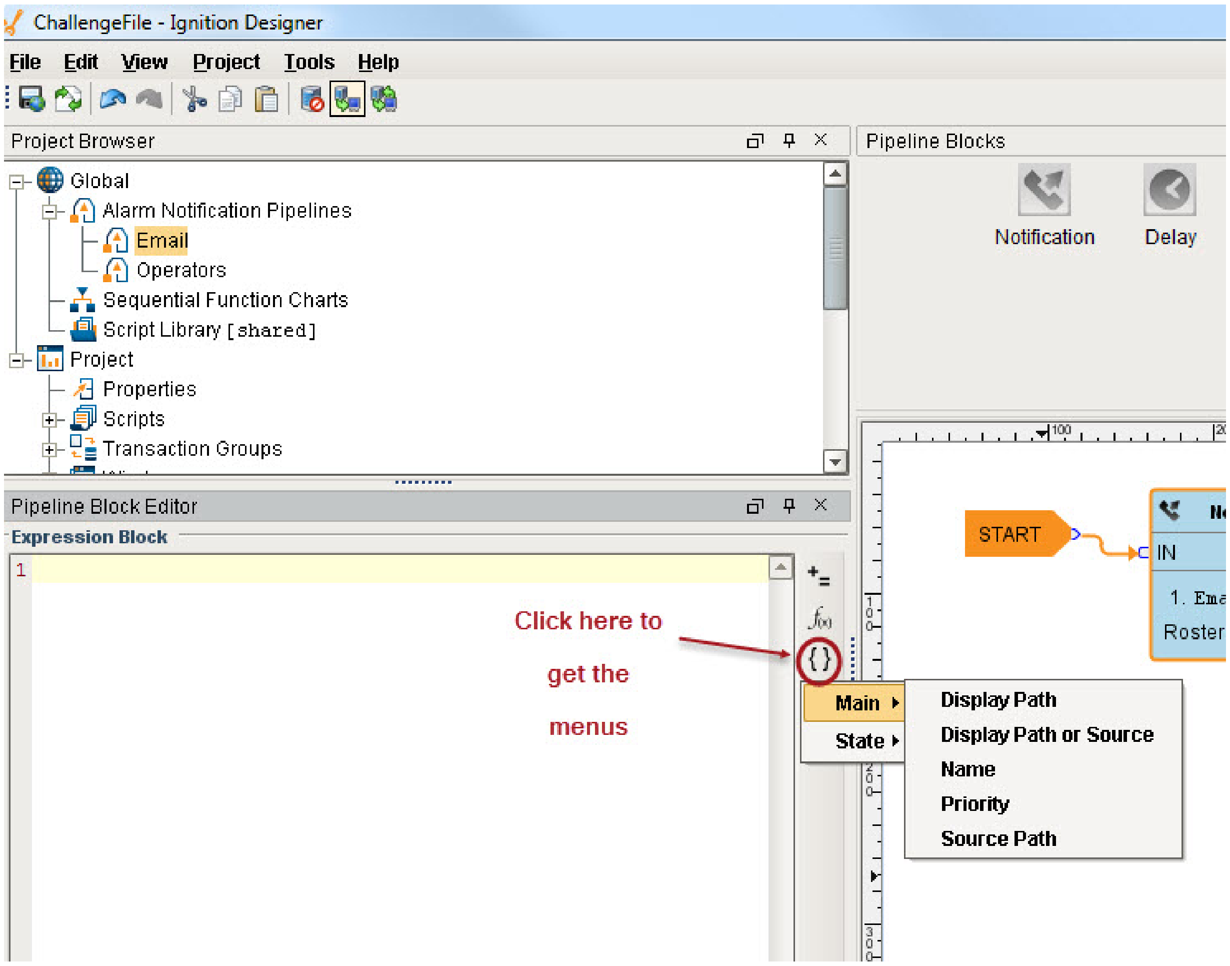
It may be possible for a specific alarm event to exist multiple times in a pipeline. This is especially true when using the Splitter block, which takes events and forwards them concurrently to multiple outputs. When this occurs, the alarm event is "branched" and a new "event instance" is created. Each event instance will start with identical properties, but future modifications of properties will only affect the particular instance. All instances of an event will properly reflect properties set later by the system, such as acknowledgement or clear state.

Pipeline Lifecycle

Given the potentially long running nature of pipelines, it's important to understand how they operate when you edit them. Each pipeline normally has only one instance running at a time, which handles all of the events that go into it. When you edit a pipeline and save, however, a new instance is instantiated. The old instance is "retired", but continues to run until all of the existing events are done. The new instance only receives events generated after the time that it was created, it does not take over the previous instance's events.

Commonly Used Properties

The following screenshot shows how you can access the two commonly used sets of properties in the Expression Block: the **Main** and **State** properties.



The **Main** properties are information about the alarm that are independent of whether the alarm is currently active, clear or acknowledged. The are:

- Display Path
- Display Path or Source
- Name
- Priority
- Source Path

The **State** properties are information about the alarm that change while the alarm is changing, and some of them even change each time the alarm goes from active to clear to ack (acknowledged).

- Ack Time
- Ack'ed By
- Ack'ed By (Name)
- Acked?
- Active Time
- Active?
- Clear Time
- Clear?
- Current State
- Event State
- Event Time
- Event Value
- Is Shelved?

Alarms events are made up of many different properties. Refer to the Alarm Properties section for information on alarm properties and how they are used.

In this section ...

Alarm Pipeline Overview

This section asssumes you have a tag and a notification profile created

To learn more, refer to Understanding Tags and Alarm Notification.

INDUCTIVE UNIVERSIT

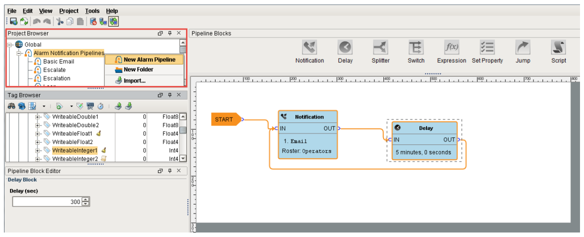
Alarm Pipeline Overview

Watch the Video

Alarm notification pipelines are the bridge between alarms becoming active or clear to messages being sent out to users.   Pipelines are a graphical mechanism for building up logic, providing an easy to use, drag and drop mechanism for creating complex alarm notification scenarios. Using pipelines, it is possible to accomplish many tasks such as alarm escalation, notification consolidation, conditional dispatch and more.

Let's look at how a pipeline works. Pipelines are created in the **Global area** of the Project Browser in the **Alarm Notification Pipeline** section in the Designer. You can right click on Alarm Notification Pipelines to create new pipelines as well as organize pipelines into different folders.

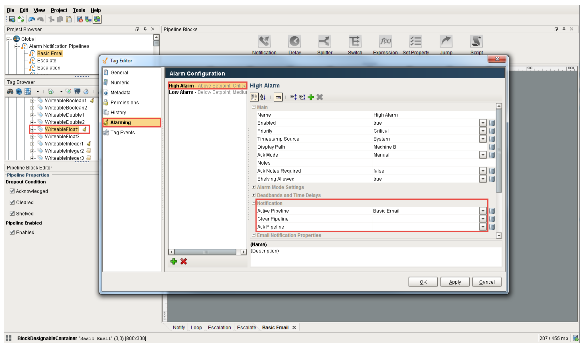
You can build your notification pipeline from the Pipeline Blocks above yourpipeline workspace. The primary goal of alarm pipelines is to notify a group of people about an active alarm.



Every pipeline has a **Start Block**. All alarms are configured to go directly into the Start Block when an alarm becomes Active, Cleared or Acknowledged.

In this example, we have a configured alarm and a pipeline called **Basic Email**. Let's configure the specific alarm to the Basic Email pipeline.

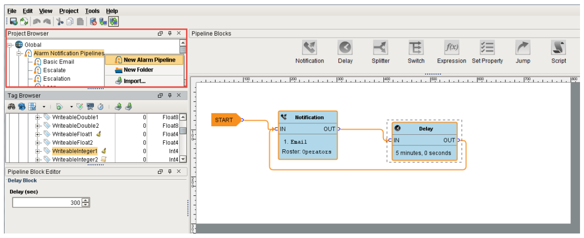
- Click on your **tag**, open the **Alarming section** of the Tag Editor and click on the **'High Alarm.'**
- In the **Alarm Mode Settings**, scroll down to the **Notification** section and select the **'Email Basic'** pipeline from the dropdown list.
- Click **OK.**



In this particular example, when the High Alarm becomes Active, it goes directly to the **Start Block** of the Basic Email pipeline then into the **Notification Block** where it notifies the roster called **Operators** using the **Email** notification profile. From there, the alarm moves to the **Delay Block** and waits 5 minutes to see if the operators acknowledge the alarm or the alarm becomes cleared.

When the alarm is cleared or acknowledged, it drops out of the pipeline. After 5 minutes, if the alarm is still active, it notifies the Operators again. This alarm will continue to loop around in the pipeline as long as it doesn't meet one of the dropout conditions.   As soon as the alarm becomes cleared or acknowledged, it drops out of the Basic Email pipeline.

The next time the alarm goes active, it goes to the Start Block and kicks off the Basic Email pipeline again.



You can use the various pipeline blocks to determine the logic you want to use to notify people of an alarm. Alarm pipelines can be simple or complex based on your business process requirements.

In the next sections, you will learn how to create your own alarm pipelines, filter on alarms, check pipeline status, and much more.

Next...

- Simple Pipeline

Simple Pipeline

The most basic Alarm Notification Pipeline will simply notify users as soon as the alarm becomes active.

Configure Alarm Notification Profile and On-Call Roster

Before you create a pipeline in the Designer, you need to configure at least one Alarm Notification Profile and at least one On-Call Roster in the Gateway.

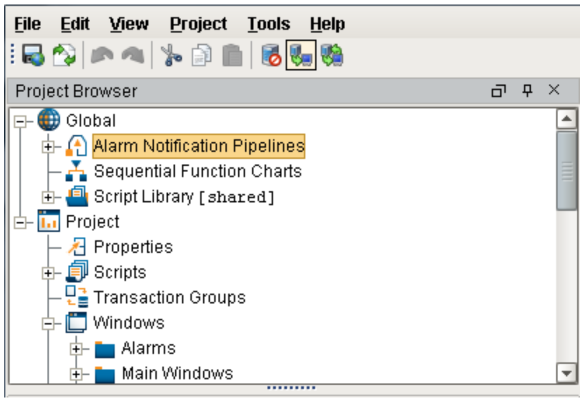
INDUCTIVE UNIVERSITY

Simple Pipeline

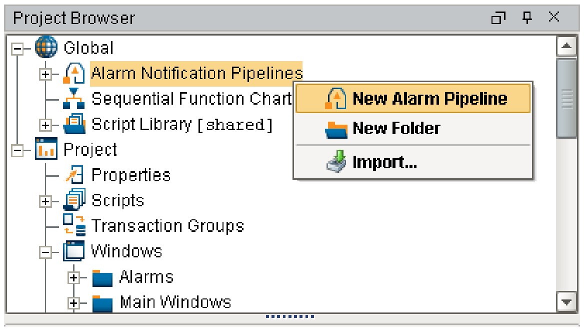
Watch the Video

Let's create a simple alarm notification pipeline in Ignition.

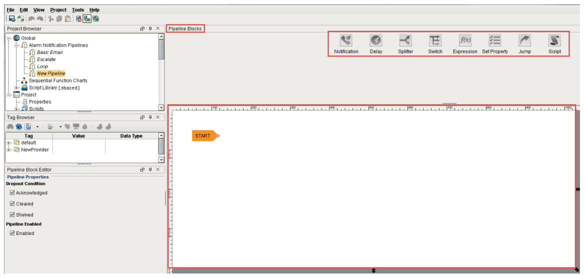
1. In the Designer, create a new pipeline in the **Project Browser** by going to the **Global > Alarm Notification Pipelines** area.



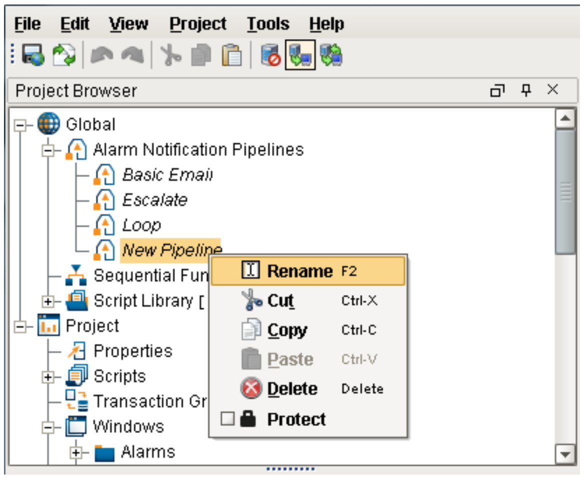
2. Select **Alarm Notification Pipelines** under the Global area, and right click to create a **New Alarm Pipeline**.



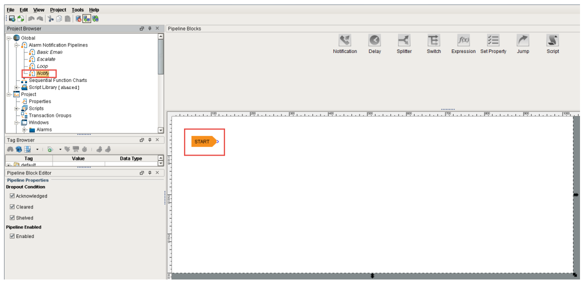
When you create a new pipeline, the Designer switches to the Pipeline Workspace. The workspace is essentially a blank canvas and contains various **Pipeline Blocks** which have specific functionality to help you build your pipeline. By default, the name of the new pipeline is called **'New Pipeline.'**



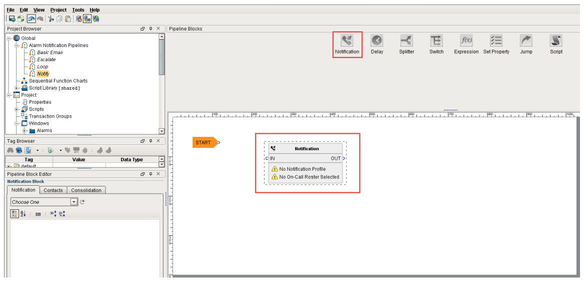
3. You can rename your pipeline to anything you want that best describes your pipeline. Select **your 'New Pipeline,'** right click and **Rename** it to **'Notify.'**



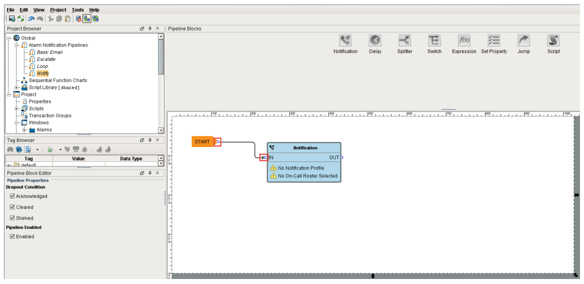
Every pipeline has a **Start Block** which you will see when you create your pipeline. When an alarm occurs, it's going to come to the **Start Block** and from there you can use other blocks to determine what you want to do with that alarm.



4. The most basic pipeline you can create is to simply notify someone right away when an alarm occurs. You will notice that there is a **Notification Block** above the canvas. Drag the **Notification Block** on to your canvas and place it next to the **Start Block**. Once you drag it in to your pipeline, it is not connected yet.

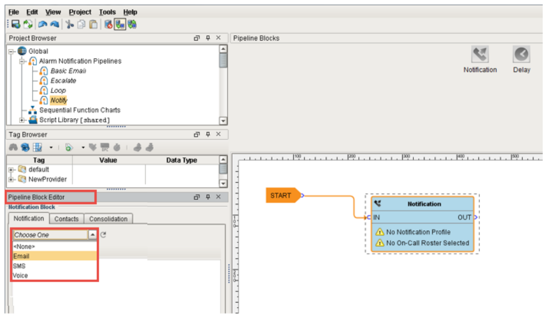


5. Next, drag the output pin from the **Start Block** to the input pin of the **Notification Block**. So now the 2 blocks are connected, and when an alarm occurs, it's going directly into the Start Block and then into the Notification Block.

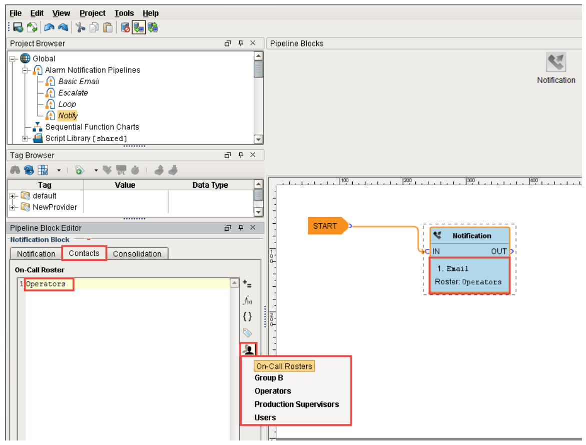


6. On the **Notification Block**, you have to configure the Alarm Notification Profile that you are going use and who you are going to send it too. You should have already configured your Alarm Notification Profile before creating your new pipeline.

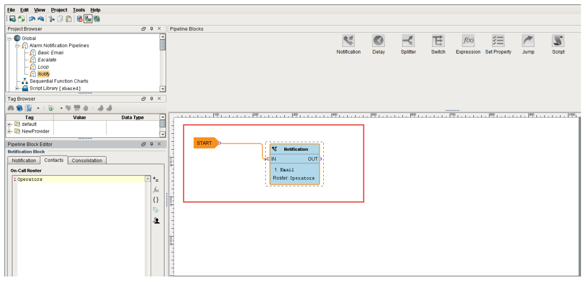
Click on the **Notification Block**, and under the **Pipeline Block Editor** choose which **Notificatio n Profile** you want to use: **Email**, **SMS** or **Voice**. You can see that all 3 profiles are configured, but for this example, choose **Email**.



7. **Under Contacts**, in the **Pipeline Block Editor**, click the **On-Call Roster** icon on the right side. Select from users and groups who you want to send this alarm too.   Your On-Call Roster should have already been created before creating your pipeline.



You just created a very simple Alarm Pipeline.   When this alarm becomes active, it goes directly to the Start Block and notifies all the operators via Email one time.



Pipelines don't do anything until an alarm is told to use it. Once you configure your Alarm Notification Pipeline, the next step is to tell your alarm to use your **'Notify'** pipeline when the alarm becomes **Active**, **Cleared** or **ACKnowledged**. How to make your alarm use a pipeline is covered in the next section.

Next...

- Adding Pipelines to Tags

Adding Pipelines to Tags

Any alarm in Ignition can use an existing alarm notification pipeline. Alarms can share pipelines resulting in a single pipeline providing notification services to many alarms. When an alarm becomes active, cleared, or acknowledged it can execute a designated alarm pipeline. The information about the alarm is ported automatically into the alarm pipeline.

Adding Pipelines to Tags

[Watch the Video]

To associate a pipeline with the alarm

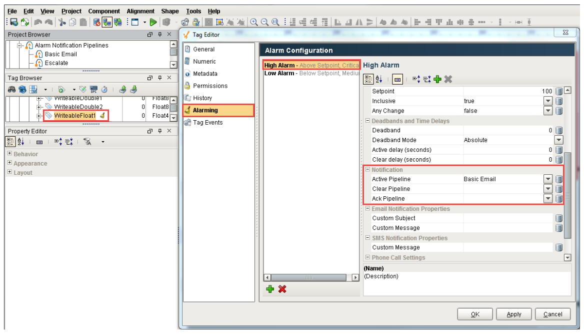
1. Double-click on a tag.
   The **Tag Editor** is displayed and you can access its properties.

2. Go to the **Alarming** tab.

3. Select an Alarm.

4. Under the **Notification** section, select the desired alarm pipeline from the drop-down.

5. Click **OK**.



Setpoint 100
Inclusive true
Any Change false
Deadbands and Time Delays
Deadband 0
Deadband Mode Absolute
Active delay (seconds) 0
Clear delay (seconds) 0
Notification
Active Pipeline Basic Email
Clear Pipeline
Ack Pipeline
Email Notification Properties
Custom Subject
Custom Message
SMS Notification Properties
Custom Message
Phone Call Settings
(Name)
(Description)

Next...

- Pipeline Blocks

Pipeline Blocks

Pipeline Blocks are the building blocks for the Alarm Pipeline Notification system. Each block has an input and zero or more outputs. Each block has its own unique functionality and performs a specific action for the pipeline, such as sending a notification, setting a property, or evaluating an expression.

Let's look at the different pipeline blocks that come with the Alarm Notification system. There are 8 blocks above your Pipeline Workspace that you can choose from to build your alarm notification pipeline.

Pipeline Blocks

Watch the Video

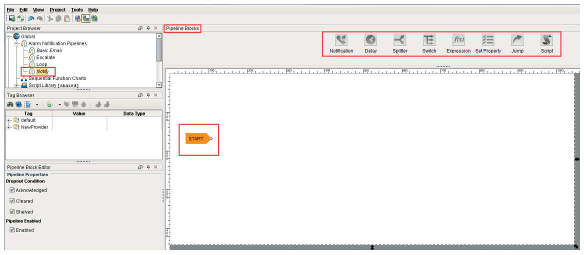
Pipeline Blocks

The **Pipeline Blocks** are described individually in this section, but here is a brief overview:

- **Notification**
  Delivers a notification through the selected Notification Profile (i.e., Email, SMS and Voice).
- **Delay**
  Blocks the alarm event for the specified amount of time.
- **Splitter**
  Forwards a single event concurrently to multiple other blocks.
- **Switch**
  Evaluates a non-boolean Expression, and forwards to an output based on the result.
- **Expression**
  Evaluates an Expression that results in either '`True`' or '`False`,' and forwards the event to that output.
- **Set Property**
  Evaluates an Expression, and sets the result as a runtime property on the alarm event.
- **Jump**
  Forwards the alarm event to a different pipeline.
- **Script**
  Executes a task outside the pipeline.

Start Block

When you create a new pipeline, you always get a **Start Block** placed in your Pipeline Workspace. You cannot delete the **Start Block** because it represents the entry point for alarm events into the pipeline.



Notification Block

As the name suggests, the **Notification Block** is responsible for sending notifications. Each notification block can target one particular notification profile, which represents one method of notification, such as Email, SMS, or Voice. Each type of notification profile will have its own relevant properties for configuration, although some features are common to all profile types. The various profiles will also behave differently, according to their capabilities, in regards to how notifications are ordered. For example, the Email profile is capable of sending a message to many recipients at once, while the SMS and Voice notification profiles must step through each contact sequentially.

Notification Block Basic Setup

There are two required settings for notification blocks: the Notification Profile and the On-Call Roster. The profile will dictate the method through which notifications occur, and the on-call roster will dictate who receives the notifications. The on-call roster provides an ordered list of contacts, which will first be pared down according to each user's schedule. The resulting contacts will be notified in order, based on how the profile operates. The settings displayed will depend on the type of profile selected.

Email Settings

- **From Address** - Email address used for the "From" field.
- **Message** - The body of the email. Like the subject, may refer to properties of the alarm.

  Message now supports HTML formatting. Simply add the <html> tag at the beginning of the Message property. An end tag (</html>) is not required.

  ```
  <html>This is my message. <br>That was a line break.
  ```

- **Consolidated Message** - The message sent when consolidation is turned on, and more than one alarm is being sent. Can refer to properties, and additionally support a special expansion syntax of "{{ ... }}", where everything inside of the double curly braces will be repeated for each alarm in the consolidation group.
- **Test Mode** - If enabled, logs to the console each time an email would be sent, instead of sending the actual email.

Voice Settings

- **Require PIN** - If true, the user will need to enter their PIN in order to hear the alarm notification messages. The user's PIN is defined in the user management section of the Gateway. If false, then anyone will be allowed to hear the messages after answering the phone.
- **Allow Acknowledgment** - If false, users will only be allowed to listen to alarms, not to acknowledge them.
- **Retries Per Contact** - The number of retry attempts per contact per alarm. The default is set to 1.
- **Delay Between Calls** - Introduces a delay between calling each contact, for each alarm. The pipeline dropout conditions are checked regularly between calls and while waiting, so this would provide time for the alarm to drop out before calling the next person. The delay is only enforced after following a "successful" call (a call that was answered). Unanswered or errored calls will move on to the next contact right away. Please note that long delays can block other alarms in the call queue. The delay is applied to all contacts for a particular alarm.

  For most phone systems, a delay of at least 1 second is advised. The system must have time between each call to hang up properly before dialing again.
- **Test Mode** - If enabled, messages will be logged to the console indicating when and to whom calls would have been made, without actually making a call.

SMS Settings

- **Message** - The message to use for single events.
- **Consolidated Message** - Like email, the message to use when multiple events are sent together, due to the block's consolidation settings.
- **Delay Between Notification** - As with voice, a delay between each message sent, to give the recipient time to acknowledge and prevent further notification.
- **Test Mode** - If enabled, logs messages that would have been sent to the Gateway console, and does not actually send them.

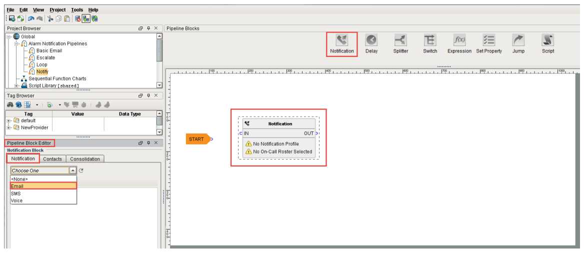
Consolidation

Notification consolidation allows you to limit the number of notifications sent by specifying a delay during which incoming events will be collected. When the delay expires, all events that have arrived during the period of time will be sent together to the notification profile. The manner in which the profile implements consolidation will vary, but in general the result will be a shorter message, or fewer messages, than would have occurred otherwise.

Consolidation is defined with two parameters:

- **Delay** - How long to wait after the first eligible alarm arrives. This setting forces a pause **before** sending a notification. The delay is used to protect against situations where an event might generate many alarms together. In other words, if an alarm comes into the Notification Block, it will be delayed for this amount of time in case there are other alarms arriving soon.
- **Frequency** - The max frequency with which the profile can send alarms. This setting forces a pause **after** sending a notification. The frequency is used to ensure that contacts aren't notified too often, for example if an alarm is rapidly going in and out of the active state.

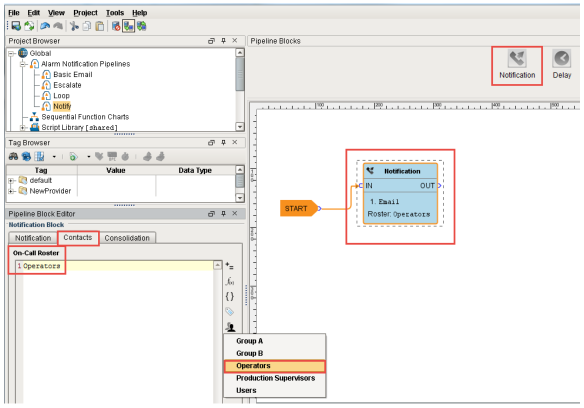
Drag the **Notification Block** to your Pipeline Workspace. Click the **Notification** tab in the Pipeline Block Editor. Choose one of the Notification profiles that you configured. For this example, choose **'Email.'**



Contacts

From the **Contacts** tab, you can specify the **On-Call Roster** who you want to send the alarm notification too. An On-Call Roster must first be created. In this example, several rosters were already configured. Select **'Operators'** as your on-call Roster.

You will notice that the Notification Block is now updated with your **'Email'** profile and **'Operators'** on-call roster.



The Contacts tab now accepts expressions, so it is possible to dynamically select a roster from the pipeline. The expression can make use of tag values as well as alarm properties and associated data.

The alarm properties are accessible from the Alarm Properties button { } located on the right side of the panel. Associated data can be manually entered using the "{" and "}" characters.

```
if({Group}= "A","Group A","Group B")
```

The above expression checks for associated data on the alarm named "Group". If the value is equal to "A", the Group A roster is used, otherwise the Group B roster will be notified.

When should I use quotation marks on the Roster name?

It is recommended to **always place quotation marks around roster names**. However, it is not always required for notifications to be sent out successfully.

If the expression consists of only the name of the roster, then quotes can be omitted. For example, specifying that a Notification block should always use the Roster named "**Operators**" could look like the following with quotes:

```
//This expression will evaluate successfully.
"Operators"
```

the expression would return the string **Operators**, which is a valid Roster.

However, quotation marks could be removed in this scenario. When removed, this causes a syntax error because strings in expressions must always be encased in quotation marks. However, instead of returning an error, the Notification block would assume you simply typed out the name of the roster, so the following example would also succesffully use the **Operators** roster:

```
//This will fail to evaluate as an expression, so the block will instead
look for a roster named "Operators",
//and the users will still be notified.
Operators
```

However, if the expression is any more complicated than the above example, such as using an expression function, then quotation marks must be used. The following example would check the priority of the alarm. If priority was set to "critical", then the **High Priority Roster** would be notified. For all other priorities, the **Low Priority Roster** would be used.

```
//This expression will evaluate successfully.
if({priority} > 3, "High Priority Roster", "Low Priority Roster")
```

However, if the quotes were removed from the roster names, the expression would fail with a syntax error:

```
//This expression will fail, so all the text is assumed to the name of a
Roster
if({priority} > 3, High Priority Roster, Low Priority Roster)
```

When this expression fails, the Notification block will assume that the text is actually the name of a roster, and will attempt to use a roster named "**if({priority} > 3, High Priority Roster, Low Priority Roster)**". For this reason, it is a good idea to always place quotes around Roster names, as well as check your expressions for syntax errors.

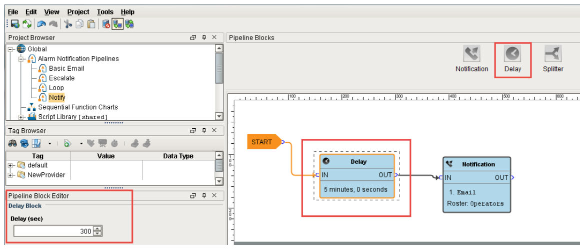
Delay Block

The **Delay Block** simply blocks alarm events for a specified period of time before moving them to the next block. They are generally used to wait for the dropout condition to become satisfied for an alarm.

For example, a 5 minute delay might be used to give operators viewing control screens a chance to acknowledge, and only send notifications if they haven't (the "active delay" deadband on the alarm could be used to delay the alarm as well, but in that case it wouldn't actually be active, and therefore not displayed, for the delay time).

Delays are often also used to control flow in a pipeline. For example, in an "escalation" scenario, a notification block might be used to send emails. Since emails are sent instantly, and acknowledgment occurs "asynchronously" (the user must see the email and click the link or log into the system), a delay block could be used to provide time for the user to acknowledge, before moving on to a voice notification block. There is no practical limit to how long a delay can be. The delay is calculated independently for each event entering the block, meaning that each event will be held for the specified time.

Click the **Delay Block** and a spinner will appear in the Pipeline Block Editor. You can specify the number of seconds to delay. In this example, if you enter **'300'** seconds, the alarm will wait 5 minutes before sending it to the Operators.



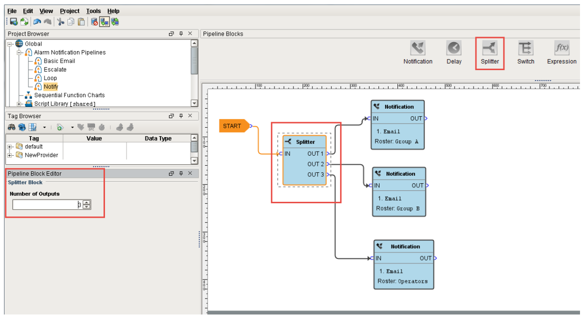
5 minutes, 0 seconds

1. Email
Roster: Operators

Pipeline Block Editor
Delay Block
Delay (sec) 300

Splitter Block

The **Splitter Block** simply forwards a single alarm event to multiple outputs at the same time. It creates a copy of the alarm event for each output and executes the alarm notifications along multiple paths in parallel. You can specify as many outputs as you want. If care is not taken, it is possible to end up with an exponential number of alarm event copies active in the pipelines at one time. Since each notification block operates on a specific on-call roster, Splitters are useful for delivering events to multiple notification blocks at once as shown in this example.

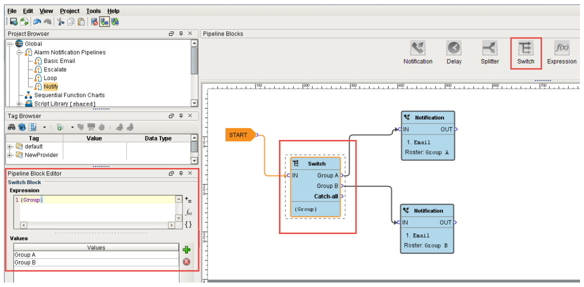


Switch Block

The **Switch Block** evaluates a non-boolean expression and forwards it to an output based on a given result. It can be a number or string, and it will be looked up against a defined list of expected results. If a match is found, the alarm will follow that path, otherwise the "Catch-all" output is used.

You can create an expression and even use the alarm event's priority, associated data, value and more. The following example has associated data for the 'Group' that they belong to with 'Group A' and 'Group B' values.   The Switch Block will evaluate the expression results against these defined values, and if a match is found, the alarm will follow that path. If neither of the values are met, the Catch-all output will be used.

This example demonstrates that a pipeline can send out an alarm to 2 different lists of people based on the **'Group.'**



Expression Block

The **Expression Block** contains an Expression which is evaluated against each alarm that enters it. The result is expected to be a boolean value, either 'True' or 'False' and forwards the alarm to the specific output. The Expression executed by the block is written in the same syntax as other Expressions in Ignition. However, in this context, it is possible to refer directly to properties of the alarm event that is being evaluated by using the standard "{path}" reference syntax. In addition, the following functions are available to quickly determine the state of the alarm.

Examples

`isActive()`

This single function returns whether the current event is active or not. An Expression block like this could be used to then dispatch to an "active" pipeline, and a "clear" pipeline (both the active and clear pipeline settings on the alarm would be set to this dispatch pipeline). This kind of setup allows you to later modify how actives are handled vs. clears in one place, without having to modify many alarms.

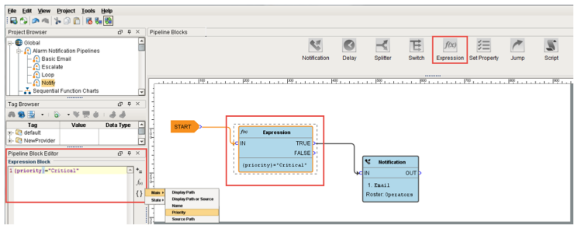
`toInt({priority})>2 && {displayPath} like "*East Area*"`
This block would forward all High and Critical alarms from the "east area" to the true path. The others would go to false, which may not actually be bound to another block, making this Expression block act like a filter.

`isPropertyDefined("isEscalated")`
This Expression checks if a property exists on the event. The "isEscalated" property is not a system defined property. Instead, in this example, it might be set using a Set Property block before forwarding back to the start of the pipeline. The first time through, this Expression would fail, but the next time, it would pass, and the "escalated" behavior could be executed.

In the **Pipeline Block Editor**, you can select the Alarm Properties from any of the icons in the Expression Block. This example uses an expression to check to see if the priority is 'Critical,' and if so, the alarm notification will follow the 'True' path. If it is not 'Critical', it will follow the 'False' path. As you can see, the 'False' output is not configured. You don't have to configure every output if you don't want too.

Priority Expression Example

```
{priority}="Critical"
```



Set Property Block

The **Set Property Block** allows you to define or redefine a property on the alarm event. The value is created from the Expression Block, which can refer to other properties or the same properties. Typically, the Set Property Block is used as a counter for the number of times the alarm has been looping or notifying people in your on-call roster. Settings modified in this way will only exist while the alarm is in the pipeline, they will not be stored to the journal, or show up in the status table.

Examples

If you want to attempt notification up to three times before stopping, you could create a pipeline that looked like [Notification Block] > [Set Property] > [Expression], with the Expression looping back to the notification block (perhaps with a delay in between).

The Set Property block would look like:
Property Name: counter
Expression: coalesce({counter}, 0)+

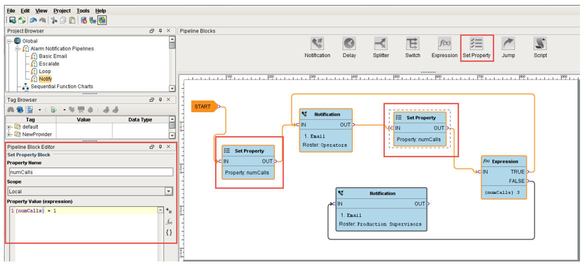
Note that the first time the block is hit, the `counter` property will not exist, and therefore the value will be null. The `coalesce` function returns the first non-null value, allowing us to start with 0. Another way to write this might be:

```
if(isPropertyDefined("counter"), getProperty("counter"), 0)+1
```

The `getProperty()` function is functionally equivalent to simply referencing the property in brackets (that is, `{counter}`).

The Expression block in this example would simply be: `{counter}`<3.

Here is another example where the operators are notified of an alarm event 3 times before being escalated to the Production Supervisors. Drag 2 **Set Property Blocks** to your pipeline workspace. Assign the first Set Property block to a Property Name **'numCalls'** and a Property Value of **'0 .'** The second Property Block you need to increment the counter. Enter **'{numCalls} + 1'** as shown in the Pipeline Property Editor. Each time the alarm goes through this block, the 'numCalls' variable increments by '1.' If no one acknowledges the alarm, it is sent to the Production Supervisor's on-call roster.

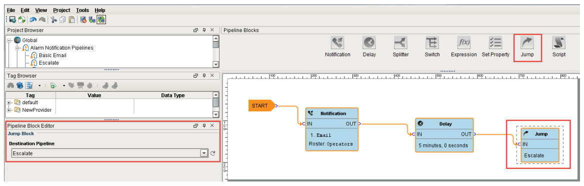


Jump Block

The **Jump Block** forwards an alarm event to a different pipeline. You can breakup a pipeline into several different pipelines and use the Jump Block to go back and forth between pipelines. This is perfect for an escalation pipeline because often times they can become one massive pipeline, and the Jump Block can make the escalation pipeline less complicated and more compact.

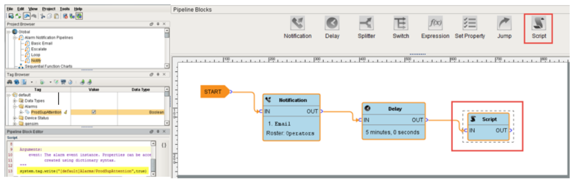
In the Pipeline Block Editor, enter the Destination pipeline called **'Escalate.'**

In this simple example, the **Jump Block** forwards the alarm to the Escalate pipeline and ends there.



Script Block

The **Script Block** allows you to execute a task outside the pipeline like writing to another alarm tag or database. In this example, the Script Block was used to notify the production supervisors of an alarm using another alarm tag as shown in the Pipeline Block Editor.



In the next sections, you will see how to use a combination of these different pipeline blocks to create your own unique pipelines.

Next...

- Pipeline - Filter on Alarm Priority

Pipeline - Filter on Alarm Priority

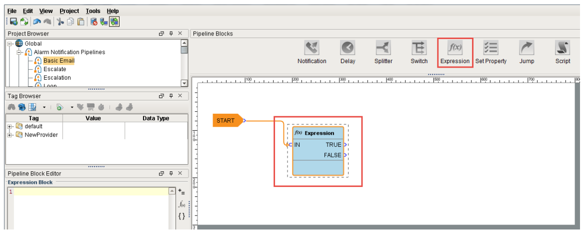
This section assumes that all alarms are configured

To learn more, go to Configure Alarm on a Tag, Alarm Notification and On-call Rosters.

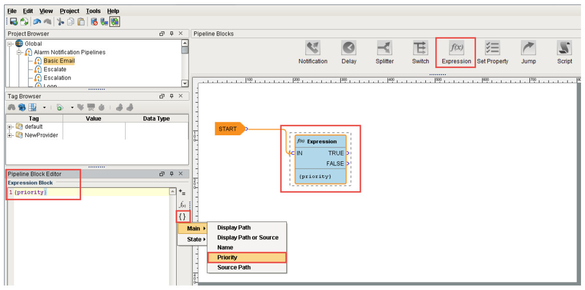
You can create a filter for an alarm priority in an alarm notification pipeline that notifies a different list of people based on the alarm's priority.

1. There are two pipeline blocks that can be used for filtering, either the **Switch Block** or the **Expression Block**. The **Switch Block** allows you to switch on the priority, doing something different for every single priority that exists. The **Expression Block** allows you to check for a specific priority, and it returns either a 'True' or 'False' value.   You can do something if it is 'True' and something else if it is 'False.' This example uses the Expression Block.

   Drag in the **Expression Block** in to your pipeline, and connect the **Start** to the **IN** of the Expression block. You will notice that the Expression block is going to return a **'True'** or **'False'** value.



2. In this expression, you want to check for the alarm's priority. Go to the **Pipeline Property Editor**, click the **curly brace icon > Main** and select the **'Priority'** expression. **Priority** is a placeholder until you enter the priority value you want to filter on. When the alarm is triggered, it will be replaced with the actual priority of that alarm.



3. Alarm priorities have predefined values for filtering and sorting. In this example, we are going to filter for all alarms with a priority of 'Critical.' **Critical** has a value of **4**.

Alarm Priorities

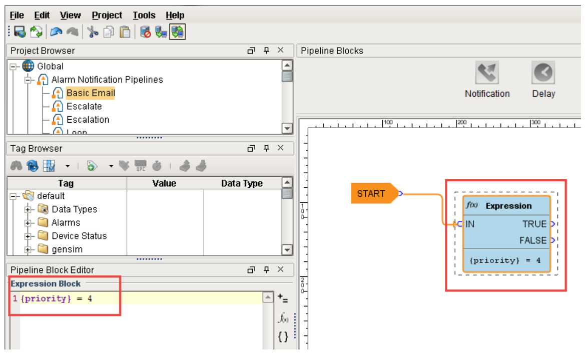
0 = Diagnostic
1 = Low
2 = Medium
3 = High
4 = Critical

Refer to Alarm Properties for more information.

If the alarm priority is **Critical**, the alarm will go to the output pin of the **'True'** value in the Expression block. If the priority is not **Critical**, the alarm will go to the output pin of the **'False'** value. Notice, we are doing something different for each of these conditions.

Pipeline - Filter on Alarm Priority

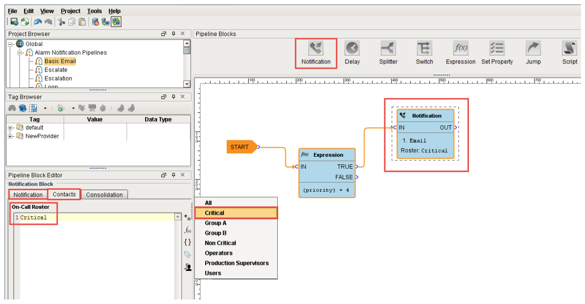
Watch the Video



```
{priority} = 4
```

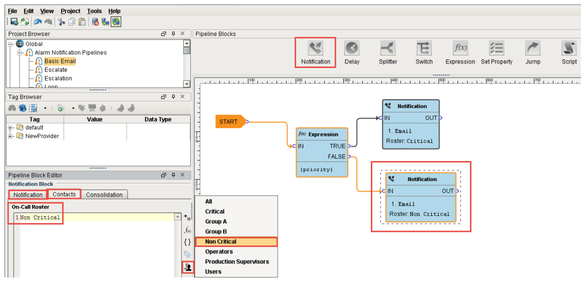
4. Drag in a **Notification Block,** and connect the **'True'** output pin into the **IN** of the Notification block.

   In the **Pipeline Block Editor** under the **Notification tab**, choose the notification type. On the **C ontacts tab**, under the **On-call Roster icon**, select the **'Critical'** list of people you want to email.



5. Drag in a second **Notification Block,** and connect the **'False'** output into the **IN** of the Notification block.

   In the **Pipeline Block Editor**, under the **Notifications tab**, choose the notification type. In the **Contacts tab**, under the **On-call Roster icon**, select the **'Non Critical'** list. The Critical list is a different list of people from the Non Critical list.



As you see in this example, the alarm notification pipeline notified different lists of people based on the alarm's priority.

Next...

- Pipeline - Filter on Alarm Associated Data

Pipeline - Filter on Alarm Associated Data

This section assumes you have certain objects configured

The examples in this section assume you have a configured alarm, alarm associated data, a notification profile, and an existing on-call roster.

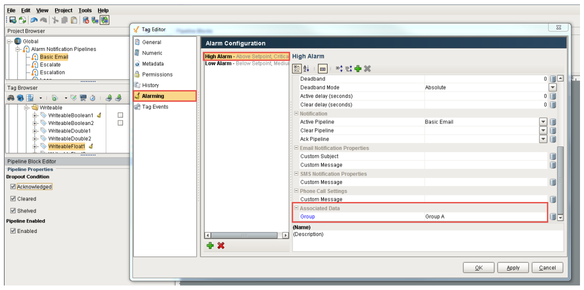
Pipeline - Filter on Alarm Associated Data

Watch the Video

Once associated data is added to an alarm, you can easily filter for associated data in an alarm pipeline. It's possible to add associated data that can be either static or dynamic.

Let's look at how to filter a notification pipeline based on associated data we added to an alarm.

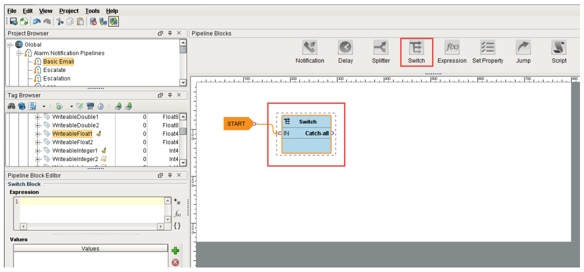
For this example, the WriteableFloat1 tag is used and has 2 alarms setup; High Alarm and Low Alarm. You can see the **'High Alarm'** configuration has **Associated Data** called **'Group,'** and a value of either Group A or Group B. We are going to use this associated data to filter the alarm notification pipeline.



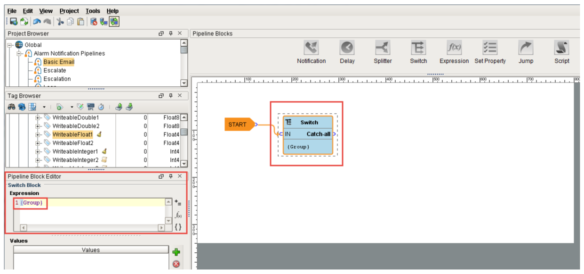
Let's create a pipeline that's going to filter on associated data.

1. There are two blocks that can be used for filtering, either the **Switch Block** or the **Expression Block**. The **Switch Block** allows you to switch on a value and do something different for Group A and Group B. The **Expression Block** does the same thing, but only returns values of 'True' or 'False.' This example uses the Switch Block.

   Drag in the **Switch Block** to your pipeline, and connect the **Start** to the **IN** of the Switch block.

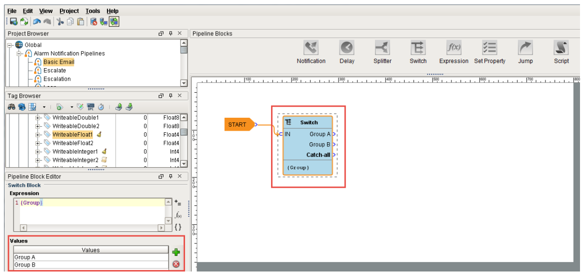
   

2. In the **Pipeline Property Editor**, you have to manually enter the Expression because you will not find any of the associated data properties in the dropdown list using the curly brace icon. Enter the name of the associated data property in curly braces as follows, **'{Group}.'** The **'Group'** associated data property acts as a placeholder for the real values.

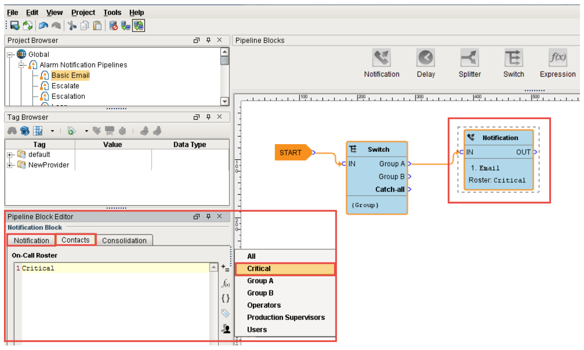
3. Next, you need to add some values to the Expression Block. You can add as many values that you have values for. Click on the **green plus icon** and add 2 values: **'Group A'** and **'Group B.'**

   Once you enter both values, you will have outputs in the Switch block for **Group A**, **Group B,** and **Catch-all**. So now you can notify a different list of people based on each Group.



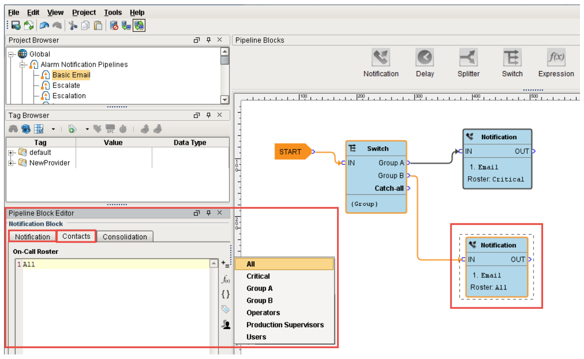
4. Drag a **Notification Block** on to your pipeline, and connect the output of **Group A** in the Switch block to the **IN** of the Notification block.

   In the **Pipeline Block Editor** under the **Notification tab**, choose your notification type. In the **C ontacts tab**, under the **On-call Roster icon**, select the roster you want to notify.



5. Drag a second **Notification Block** to your pipeline, and connect **Group B** from the Switch block to the **IN** on the second Notification block.

   Choose your notification type and specify a different On-call Roster to notify a different group of people.



So, what we have successfully done here, is notify people on the 'Critical' list based on Group A, and the 'All' list based on Group B by filtering on alarm associated data.

Next...

- Pipeline - Escalation

Pipeline - Escalation

It is possible to setup escalation procedures in alarm notification pipelines in Ignition. The following example will create a pipeline that notifies operators 3 times. If a user does not acknowledge the alarm after the third time, the alarm escalates to the Production Supervisor's on-call roster.

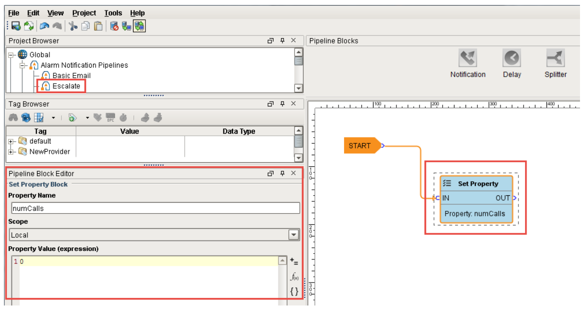
The following example requires that two seperate on-call rosters have been configured ("Production Supervisors" and "Operators"), as well as an email notification profile.

Pipeline - Escalation

Watch the Video

1. In the **Project Browser** under the **Alarm Notification Pipelines**, create a New Alarm Pipeline called **'Escalate.'**

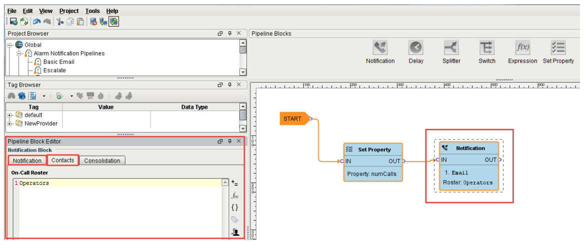
2. In order to keep track of how many times the operators get notified, create a counter using the Set Property block inside your pipeline. Drag the Set Property Block to the Pipeline Workspace and connect the **START** block to the IN of the Set Property block.
   In the Pipeline Block Editor, set the **Property Name** to **'numCalls'** and initialize the **Property Value** to **'0.'**



Pipeline Block Editor
Set Property Block
Property Name: numCalls
Scope: Local
Property Value (expression): 0

3. Drag the **Notification Block** to the Pipeline Workspace and connect the OUT from the Set Property block to the IN of the Notification block.

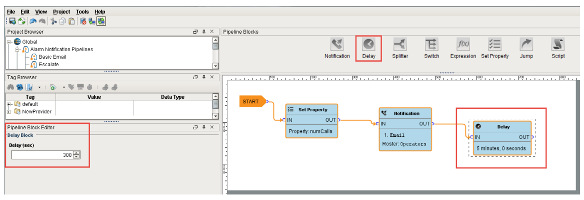
   In the **Pipeline Block Editor**, set the Notification to use **'Email'** and set the Contacts to use the **'Operators'** on-call roster.



Pipeline Block Editor
Notification Block
Notification | Contacts | Consolidation
On-Call Roster: Operators

4. Let's add in a Delay Block so a little time is given to the operators to respond to the alarm before immediately sending back to the operators or escalating it the production supervisors.

   Drag in the **Delay Block** and connect the **OUT** from the Notification block to the **IN** of the Delay block. In the Pipeline Block Editor, set the **spinner** for **'300' seconds**. It will give the operators 5 minutes to respond to the alarm before going to the next block.

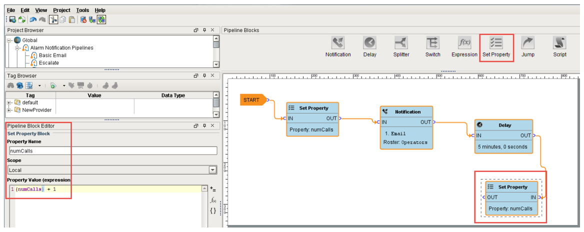


Pipeline Block Editor
Delay Block
Delay (sec): 300

5. If the alarm is still active and not acknowledged, the counter will increment by 1. In order to increment the counter, drag in another **Set Property Block** and connect the **OUT** of the Notification block to the **IN** of the Set Property block. You can right click on the Set Property block to **reverse the orientation**.

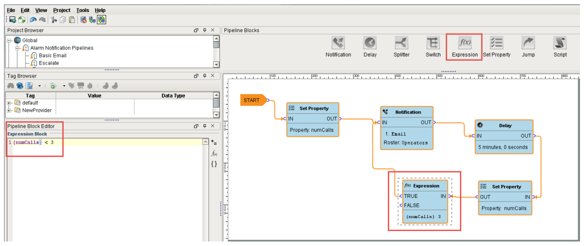
   In the Pipeline Block Editor, set the **Property Name** to **'numCalls.'** In the **Property Value**

Expression type in, **'{numCalls} + 1.'** In this expression, use the curly braces to refer to that variable so the counter gets incremented by 1.



6. The **Expression Block** will check to see if **'numCalls' is < 3.'** Drag in an Expression Block to your pipeline workspace and right click on the Expression block to reverse the orientation. Connect the **OUT** of the Set Property block to the **IN** of the Expression block.

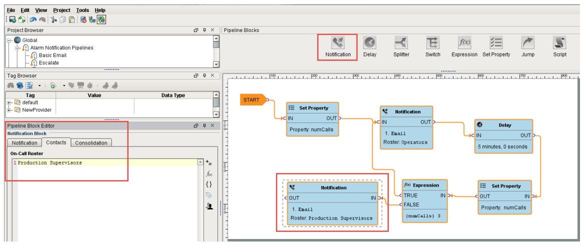
   Since the value of 'numCalls is < 3,' the alarm event will follow the 'True' path. Connect the **'True'** pin to the **Operator Notification block**.



7. If the **'numCalls is > 3,'** the alarm event will follow the 'False' path and it will escalate to the Production Supervisors.

   Drag in another **Notification Block** and reverse the block orientation. In the Pipeline Block Editor, set the Notification to use **'Email'** and set the Contacts to use the **'Production Supervisors'** on-call roster.

   Connect the **'False'** pin to the **IN** of the **Production Supervisors Notification block**.



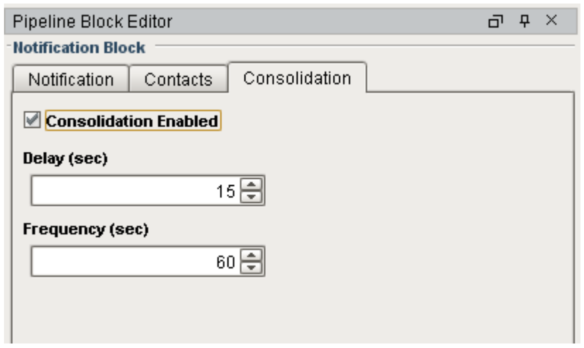
As you can see, when the Escalate Pipeline sets the 'numCalls' variable to a value of 0, it's going to notify the Operators on-call roster through email that the alarm is active. It will wait 5 minutes, and if the alarm is still active, it will increment the counter by 1. This will repeat until the 'numCalls' is > 3. After the 3rd time, if the alarm is still active and in the pipeline, then the Production Supervisors will be notified. Keep in mind, that if the alarm at any point meets the dropout conditions, it will not go to the next block of the pipeline.

Next...

- Pipeline - Notification Block Consolidation

Pipeline - Notification Block Consolidation

Notification consolidation allows you to limit the number of notifications sent by specifying a delay during which incoming events will be collected. When the delay expires, all events that have arrived during the period of time will be sent together to the notification profile.



☑ Consolidation Enabled

Delay (sec)

15

Frequency (sec)

60

Pipeline - Notification Block Consolidation

Watch the Video

For Example:

Once an alarm enters the Notification block in this pipeline, it will wait for **Delay** number of seconds (in this case 15 seconds) before sending the alarm. If additional alarms enter this notification block, it will send using the **Consolidated** Message/Subject from the **Notification** tab depending on your notification type.

Once an alarm has been sent, whether it was a consolidated notification or not, this Notification Block will wait for **Frequency** number of seconds (in this case 60 seconds) before it can send another email.

Note that Consolidation only effects this pipeline. Any notification blocks in other pipelines will not be effected.

Next...

- Pipeline Status

Pipeline Status

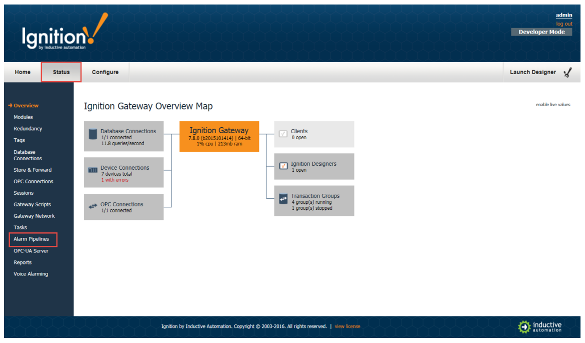
INDUCTIVE UNIVERSITY

Pipeline Status

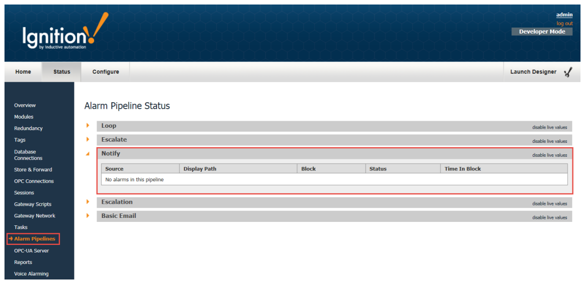
Watch the Video

It's possible to monitor the status of your alarm notification pipelines to see exactly where your alarms are in the notification process in Ignition. You can do this from the **Ignition Gateway Webpage** in your browser.

Go to the **Status tab** of your Gateway webpage and on the left side of the navigation bar, click on **Alarm Pipelines**.



The Alarm Pipeline Status shows all the pipelines you created including all the active alarms and where they are in the system. In this example, we have several pipelines, but let's look at the one called **Notify**. As you can see, there are no alarms in this pipeline.

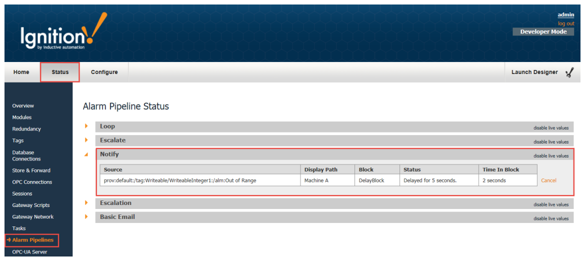

Alarm Pipeline Status

Loop
Escalate
Notify

Source	Display Path	Block	Status	Time In Block
No alarms in this pipeline				

Escalation
Basic Email

In this next example, you can see that there is one active alarm in the Notify pipeline. An Out of Range alarm was triggered for Machine A, and it's currently in the Delay block. As this alarm moves through the pipeline, you can monitor its status. You can even **Cancel** the alarm to delete it from the pipeline.


Alarm Pipeline Status

Loop
Escalate
Notify

Source	Display Path	Block	Status	Time In Block	
prov:default:/tag:Writeable/WriteableInteger1:/alm:Out of Range	Machine A	DelayBlock	Delayed for 5 seconds.	2 seconds	Cancel

Escalation
Basic Email

How long the alarm stays in the in the Alarm Pipeline Status depends on how the pipeline is configured, the type of notification (i.e., Email, SMS or Voice) and what the pipeline is doing (i.e., notifying users, looping, executing a script, etc.).

Next...

Customizing Alarm Notification

When Ignition sends out either an email or a voice notification, by default, there is a built-in script for the email body and a text-to-speech engine for the voice content. You can override the default on each alarm, by simply editing your alarm and setting the custom scripts.

In this section ...

Custom Email, SMS, and Voice Scripts

Introduction

Ignition has a built-in script that it uses when it sends out an alarm notification via Email, SMS, and Voice. There is a different script for each notification type. Each notification type has it's own relevant properties for configuration, although, some features are common to all profile types that you will soon discover.

Custom Email, SMS, and Voice Scripts

Watch the Video

Alarm Notification Setup Requirements

Email, SMS and Voice scripts in Ignition are extremely easy to use and setup. Before you begin using alarm notifications and customizing message scripts, you need to have Email, SMS and Voice modules installed, alarm notification profiles created, and settings configured. It's best to run through these alarm notification requirements before setting up your custom scripts so you have a smooth implementation of alarm notifications. You only need to install the modules and the profiles that you plan to use.

The following modules can be found on the Ignition Downloads page.

- Email Notification Module
- SMS Notification Module
- Voice Notification Module
- Text-to-Speech Voice (language)

Refer to each alarm notification profile below for how to create and modify a profile.

- Email Notification Profile - **Two-Way Acknowledgement** is enabled here if you want users who receive the alarm to acknowledge the alarm.
- SMS Notification Profile - **Two-Way Acknowledgement** is enabled here.
- Voice Notification Profile - Acknowledgement of the alarm is part of the call script, and provided as an instruction as part of the call.
- User Profile - when setting up a Voice alarm notification, each user's profile requires a contact telephone number and a Security PIN.
- Alarm Notification Pipelines - controls how alarm notifications are sent out.
- On-Call Rosters - are the group of users that will receive the alarm notifications. On-Call Rosters are available in the Contacts tab of the Notification Block.

Methods of Customizing Message Scripts and Sending Notifications

You can configure and customize scripts for Email, SMS and Voice in two locations; in the Alarm Configuration section of the Tag Editor, and in the Notification Block of the alarm pipeline.

Every Notification Block in a pipeline has to have a message to send out. Here you have all the alarm properties to select from to build and test your script. You can specify your Contacts, and enable Consolidation all from the same screen.
The Alarm Configuration has fields to override the message sent from a Pipeline's Notification Block. This is particularly useful if your alarm has a dynamic Pipeline or is being sent to multiple Pipelines. If you're creating your alarm notification for the first time, make sure to specify a Pipeline for the alarm to use.

The alarm's Custom Message property takes precedence over any Notification Block in a pipeline.
There are two methods of sending out alarm notifications: one notification per alarm, and notification consolidation (i.e., multiple alarms in a single Email or SMS). By default, alarm notifications are sent out one message for each alarm.

If you want to use Consolidation, you can use the default Consolidated Message script or create a custom script using the alarm properties. Consolidation is enabled in the Notification Block. When enabled, multiple alarm events are sent together in one message.
Each alarm notification type has a default message script. The Email and SMS default message script is found in the Notification Block shown below. For Voice, the default call script is located in the Voice Notification Profile.

Default Message Script

```
 At {eventTime|hh:mm:ss}, alarm "{name}" at
"{displayPath}" transitioned to {eventState}
```

You can override the default message script on a per alarm basis by simply creating a Custom Message script.

Custom Message always takes precedence over the default message

All alarm notification types allow you to define a custom message. If a custom message is created, it will be used instead of the default message in the script.

Custom Message Scripts

The following sections describe how to use the default message scripts, build your own custom message scripts, and send out alarm notifications for Email, SMS, and Voice.

Email

Email notifications are the most common method of sending alarm notifications.

Customizing Email Message Scripts using Alarm Configuration in the Tag Editor

1. To customize an Email alarm notification, select your tag, and go to the **Alarm Configuration** sc reen.   Scroll down and you will see several sections; **Notifications**, **Email Notification Properties, SMS Notification Properties**, and **Phone Call Settings**.

2. In the **Notification** section, you can specify the name of the alarm pipeline that you want to trigger when an alarm becomes Active, Cleared and Acknowledged. Simply enter the name of the alarm pipeline next to the relevant state you want the email sent out from. This example uses the **'Basic Email'** pipeline, and the alarm is triggered on the **'Active'** state.

3. Enter a **Custom Subject** called **'Ignition Alarm.'**

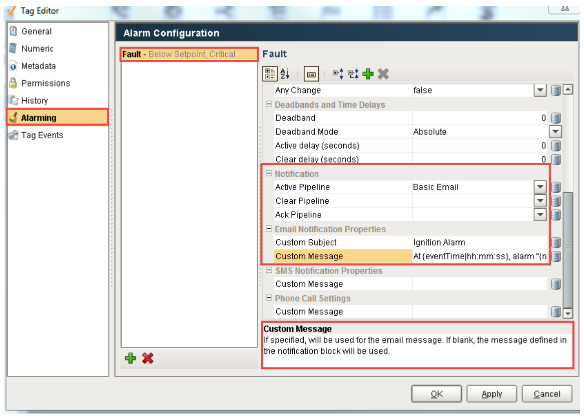
4. Create your **Custom Message**. You can use static text, bring in tag values and special alarm properties. You can even bind a particular property to an expression to concatenate a string that using other values that are in the system. This example displays the alarm name, state, and time in hours, minutes, and seconds.

   Custom Message Script

   ```
   The "{name}" alarm became {eventState} at {eventTime|hh:mm:ss} with
   a value of "{Value}"
   ```

   Notice that when you select any fields in the Alarm Configuration, the Description Area at the bottom of the window displays details about the property's behavior.

5. When complete, press **Save**.



Any Change	false
Deadband	0
Deadband Mode	Absolute
Active delay (seconds)	0
Clear delay (seconds)	0
Active Pipeline	Basic Email
Clear Pipeline	
Ack Pipeline	
Custom Subject	Ignition Alarm
Custom Message	At {eventTime|hh:mm:ss}, alarm "{n

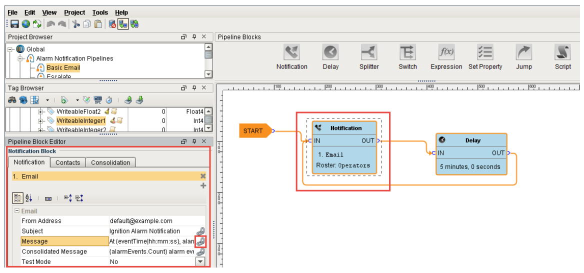
Custom Message
If specified, will be used for the email message. If blank, the message defined in the notification block will be used.

6. In the Pipeline **Notification Block**, go to the **Contacts** tab, and select the **On-Call Roster** that you want to send alarm notifications too.

7. If Consolidation is required, enable **Consolidation** and edit the **Delay** and **Frequency** settings. For more information on the Delay and Frequency settings, refer to Consolidation.

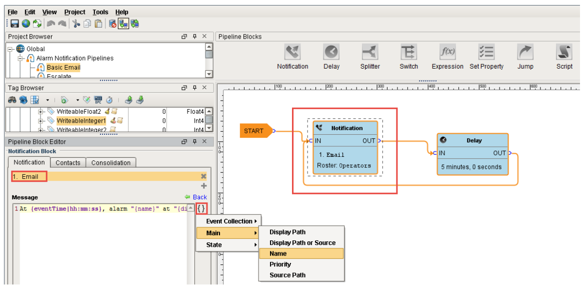
Customizing Email Message Scripts using the Pipeline Notification Block

1. Click on the **Notification Block** of your pipeline, and choose **Email**. You can see all the default Email settings for each field in the Notification Block. Click on the **Link** icon next to the **Message** field to expand the script.



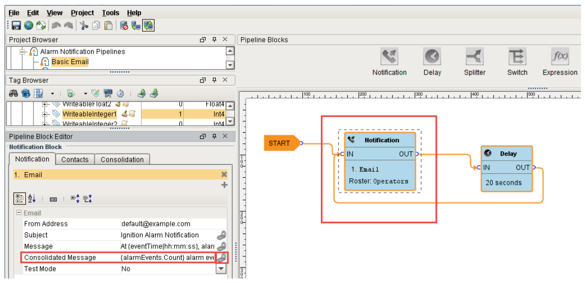
From Address	default@example.com
Subject	Ignition Alarm Notification
Message	At {eventTime|hh:mm:ss}, alarm
Consolidated Message	{alarmEvents.Count} alarm eve
Test Mode	No

2. Here you can customize the default message script using the built-in alarm properties. Click on the **Curly Braces** icon to see a list of properties to help you build your custom message script.



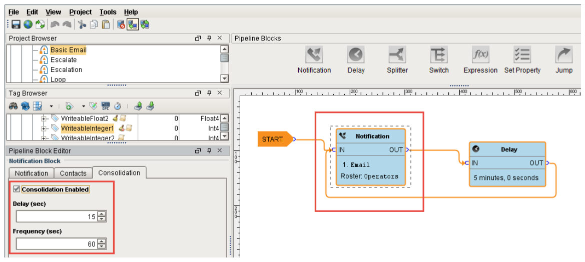
```
At {eventTime|hh:mm:ss}, alarm "{name}" at "{di
```

3. If you want to use Consolidation, you can use the default **Consolidated Message** script or create a custom script using the alarm properties. Click on the **Link** icon to expand the Consolidated Message script.



From Address	default@example.com
Subject	Ignition Alarm Notification
Message	At {eventTime|hh:mm:ss}, alarm
Consolidated Message	{alarmEvents.Count} alarm ev
Test Mode	No

4. If required, go to the **Consolidation** tab, and enable **Consolidation**. When enabled, multiple alarm events are sent together in one email which results in fewer messages being sent out. For more information on the Delay and Frequency settings, refer to Consolidation.

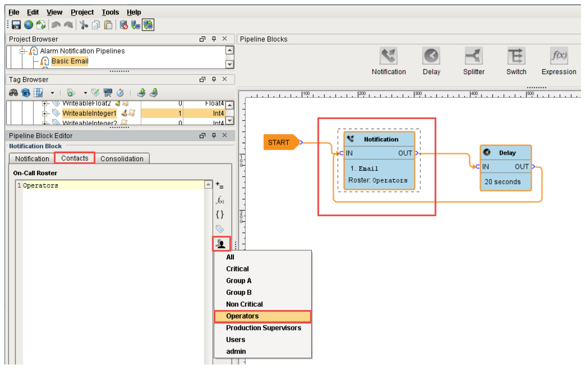


Consolidation Enabled ✓

Delay (sec) 15

Frequency (sec) 60

5. Next, from the **Contacts** tab, choose the **On-Call Roster** that you want to send email notifications too.



On-Call Roster

1 Operators

SMS

SMS works the same way as Email except for a few subtle differences.

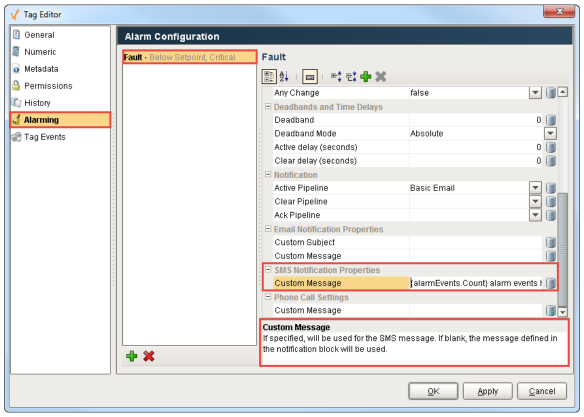
Customizing SMS Message Scripts in the Alarm Configuration of the Tag Editor

1. To customize an SMS alarm notification, select your tag, and go to the **Alarm Configuration**. Scroll down to **SMS Notification Properties.**

2. In the **Notification** section, specify the name of the alarm pipeline that you want to use when the alarm becomes Active, Cleared and Acknowledged. In this example, the name of the pipeline is **'Basic Email'** and is triggered on the **'Active'** state.

3. Enter your **Custom Message** script or use the default script. You will notice there is no Subject, only a Custom Message.

4. When you're finished customizing the alarm configuration, click **Save**.



Tag Editor — Alarm Configuration

Fault - Below Setpoint, Critical

Any Change	false
Deadbands and Time Delays	
Deadband	0
Deadband Mode	Absolute
Active delay (seconds)	0
Clear delay (seconds)	0
Notification	
Active Pipeline	Basic Email
Clear Pipeline	
Ack Pipeline	
Email Notification Properties	
Custom Subject	
Custom Message	
SMS Notification Properties	
Custom Message	{alarmEvents.Count} alarm events h
Phone Call Settings	
Custom Message	

Custom Message
If specified, will be used for the SMS message. If blank, the message defined in the notification block will be used.

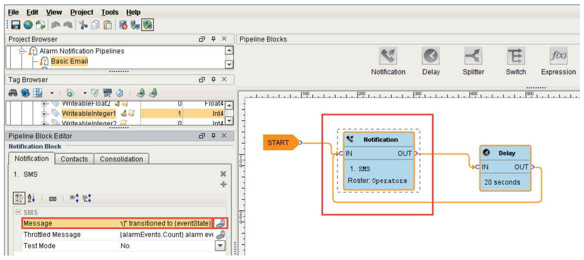
5. Go to the **Pipeline Notification Block** and from the **Contacts** tab, choose the **On-Call Roster** t hat you want to send alarm notifications too.

6. If required, go to the **Consolidation** tab, and **enable Consolidation**. You can use the default Consolidated Message script or customize the script. Set the **Delay** and **Frequency** settings.

Customizing SMS Message Scripts in the Pipeline Notification Block

Another way to customize the default SMS alarm notification is to build your custom message in the Pipeline Notification Block.

1. Click on the **Notification Block** of your pipeline, and choose **SMS**. You can see all the default S MS settings in the Notification Block. Click on the **Message link** icon to expand the script.

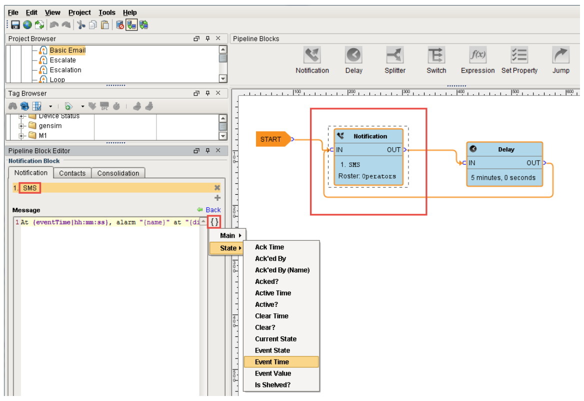


Pipeline Block Editor — Notification Block

1. SMS

Message	n}" transitioned to {eventState}
Throttled Message	{alarmEvents.Count} alarm eve
Test Mode	No

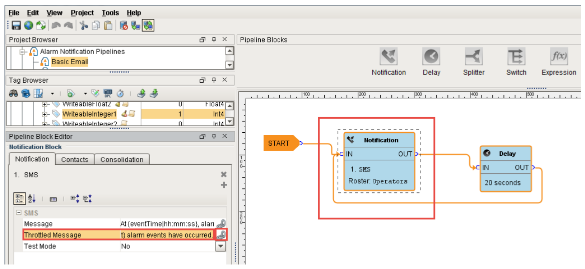
2. You can either use the default message script or customize one using the alarm properties. Click on the **Curly Braces** to see a list of available properties.



Message

```
At {eventTime|hh:mm:ss}, alarm "{name}" at "{di
```

3. Like Email, use the Consolidated Message when multiple alarm events are sent together. In **SMS** , click on the **Throttled Message link** icon to expand the consolidated script. You can either use the default script or create your own custom message using the alarm properties.



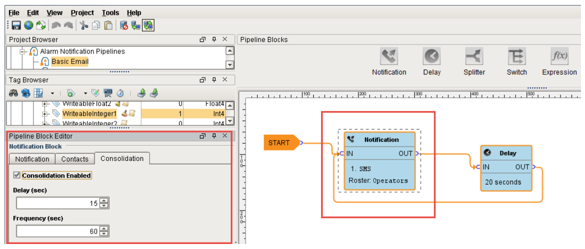
1. SMS

Roster: Operators

Message	At {eventTime|hh:mm:ss}, alarr
Throttled Message	} alarm events have occurred.
Test Mode	No

4. From the **Contacts** tab, choose the **On-Call Roster** that you want to send alarm notifications too.

5. If required, go to the Consolidation tab, and **enable Consolidation**. Enter your **Delay** and **Frequency** settings.



☑ Consolidation Enabled

Delay (sec): 15

Frequency (sec): 60

Voice

The Voice Notification Module lets you deliver alarm notifications to users via telephone.

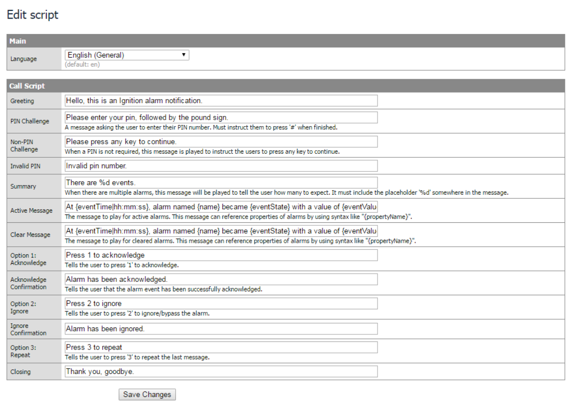
SIP and VOIP

Voice calls are delivered through any SIP compatible phone system. No dedicated hardware is required. Using a hosted service like Skype is the quickest way to implement the Voice Notification Module. For more information on how to configure any SIP compatible phone system, refer to the Voice Notification Profile.

Voice Call Script

The voice message played to the the user during the phone call is defined in the call script which is located in the Voice Notification Profile. The voice message is built off a message template that can reference properties in the alarm. Be aware that the voice message is shared across all voice notification profiles. If you define a custom message, it will be used instead of the default message in the call script.

The default call message template is shown below.


Edit script

Main	
Language	English (General) (default: en)

Call Script	
Greeting	Hello, this is an Ignition alarm notification.
PIN Challenge	Please enter your pin, followed by the pound sign. A message asking the user to enter their PIN number. Must instruct them to press '#' when finished.
Non-PIN Challenge	Please press any key to continue. When a PIN is not required, this message is played to instruct the users to press any key to continue.
Invalid PIN	Invalid pin number.
Summary	There are %d events. When there are multiple alarms, this message will be played to tell the user how many to expect. It must include the placeholder '%d' somewhere in the message.
Active Message	At {eventTime|hh:mm:ss}, alarm named {name} became {eventState} with a value of {eventValu The message to play for active alarms. This message can reference properties of alarms by using syntax like "{propertyName}".
Clear Message	At {eventTime|hh:mm:ss}, alarm named {name} became {eventState} with a value of {eventValu The message to play for cleared alarms. This message can reference properties of alarms by using syntax like "{propertyName}".
Option 1: Acknowledge	Press 1 to acknowledge Tells the user to press '1' to acknowledge.
Acknowledge Confirmation	Alarm has been acknowledged. Tells the user that the alarm event has been successfully acknowledged.
Option 2: Ignore	Press 2 to ignore Tells the user to press '2' to ignore/bypass the alarm.
Ignore Confirmation	Alarm has been ignored.
Option 3: Repeat	Press 3 to repeat Tells the user to press '3' to repeat the last message.
Closing	Thank you, goodbye.

Save Changes

Configuring the Voice Call Script

The Voice Notification system is primarily accessed through the Notification Block. The only settings you can configure in the Alarm Configuration section is the Custom Message, and the Alarm Pipeline which defines what state you want to trigger the alarm. All Voice settings that you need to configure are in the Notification Block.

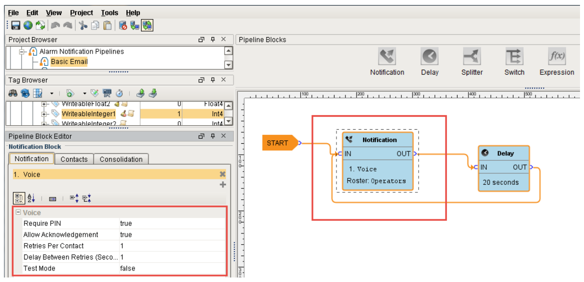
1. Click on the **Notification Block** of your pipeline, and choose **Voice**.

2. Configure each setting for your Voice alarm notification, as required.

    - **PIN Required** - If required, the user will need a PIN to hear the alarm notification. PINs are defined in the user's profile.
    - **Allow Acknowledgement** - If Acknowledgement is not required, users will only be allowed to listen to an alarm and not acknowledge them.
    - **Retries per Contact** - Number attempts to contact user by phone.
    - **Delay between Calls** - Creates a delay between each contact for each alarm. For most phone systems, a delay of 1 second is recommended.
    - **Test Mode** - If enabled, messages will be logged to the console indicating when and to whom calls would have been made without actually making a call.

3. This example shows how you might configure your Voice settings, but it depends primarily on your business requirements.

    - PIN Required - **True**
    - Allow Acknowledgement - **True**
    - Retries per contact - **1**
    - Delay between Calls - **1**
    - Test Mode - **false**



Voice	
Require PIN	true
Allow Acknowledgement	true
Retries Per Contact	1
Delay Between Retries (Seco...	1
Test Mode	false

4. From the **Contacts** tab, specify your **On-Call Roster.** Users in the Voice On-Call Roster must have a telephone number, and an established PIN in their user profile.
5. Voice notification supports message consolidation. If required, enable **Consolidation** and enter **Delay** and **Frequency** settings.

Next...

- User Pin for Voice

User Pin for Voice

You can force an operator to enter a personal identification number (PIN) in order to hear a Voice alarm notification message. It’s a good idea to validate that the person who is receiving the Voice alarm is the person they say they are by entering their PIN.

When alarms are sent out via Voice, you typically want an operator to acknowledge the alarm by pressing **‘1’** on their phone. Requiring a PIN also prevents other people who might be sharing a phone from acknowledging important alarms.

It is easy to setup Security PINs in Ignition in a couple simple steps. First, a PIN number will need to be created in each user’s profile to listen to Voice messages and acknowledge SMS messages. The second step is to configure the Voice Notification Profiles.

INDUCTIVE UNIVERSITY

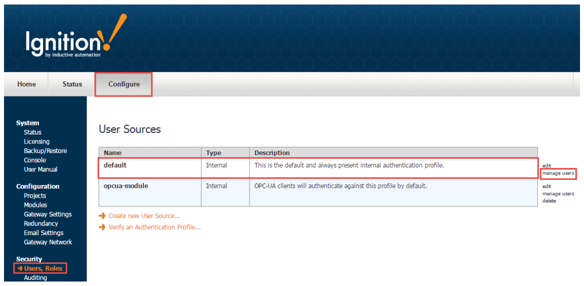
User Pin for Voice

Watch the Video

User Profile

Let’s begin by configuring a Security PIN for each user that will be receiving Voice alarm messages.

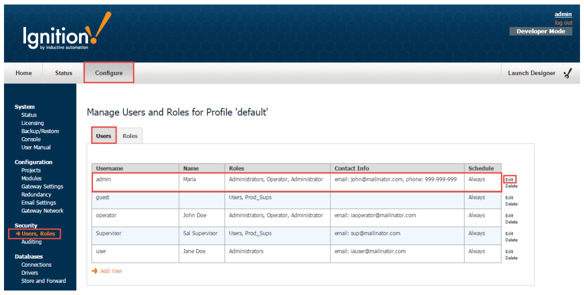
1. Go to the **Gateway Configuration** page, **Security > Users, Roles**, and under **default user sources**, select the **manage users** link.



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.

2. Here you will see list of all your users. For every user that is using Voice alarm notifications, you will need to edit their user profile to create a PIN. Select **edit** link for one of your users.



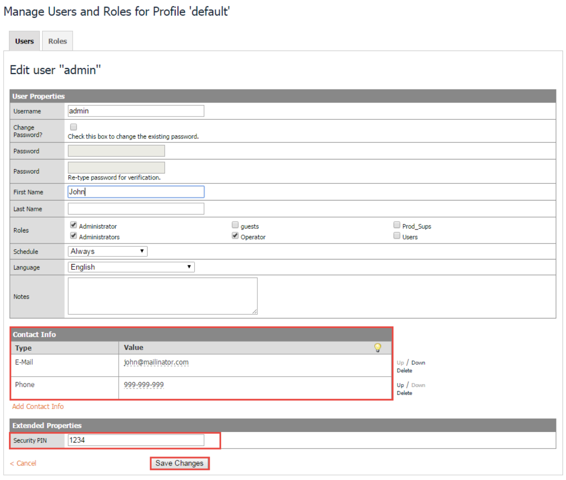
Manage Users and Roles for Profile 'default'

Username	Name	Roles	Contact Info	Schedule
admin	Maria	Administrators, Operator, Administrator	email: john@mailinator.com, phone: 999-999-999	Always
guest		Users, Prod_Sups		Always
operator	John Doe	Administrators, Operator, Administrator	email: iaoperator@mailinator.com	Always
Supervisor	Sal Supervisor	Users, Prod_Sups	email: sup@mailinator.com	Always
user	Jane Doe	Administrators	email: iauser@mailinator.com	Always

3. The user profile screen will open. Make sure the contact information for Email and Phone is up-to-date.

   Create a 4-digit **Security Pin**, and press **Save**.

   Now, this PIN is associated with this user.



Manage Users and Roles for Profile 'default'

Users | Roles

Edit user "admin"

User Properties

Username	admin
Change Password?	☐ Check this box to change the existing password.
Password	
Password	Re-type password for verification.
First Name	John
Last Name	
Roles	☑ Administrator ☑ Administrators ☐ guests ☑ Operator ☐ Prod_Sups ☐ Users
Schedule	Always
Language	English
Notes	

Contact Info

Type	Value
E-Mail	john@mailinator.com
Phone	999-999-999

Add Contact Info

Extended Properties

Security PIN	1234

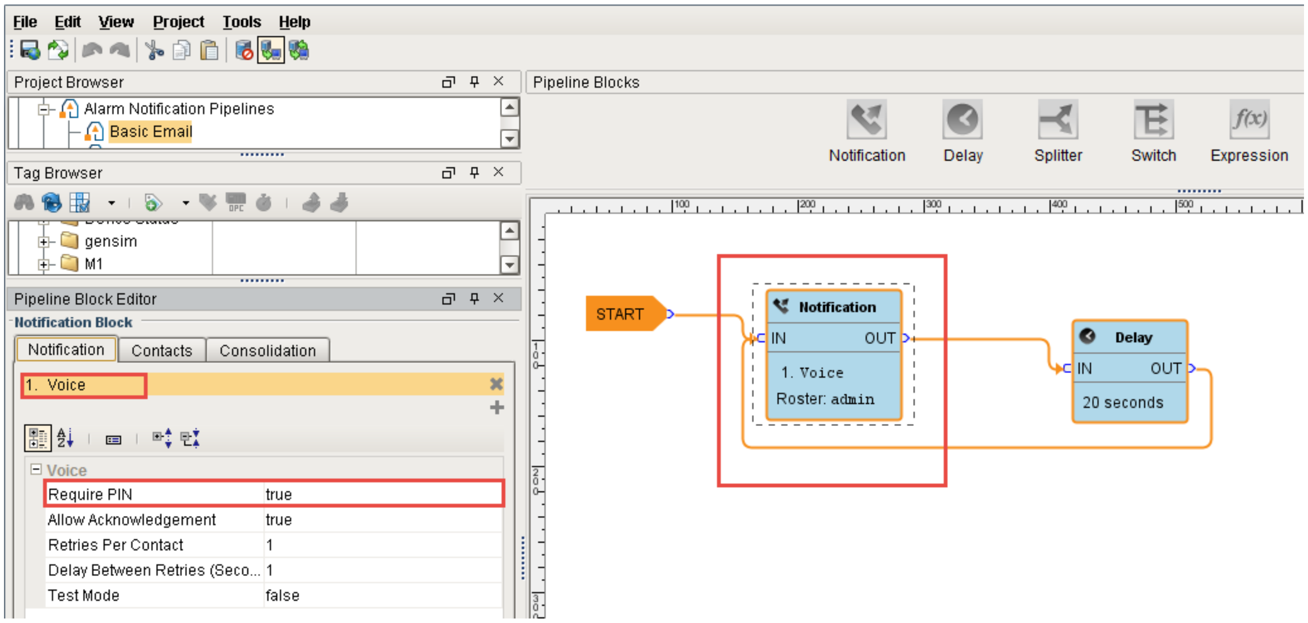
< Cancel   Save Changes

Voice Notification Profile

The next step is to configure the Voice Notification Profile to require a PIN.

1. In **Designer**, go to your **Alarm Notification Pipelines** and open your pipeline. This example uses the **Basic Email** pipeline.   Select the **Notification Block** in the workspace.
2. Click on the **Notification** tab, and choose **Voice**.
3. You'll see several settings for Voice. Set the **Required PIN** to **'true.'**

Now, when the operator receives a call, they will have to type in their PIN number followed by the pound sign to hear the message.



Require PIN	true
Allow Acknowledgement	true
Retries Per Contact	1
Delay Between Retries (Seco...	1
Test Mode	false

Next...

- Voice Languages

Voice Languages

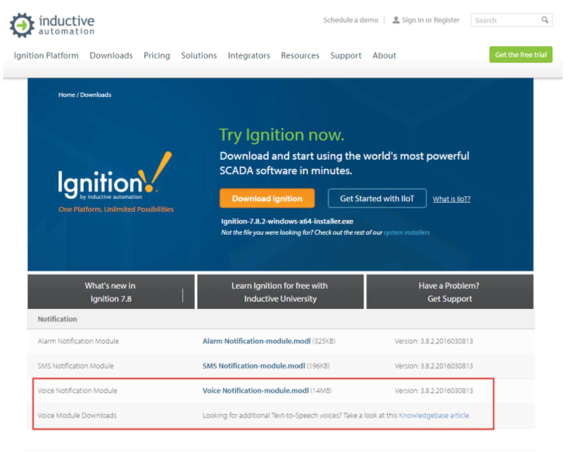
In order to let Ignition make calls when alarm events happen there are several modules, the Voice Notification Module, and many voice language modules. The voice notification module for Ignition supports making calls using male and female voices in multiple languages. The language modules (i.e., English, Spanish, Italian, etc.) contain the text to speech engine that is specific for that language and gender.

INDUCTIVE UNIVERSIT

Vocie Languages

Watch the Video

You can download these modules from Inductive Automation's Downloads page on the website, and install them in your gateway.



inductive automation
Schedule a demo
Sign In or Register
Search
Ignition Platform
Downloads
Pricing
Solutions
Integrators
Resources
Support
About
Get the free trial
Home / Downloads
Ignition by inductive automation
One Platform, Unlimited Possibilities
Try Ignition now.
Download and start using the world's most powerful SCADA software in minutes.
Download Ignition
Get Started with IIoT
What is IIoT?
Ignition-7.8.2-windows-x64-installer.exe
Not the file you were looking for? Check out the rest of our system installers
What's new in Ignition 7.8
Learn Ignition for free with Inductive University
Have a Problem? Get Support

Notification		
Alarm Notification Module	**Alarm Notification-module.modl** (325KB)	Version: 3.8.2.2016030813
SMS Notification Module	**SMS Notification-module.modl** (196KB)	Version: 3.8.2.2016030813
Voice Notification Module	**Voice Notification-module.modl** (14MB)	Version: 3.8.2.2016030813
Voice Module Downloads	Looking for additional Text-to-Speech voices? Take a look at this Knowledgebase article.	

The following table shows a list of the current Text-to-Speech voices that you can download after you have the Voice Notification Module installed. To download Text-to-Speech voices, go to the Knowledge Base article on Voice Notification Downloads where you can also find some good information.

Voices	Languages
Katherine	English
Laura	Italian
Sara	Spanish
Suzanne	French
Alex	German

Once you have installed the voice modules, you can configure the voice scripts for any language in the voice notification module. For example, you may want to have a different script that gets sent out when you're calling someone in English verses someone that you're calling in Spanish.

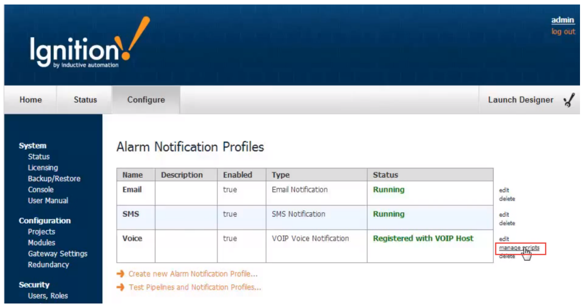
Important!

On this page we mention "**scripts**" a lot. These scripts have **nothing** to do with the scripting language (Python) that you write code with. When we say "script" on this page, we are talking about a speech or a dialog for Ignition to read to your users over the phone.

Modifying a Voice Notification Script

From Voice Notification Profiles page, you can manage your voice scripts. Two examples are provided here; one in English, and the other in Spanish.

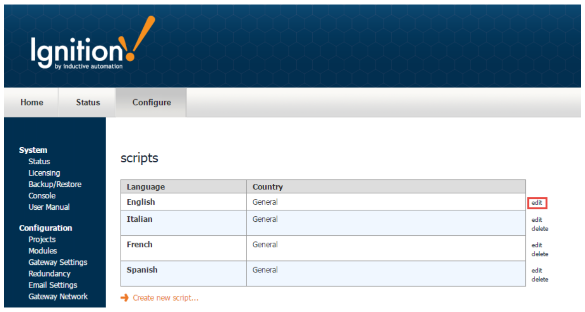
1. Go to your **Gateway Configuration** page, and select **Alarm Notifications**. To the right of the Voice notification profile, click on the **manage scripts** link.


Alarm Notification Profiles

Name	Description	Enabled	Type	Status	
Email		true	Email Notification	Running	edit delete
SMS		true	SMS Notification	Running	edit delete
Voice		true	VOIP Voice Notification	Registered with VOIP Host	edit manage scripts delete

Create new Alarm Notification Profile...
Test Pipelines and Notification Profiles...

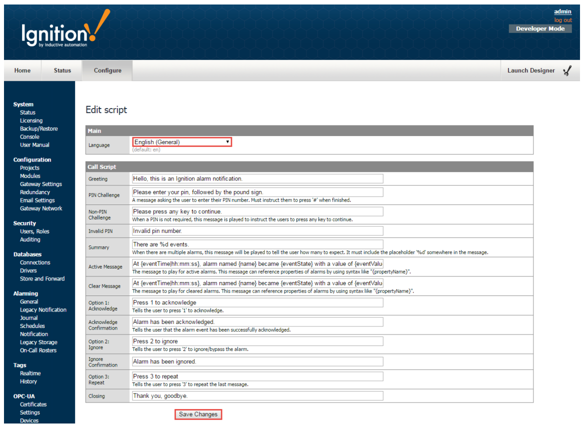
2. Here you can manage all the scripts for every language, and can even create new scripts. The languages that come with Ignition are English, Italian, French and Spanish. There are a host of other languages that Ignition supports, as you will soon see. For this example, click on the **Edit** link for **English**.


scripts

Language	Country	
English	General	edit
Italian	General	edit delete
French	General	edit delete
Spanish	General	edit delete

Create new script...

3. Default voice scripts are the same in all supported languages. This is the default script for English. It consists of a number of little scripts. Each script is what the recipient of the call will hear over the phone. These are instructions informing the recipient of one or more the alarm(s) depending on how the alarm notification is configured. When you receive a voice notification, you will hear each script or instruction one at a time. For example, the first script you will hear is “Hello this is an Ignition alarm notification” which, of course, is the greeting. It will then go through each script and inform the recipient of the alarm name, date and time of the alarm, alarm status, and how to acknowledge the alarm. You can edit this default script to your requirements. When you're finished editing, click **Save**.

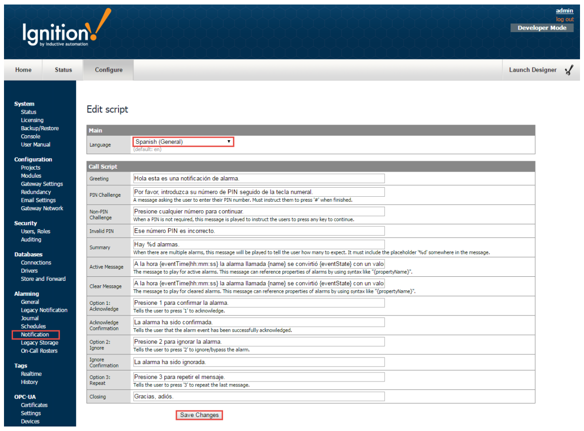

Edit script

Main	
Language	English (General) (default: en)

Call Script	
Greeting	Hello, this is an Ignition alarm notification.
PIN Challenge	Please enter your pin, followed by the pound sign. A message asking the user to enter their PIN number. Must instruct them to press '#' when finished.
Non-PIN Challenge	Please press any key to continue. When a PIN is not required, this message is played to instruct the users to press any key to continue.
Invalid PIN	Invalid pin number.
Summary	There are %d events. When there are multiple alarms, this message will be played to tell the user how many to expect. It must include the placeholder '%d' somewhere in the message.
Active Message	At {eventTime|hh:mm:ss}, alarm named {name} became {eventState} with a value of {eventValu The message to play for active alarms. This message can reference properties of alarms by using syntax like "{propertyName}".
Clear Message	At {eventTime|hh:mm:ss}, alarm named {name} became {eventState} with a value of {eventValu The message to play for cleared alarms. This message can reference properties of alarms by using syntax like "{propertyName}".
Option 1: Acknowledge	Press 1 to acknowledge Tells the user to press '1' to acknowledge.
Acknowledge Confirmation	Alarm has been acknowledged. Tells the user that the alarm event has been successfully acknowledged.
Option 2: Ignore	Press 2 to ignore Tells the user to press '2' to ignore/bypass the alarm.
Ignore Confirmation	Alarm has been ignored.
Option 3: Repeat	Press 3 to repeat Tells the user to press '3' to repeat the last message.
Closing	Thank you, goodbye.

Save Changes

4. Here is an example of the default scripts in Spanish. Once again, you can edit the scripts to your requirements. When editing is complete, click **Save**.



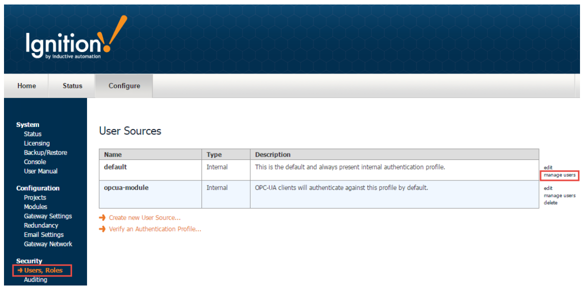
Edit script

Main	
Language	Spanish (General) (default: en)

Call Script	
Greeting	Hola esta es una notificación de alarma.
PIN Challenge	Por favor, introduzca su número de PIN seguido de la tecla numeral. A message asking the user to enter their PIN number. Must instruct them to press '#' when finished.
Non-PIN Challenge	Presione cualquier número para continuar. When a PIN is not required, this message is played to instruct the users to press any key to continue.
Invalid PIN	Ese número PIN es incorrecto.
Summary	Hay %d alarmas. When there are multiple alarms, this message will be played to tell the user how many to expect. It must include the placeholder '%d' somewhere in the message.
Active Message	A la hora {eventTime|hh:mm:ss} la alarma llamada {name} se convirtió {eventState} con un valo The message to play for active alarms. This message can reference properties of alarms by using syntax like "{propertyName}".
Clear Message	A la hora {eventTime|hh:mm:ss} la alarma llamada {name} se convirtió {eventState} con un valo The message to play for cleared alarms. This message can reference properties of alarms by using syntax like "{propertyName}".
Option 1: Acknowledge	Presione 1 para confirmar la alarma. Tells the user to press '1' to acknowledge.
Acknowledge Confirmation	La alarma ha sido confirmada. Tells the user that the alarm event has been successfully acknowledged.
Option 2: Ignore	Presione 2 para ignorar la alarma. Tells the user to press '2' to ignore/bypass the alarm.
Ignore Confirmation	La alarma ha sido ignorada.
Option 3: Repeat	Presione 3 para repetir el mensaje. Tells the user to press '3' to repeat the last message.
Closing	Gracias, adiós.

As you can see, you can attach a different script for every language depending on the user, and whatever that user is setup to do. So, if the script is in Spanish, the user will receive the voice notification in Spanish assuming the Spanish voice module is installed in Ignition.

5. Once you have the appropriate voice module and language module(s) installed, go to **Security > Users and Roles**. Select the **manage users** link for the default user source.

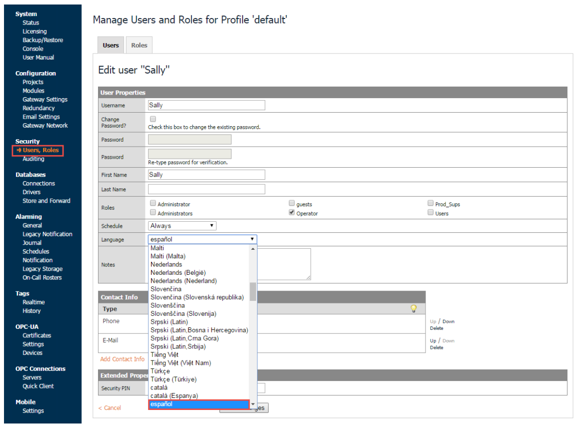

Home
Status
Configure
Users, Roles

User Sources

Name	Type	Description	
default	Internal	This is the default and always present internal authentication profile.	edit manage users
opcua-module	Internal	OPC-UA clients will authenticate against this profile by default.	edit manage users delete

Create new User Source...
Verify an Authentication Profile...

6. You can edit every single user, and put each user into a different language, as long as the language that you select for each user has the voice module for that language installed. In this example, let's edit a user, Sally. Put Sally into the Spanish language. If the voice module is installed for the Spanish language, Ignition will automatically choose the Spanish script.



Manage Users and Roles for Profile 'default'

Edit user "Sally"

User Properties	
Username	Sally
Change Password?	☐ Check this box to change the existing password.
Password	
Password	Re-type password for verification.
First Name	Sally
Last Name	
Roles	☐ Administrator ☐ Administrators ☐ guests ✓ Operator ☐ Prod_Sups ☐ Users
Schedule	Always
Language	español
Notes	

Now, when the alarm notification is received, the voice message will be in Spanish. You can also change a user's language preferences from the User Management component.

Next...

Transaction Groups

Transaction groups are configured through the Ignition Designer. They are units of execution that perform actions such as storing data historically, synchronizing database values to OPC, and loading recipe values. While they are stored in a project, they execute on the Gateway. This ensures that only one instance of each Transaction Group is run at all times, regardless of the number of Clients being open.

In this section ...

Understanding Transaction Groups

Transaction Groups are the heart of the SQL Bridge module. They are units of execution that perform a variety of actions, such as storing data historically, synchronizing database values to OPC, or loading recipe values. A variety of group types, items types, and options means that Transaction Groups can be configured to accomplish almost any task.

The Transaction Group Workspace

Transaction groups are edited through the Ignition Designer. When a group is selected, you are presented with the transaction group workspace.

The workspace is broken into several parts:

1. Title bar - Shows the name of the currently selected group, as well as options to set it as Enabled or Disable, and to Pause, if it's currently executing.
2. Item configuration - Shows all of the items configured in the selected group. Many settings can be modified directly through the display, the rest by double-clicking the item, or selecting "edit" in the context menu.
3. Action / Trigger / Options tabs - Define how and when a group executes. Holds most of the options that apply to the group in general, such as the update rate, and which data connection it uses.
4. Status / Events tabs - Provides information about the executing group, including the most recent messages that have been generated.

Enabling Group Execution

In order for groups to be evaluated, they must first be enabled. This is done by selecting "enabled" in the group title bar, and then saving the project. The group executing can be stopped by reversing the procedure and selecting "disabled" before saving. If you want to quickly and temporarily stop the group's evaluation, toggle the "pause" button. This will prevent execution until the group is unpaused, or until the system is restarted.

Editing Group Settings

Group settings may be modified at any time, regardless of whether or not the group is executing. Modifications will be applied when the project is saved, and the group will be started or stopped as required. Some changes such as modifying items may cause features like live values to appear to be incorrect. It is therefore important to note the modified icon that appears next to the group, and to save often. If you would prefer to stop the group before making edits you can simply pause the group. Execution will begin again after the project is saved.

How Groups Execute

Generally speaking, groups work on a timer. They are set to run at a certain rate, and at that rate, the check the rest of the settings. If the trigger conditions pass, the group is executed fully. The following section provides a fuller outline of the execution cycle.

In this section ...

About Transaction Groups

Transaction Groups allow you to perform tasks such as: database to OPC synchronization, recipe management, and historical data logging.

INDUCTIVE UNIVERSITY

About Transaction Groups

Watch the Video

Execution Cycle

All of the groups follow a similar execution cycle. The core evaluation may differ, but the general cycle is the same.

1. Timer executes, group enters execution
2. Is the group paused? Break execution.
3. Is the Gateway part of a redundant pair? If so, is it active? If not active, break execution. Groups only execute on the active node.
4. Evaluate "run-always" items: OPC items, tag references, and Expression items set to ignore the trigger (or placed in the "run always" section of the configuration window).
5. Is trigger set/active? If there is a trigger defined, but it is not active, break execution.
6. Evaluate "triggered" expression items.
7. If applicable, read values from the database.
8. Execute a comparison between items and their targets.
9. Execute any writes to other Tags or the Database that result from execution.
10. Report alerts.
11. Acknowledge the trigger, if applicable.
12. Write handshake value, if applicable.

If an error occurs at any stage besides the last stage, execution will break and the failure handshake will be written if configured. The group will attempt execution again after the next update rate period.

Anatomy of a Group

Action Settings

The **action settings** of a group define how often the group will be evaluated, as well as important settings that apply to the group as a whole.

They are found on the **Action** tab, the first of the tabs on the right side of the Transaction Group workspace.

Common Settings

The settings vary for the different types of groups, but a few setting are common to most of them:

Execution scheduling	How often the group is evaluated. For a number of reasons, the group may not execute during the evaluation. The most common reason is the trigger, but see Execution Cycle above for more possible reasons why evaluation will exit.
Data source	The data connection to use for the group. Can be **Default**, which will use the default connection for the project.
Update mode	For groups that support it, sets the default for how items are compared to their targets.
Store timestamp	Stores a timestamp along with the data any time the group executes.
Store quality code	Stores an aggregate quality for the group along with the regular data. The aggregate quality is a bit-wise AND of the qualities of the items in the group.

Execution Scheduling

There are two ways to specify when the group should execute: timer and schedule mode.

Timer Mode

In this mode, the group is evaluated regularly at the provided rate. As mentioned in the previous sections, due to trigger settings, full execution may not occur, but the trigger will at least be evaluated at this rate.

Schedule Mode

With schedule mode, you are providing a list of time (or time ranges) that the group should run at. If the pattern specified includes a time range, at rate must be provided, and the group will execute as in timer mode during that period.

The schedule pattern

The schedule is specified as a comma separated list of times or time ranges. You may use the following formats:

24-hour times. Ie. "8:00, 15:00, 21:00", for execution at 8am, 3pm, and 9pm.

12-hour with am/pm (if not specified, "12" is considered noon): "8am, 3pm, 9pm"

Ranges, "8am-11am, 3pm-5pm"

Notes

It is allowed for time ranges to span over midnight, such as "9pm - 8am"

When using ranges, the execution times will be aligned to the start time. For example, if you specify a schedule of "9am - 5pm" with a rate of "30 minutes", the group will execute at 9, 9:30, 10, etc., regardless of when it was started. This is a useful difference compared to the Timer Mode, which runs based on when the group was started. For example, if you wanted a group that ran every hour, on the hour, you could specify a 1 hour rate with a range of "0-24".

Trigger and Handshake Settings

The trigger settings determine when a group will actually execute. They are examined each time the group evaluates (according to the update rate of the group). If they pass, the group will run and perform its action against the database.

The trigger settings are the same for all group types. They are found on the second tab (labeled "Trigger"), in the right side of the Transaction Group workspace.

Only execute when value have changed (asynchronous trigger)

These settings are evaluated first. If set, the group will examine whether the values in the specified tags have changed, and if not, will exit evaluation.

It is possible to monitor all Run-Always tags in the group, or only specific ones.

Execute this group on a trigger

Enables trigger on a specific item in the group. The trigger item can be any Run-Always item, such as an OPC item, Tag reference, or an Expression item set to "Run-Always" mode.

In addition to the numeric settings that define the trigger, there are several other options:

- **Only execute once while trigger is active** - The group will only execute once when the trigger goes into an active state, and will not execute again until the trigger goes inactive first. If unselected, the group will execute each time the trigger conditions evaluate to true.
- **Reset trigger after execution** - If using the ">0" or "=0" trigger modes, the trigger can be set to write an opposite value after the group has executed successfully. This is useful for relaying the execution back to the PLC.
- **Prevent trigger caused by group start** - If selected, the group will not execute if the trigger is active on the first evaluation of the group. In the course of designing a group, it is common to stop and start it many times, and sometimes it is not desirable to have the group execute as a result of this. Selecting this option will prevent these executions, as well as executions caused by system restarts.

Handshake Settings

Group handshakes are also defined on the trigger tab. It is possible to specify both a success and failure handshake. The success handshake will write the specified value to the given item when the group has finished all other triggered execution without error.

The failure handshake, on the other hand, will be written when the group execution is cut short due to an error, such as an error writing to the database or an item.

Advanced Settings

Transaction groups offer several advanced settings that affect how execution occurs. These settings can be found under the **Options** tab for a group.

OPC Data Mode

This setting modifies how the group receives data from OPC.

Subscribe - Data points are registered with the OPC server, and data is received by the group **on-change**. This is the default setting and generally offers the best performance, as it reduces unnecessary data flow and allows the OPC server to optimize reads.

Note: Data is received by the group asynchronously, meaning that it can arrive at any time. When the group executes, it "snapshots" the last values received and uses those during evaluation. If some values arrive after execution begins, they will not be used until the following execution cycle.

Read - Each time the group executes it will first read the values of OPC items from the server. This operation takes more time and involves more overhead than subscribed evaluation, but ensures that all values are updated together with the latest values. It is therefore commonly used with batching situations, where all of the data depends on each other and must be updated together. It's worth noting that when using an OPC item as the trigger, the item will be subscribed, and the rest of the values read when the trigger condition occurs. Note: This option was previously referred to as "polled reads" in earlier versions of the software.

Bypass Store and Forward System

Only applicable to groups that insert rows into the database. Causes groups to target the database directly instead of going through the store-and-forward system. If the connection becomes unavailable, the group will report errors instead of logging data to the cache.

Override OPC Subscription Rate

Specifies the rate at which OPC items in the group will be subscribed. These items are normally subscribed at the rate of the group, but by modifying this setting it is possible to request updates at a faster or slower rate.

Creating a Transaction Group

Transaction groups are used to store history, log events, synchronize databases tables with PLC, perform calculations, and many more data-centric tasks.

To create a transaction group

Let's create a basic History Group and start logging some PLC values to your database.

1. Click on the **Transaction Groups** in the **Project Browser** panel to switch the Designer's workspace to the Transaction Group workspace.

2. Click on **File > New > New Standard Group** to make a new Historical Group.
   Your new group, named **Group** will be selected. The **OPC Browser** panel is now docked to the lower-left of the screen.

3. Browse your OPC device and drag some OPC tags to the **Basic OPC/Group Items** section.
   Your group starts out disabled by default.

4. Click the **Enabled** button above the item tables to enable logging.

5. Go to the **Action** tab and change the **Table Name** in your table to something interesting.
   Right now your group only exists in the Designer.

6. **Save** the project to start the group.
   Your group is now running and logging data to your database connection. To see the data, you can use the Ignition Designer's built-in database query browser. The easiest way to do this is to click on the Query Browser ( ) button to the right of your group's Table Name field. The Query Browser is a convenient way to directly query your database connection without leaving the Ignition Designer. Of course - you can also use any query browser tools that came with your database.

Next...

- Types of Groups

Types of Groups

The SQL Bridge Module provides four different types of transaction groups:

- Standard
- Block
- Historical
- Stored Procedure

See an overview of these different types of groups, and how they vary in their operation and use for data logging and database to PLC synchronization.

Types of Groups

Watch the Video

Standard Group

The standard group is called such because it's a flexible, general use group that can be adapted to a variety of situations. The data model is row based, with items mapping to columns and the data corresponding to a specific row of a table.

General Description

The standard group contains items, which may be mapped to the database, or used internally for features such as triggering or handshakes. Items that are mapped to the database target a specific column of a single specific row, chosen according to the group settings. Items can be mapped in a one-way fashion, or bi-directionally, in which the value of the database and the item will be synchronized.

The group may also insert new rows instead of updating a specific row. In this manner, data can be inserted for historical purposes based on a timer, with an optional trigger.

Group Settings

The standard group uses a timer-based execution model shared by all groups, and the normal trigger settings.

Additionally, there are several settings specific to the group type:

Automatically create table - If the target table does not exist, or does not have all of the required columns, it will be created/modified on group startup. If not selected and the table doesn't match, an error will be generated on startup.

Store timestamp - Specifies whether or not to store a timestamp with the record, and the target column. The timestamp will be generated by the group during execution. For groups that update a row, the timestamp will only be written if any of the values in the group is also written.

Store quality code - If selected, stores an aggregate quality for the group to the specified column. The aggregate quality is the combined quality of all of the items that write to the table. For more information about quality values, see Data Quality Delete records older than - If selected, records in the target table will be deleted after they reach the specified age. This setting is useful for preventing tables from growing in an unbounded manner, which can cause disk space and performance problems over time.

Table action - This section details how the group interacts with the table. The group can insert a new row each execution, or update the first, last or custom record. A custom update clause is essentially the where clause of the SQL query that will be generated to read and write the group. In addition to standard SQL syntax, you can bind to items in the group in order to inject dynamic values.

Typical Uses

Standard groups can be used any time you want to work with a single row of data. This can include:

Historical logging - set the group to insert new records, and log data historically either on a timer, or as the result of a trigger. Flexible trigger settings and handshakes make it possible to create robust transactions.

Maintain status tables - Keep a row in the database updated with the current status values. Once in the database, your process data is now available for use by any application that can access a database, dramatically opening up possibilities.

Manage recipes - Store recipe settings in the database, where you have a virtually unlimited amount of memory. Then, load them into the PLC by mapping DB-to-OPC using a custom where clause with an item binding in order to dynamically select the desired recipe.

Sync PLCs - Items in the group can be set to target other items, both for one-way and bidirectional syncing. By adding items from multiple PLCs to the group, you can set the items of one PLC to sync with the others. By creating expression items that map from one PLC item to the other, you can manipulate the value before passing it on.

Block Group

The block group is so named because it writes "blocks" of data to a database table, consisting of multiple rows and columns.

General Description

A block group contains one or more block items. Each block item maps to a column in the group's table, and then defines any number of values (OPC or SQLTag items) that will be written vertically as rows under that column. The values may be defined in the block item in two modes. The first, List mode, lets a list of value-defining items to be entered. These value items may either by OPC items, SQLTag items, or static values. The second mode, Pattern mode, can be useful when OPC item paths or tag paths contain an incrementing number. You may provide a pattern for the item's path, using the wildcard marker {?} to indicate where the number should be inserted.

Block groups are very efficient, and can be used to store massive amounts of data to the database (for example, 100 columns each with 100 rows- 10,000 data points- will often take only a few hundred milliseconds to write, depending on the database). They are also particularly useful for mirroring array values in the database, as each element will appear under a single column, and share the same data type.

Like the standard group, the block group can insert a new block, or update the first, last or a custom block. Additionally, the group can be set to only insert rows that have changed in the block.

In addition to block items, the group can have other OPC items, tag references, and Expression items. These items can be used for triggers, handshakes, etc. They may also target a column to be written, and will write their single value to all rows in the block.

Group Settings

Beyond the differences in the data, namely that the block group works with multiple rows instead of just 1, this group type shares many similarities with the Standard Group.

The unique settings are:

Store row id - Each row will be assigned a numeric id, starting at 0. If selected, this id will also be stored with the data.

Store block id - If selected, an incremental block id will be stored along with the data. This number will be 1 greater than the previous block id in the table.

Insert new block vs. Insert changed rows - If "insert new block" is selected, each row of the block will be inserted when the group executes, even if the data has not changed. By contrast, "insert changed rows" will only insert the rows that have new data. The latter mode is particularly useful for recording history for many data points on a "on change" basis, provided there is a unique id column defined. The "store row id" feature is useful for this, as well as the ability to reference the item path in an item's value property.

Update Custom block - Like standard groups, this setting allows you to target a specific section of the table, using SQL where clause syntax, with the ability to bind to dynamic item values. Unlike standard groups, however, the where clause specified should result in enough rows to cover the block. Excess rows will not be written to, but fewer rows will result in a group warning indicating that some data could not be written.

Typical Uses

Block groups are useful in a number of situation where you need to deal with a lot of data efficiently. Mirroring/Synchronizing array values to DB - Arrays are often best stored vertically, which makes them perfect for block groups. Pattern mode makes configuration a breeze by allowing to you specify the array as a pattern, and set the bounds.

Recipe management - Like standard groups, but when set points are better stored vertically than horizontally.

Vertical history tables - Group data points by data type (int, float, string), create a copy of the item that stores item path, and then use the insert changed rows option to create your own vertically storing historical tables. Create additional copies of the block item that refer to quality and timestamp in order to get further information about the data point.

Historical Group

The historical group makes it easy to quickly log data historically to a SQL database.

General Description

The historical group inserts records of data into a SQL database, mapping items to columns. Full support for triggering, expression items, hour & event meters and more means that you can also set up complex historical transactions. Unlike the standard group, the historical group cannot update rows, only insert. It also cannot write back to items (besides trigger resets and handshakes).

Group Settings

The settings of the historical group are identical to the settings in the Standard Group, but limited to inserting rows.

Typical Uses

Basic historical logging - Recording data to a SQL database gives you incredible storage and querying capabilities, and makes your process data available to any application that has DB access.

Shift tracking - Use an expression item to track the current shift based on time, and then trigger off of it to record summary values from the PLC. Use a handshake to tell the PLC to reset the values.

Stored Procedure Group

The stored procedure group lets you quickly map values bi-directionally to the parameters of a stored procedure.

General Description

The stored procedure group is similar to the other groups in terms of execution, triggering, and item configuration. The primary difference is that unlike the other group types, the target is not a database table, but instead a stored procedure.

Items in the group can be mapped to input (or inout) parameters of the procedure. They also can be bound to output parameters, in which case the value returned from the procedure will be written to the item. Items can be bound to both an input and output at the same time.

Group Settings

The stored procedure group's settings look and act the same as those of the Historical Group. The primary difference, of course, is that instead of specifying a table name and column names, you'll specify parameter names.

Parameters may be specified using either **parameter names** or **numerical index**. That is, in any location where you can specify a parameter, you can either use the name defined in the database, or a 0-indexed value specifying the parameter's place in the function call. **Important:** You cannot mix names and indices. That is, you must consistently use one or the other.

If using parameter names, the names should not include any particular identifying character (for example, "?" or "@", which are used by some databases to specify a parameter).

Typical Uses

Call stored procedures - The stored procedure group is the obvious choice when you want to bind values to a stored procedure. It can also be used to call procedures that take no parameters (though this can also be accomplished from Expression Items/SQLTags.

Replace RSSQL - The stored procedure group is very popular among users switching from RSSQL, given that application's heavy use of stored procedures.

Known Issues

When using Oracle, you must use indexed parameters.

Next...

- Group Update Rate

Group Update Rate

The execution schedule controls the rate at which the transaction group executes. On the **Action** tab of a group you have selected, under **Execution Scheduling**, there are two options: Timer and Schedule. Timer, executes the group at a certain rate. Schedule, executes the group at specific times. When the Schedule option spans across a period of time, you must specify the rate at which the group executes during that time.

Timer

The Timer acts as the heartbeat of the transaction group. It can run at millisecond, second, minute, hour, or day rates. The Timer specifies the OPC tag subscription rate for the OPC tags. When a Timer is running the transaction group first analyzes the tags inside the **Basic OPC/Group Items** section of the transaction group. Then it looks at the trigger configuration and evaluates for tag changes. Then it evaluates the specific trigger conditions and decides to execute on a trigger. If the triggered condition is true, the transaction group proceeds to the **Triggered Expressi**on Items section of the transaction group. Only after this flow is complete, will the transaction group interacts with the database and for example, insert the tag values into the database.

Schedule

An important difference between the Timer and the Schedule options is that the schedule option will automatically align to the specified start time on the update rate. For example, if the Schedule option is selected and the times are entered as **8,12pm-3pm** and the update rate is **15 minutes**, the transaction group will execute at 8am and then at 12pm, 12:15pm, 12:30pm and so on.

INDUCTIVE UNIVERSITY

Group Update Rate

Watch the Video

Next...

- Item Types

Item Types

Items are the core elements of a group. They are executed, and the values are then used by the group for logic purposes, by other items, and to write to the database. They can be written to from the database or from other items.

Type of Item	Description
OPC Item	Directly subscribed to an OPC server at the rate of the group. Executed by the group, so alarms are evaluated when the group is executed. These items are executed even when the trigger isn't active.
Run-Always Expression Item	Much like an expression tag, can be either a static value, an expression, or a database query. Run-Always expression items are evaluated at each group interval, before the trigger state is evaluated.
Triggered Expression Item	Same as Run-Always expression items, except that they are only executed after the trigger has been evaluated and is active.
Tag Reference	A reference to a Tag. Allows a Tag to be used in a group like any other item type, except that the tag is evaluated by its scan class instead of by the group. For more information, see Tag References vs. OPC Items.

INDUCTIVE UNIVERSIT

Item Types

Watch the Video

Execution Order

Items generally aren't executed in a reliable order, with the exception of **Expression** items. Expression items can be ordered using the up and down arrows located to the right of the list where the items are displayed. This can be crucial for performing complex operations that require a specific sequence.

Next...

- Tag References vs. OPC Items

Tag References vs. OPC Items

It is easy to confuse the definition and purpose of Tag reference items (Scan Class) and direct OPC items (Timer) in transaction groups, though they have distinct benefits.

Tags may be referenced inside of groups, however it is critical to remember that they are executed by the Ignition Gateway, according to their scan classes, and independent of the group. Adding a tag into a group is like creating a shortcut to that tag. However, once in the group, the item can be used like any other item. That is, it may be mapped bi-directionally to the database, used as a trigger, be the target of another item, and so on. It is even possible to create an hour meter out of the item. Core properties of the tag such as alarming and scaling, however, are defined in the actual tag, not in the group.

OPC Items in groups (as well as expression items in groups), however, are completely executed by the group. They do not exist outside of the group in which they are defined. They are subscribed and evaluated according to the rate of the group. Generally speaking, it is most common to create OPC items in groups, even if a particular point might already exist in tags. This leads to more understandable group execution, as all evaluation occurs in the group according to the timer and trigger settings. Tag references are useful when it is necessary to have a single value in multiple groups, for example, as a trigger in order to coordinate execution.

INDUCTIVE UNIVERSITY

Tag References vs. OPC Items

Watch the Video

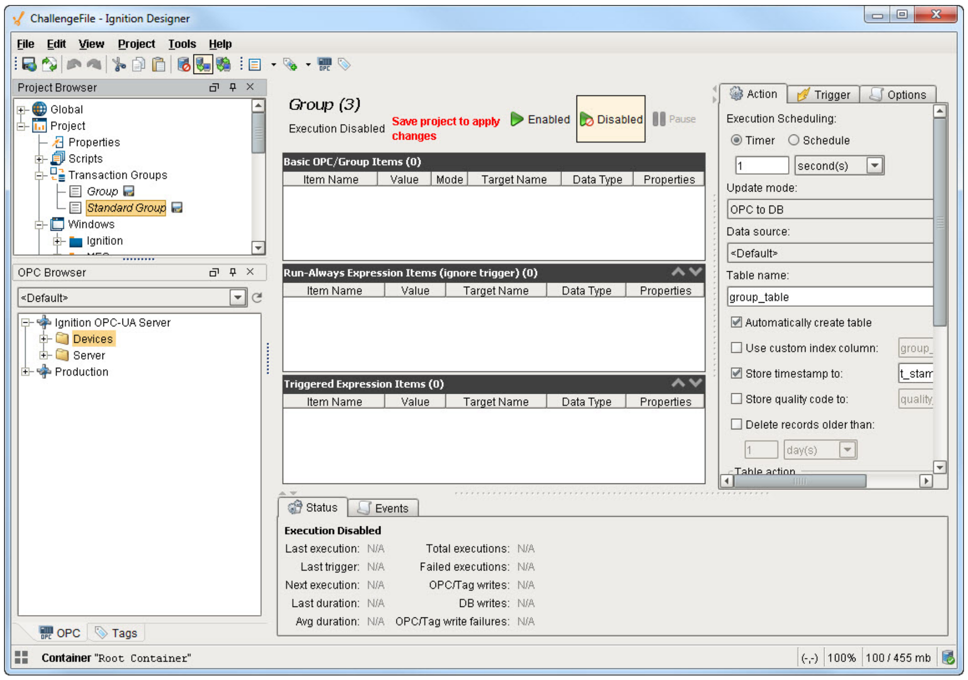
OPC Item

OPC Items are the backbone of a group. They get their values from PLCs and the values are then used by other items in the group and/or to write to the database. They are directly subscribed to an OPC server at the rate of the group and are executed by the group so their alarms are evaluated when the group is executed. These items are executed even when the trigger isn't active.

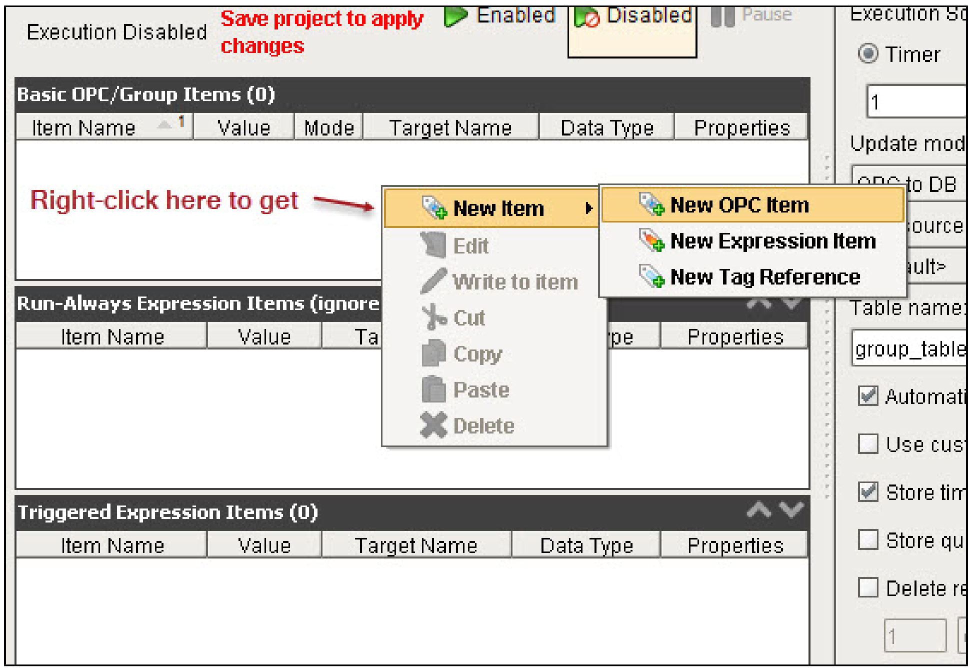
To create a new OPC Item

1. In the Designer, go to Project Browser, and click on **Transaction Groups**.
   The workspace now changes to the Transaction Group workspace.

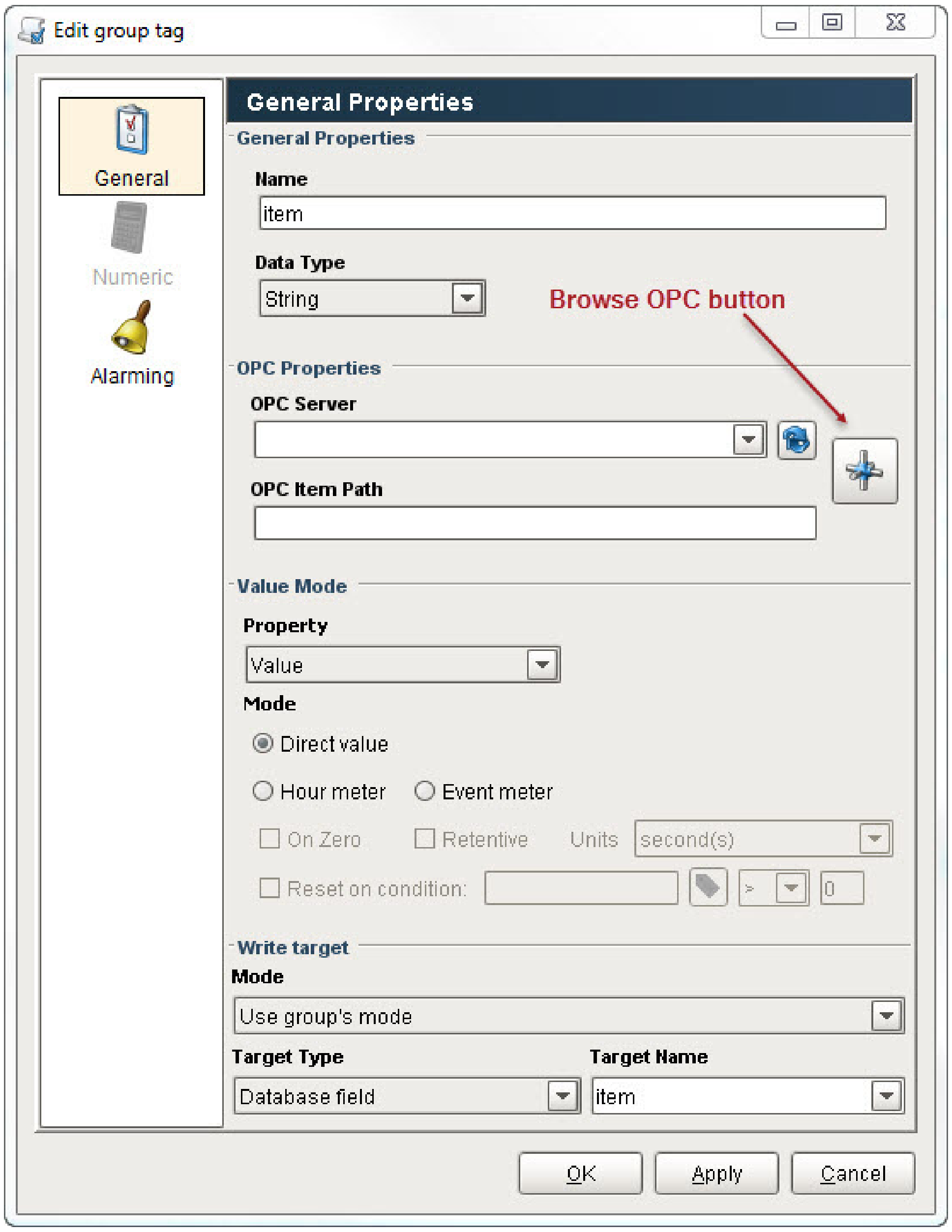
2. Right-click on **Transaction Group** to create a New Transaction Group, or click on a group you have previously created.
   You will now see the workspace changes to look like:



3. Right-click in the white area, and choose **New Item > New OPC Item**.



The **Edit group tag** window is displayed as follows:



General Properties

Name: item

Data Type: String

OPC Properties

OPC Server:

OPC Item Path:

Value Mode

Property: Value

Mode: (● Direct value) ○ Hour meter ○ Event meter

☐ On Zero ☐ Retentive Units second(s)

☐ Reset on condition: > 0

Write target

Mode: Use group's mode

Target Type: Database field

Target Name: item

The following table describes all the properties on the **Edit group tag** window.

OPC Item General Properties	
General Properties	
Name	The name of the OPC item in the group. There cannot be duplicate names within a group.
Data Type	The datatype used to read values from the PLC.
OPC Properties **Click on the Browse OPC button, in this section, select the tag you want, and Ignition fills out the following fields for you.**	
OPC Server	The Selected OPC Server. This is a drop-down list showing all the OPC Servers added in the Ignition Gateway.
OPC Item Path	The OPC Item's address Including Device name or Channel if required.

Value Mode	
Property	Which property of the OPC item you want to use. • Value - Item value • Quality - Quality code from OPC Server (192 = GOOD_DATA) • Timestamp - The last time the item value changed • Name - The SQLBridge Item Name property of this Item
Mode	Options for displaying values based on the Item value. • Direct Value - Item value • Hour Meter - Record the amount of time the Item value is non-zero. This accumulation will reset to zero when the item value goes to zero. The datatype should be set to integer or float when using an Hour Meter regardless of the OPC Item type. ○ On Zero - Use a zero value to accumulate time instead of a non-zero value ○ Retentive - Retain the Hour Meter value when it is not accumulating. ○ Units - The time units to display. • Event Meter - Record the number or times the Item value is non-zero. The datatype should be set to integer when using an Event Meter regardless of the OPC Item type. ○ On Zero - Use a zero value to accumulate events instead of a non-zero value
Write Target	
Mode	Changes the items directional read/write option. • Use group's mode - Inherit the Update Mode from the Item's Group. • OPC to DB - Only read from the OPC server and write to the database. • DB to OPC - Only read from the database and write to the OPC Server. • Bi-directional OPC wins - Read and Write to both the database and OPC Server. On group start, write OPC Server values to the database. • Bi-directional DB wins - Read and Write to both the database and OPC Server. On group start, write database values to the database.
Target Type	This is the selection for what the Item will write to when the group executes. • None, read-only item - Do not write this value to the database. • Database field - Write the Item value to the specified column in the database table. This list will populate with all the column names from the Group's target table after the first time the group is run.
Target Name	The name of the column in the database that this Item will write to when the group executes. The Target Name list will populate with all the column names from the Group's target table if the Target Type is Database field.
Alarming **The Alarming settings for the OPC items. See Alarming Properties for a full explanation.**	

Tag Reference

Tag References are used just like OPC Items, adding the convenience of using a Tag that has already been set up with scaling and alarm data.

Tag Item	
General	
Name	The name of the OPC item in the group. There cannot be duplicate names within a group.
Tag Path	The path to the tag being referenced. This value is not editable except by clicking the Insert Tag button. There cannot be duplicate names within a group.
Data Type	The datatype to write to into the database if this item is not read-only.
Value Mode	
Property	Which property of the Tag you want to use. • Value - Item value • Quality - Quality code of the Tag (192 = GOOD_DATA) • Timestamp - The last time the item value changed

	• Name - The SQLBridge Item Name property of this Item.
Mode	Options for displaying values based on the Item value. • Direct Value - Item value • Hour Meter - Record the amount of time the Item value is non-zero. This accumulation will reset to zero when the item value goes to zero. The datatype should be set to integer or float when using an Hour Meter regardless of the OPC Item type. On Zero - Use a zero value to accumulate time instead of a non-zero value Retentive - Retain the Hour Meter value when it is not accumulating. Units - The time units to display. • Event Meter - Record the number or times the Item value is non-zero. The datatype should be set to integer when using an Event Meter regardless of the OPC Item type. On Zero - Use a zero value to accumulate events instead of a non-zero value
Write Target	
Mode	Changes the items directional read/write option. This is only editable when the target Type is set to Database field. • Use group's mode - Inherit the Update Mode from the Item's Group. • OPC to DB - Only read from the OPC server and write to the database. • DB to OPC - Only read from the database and write to the OPC Server. • Bi-directional OPC wins - Read and Write to both the database and OPC Server. On group start, write OPC Server values to the database. • Bi-directional DB wins - Read and Write to both the database and OPC Server. On group start, write database values to the database.
Target Type	This is the selection for what the Item will write to when the group executes. • None, read-only item - Do not write this value to the database. • Database field - Write the Item value to the specified column in the database table.
Target Name	The name of the column in the database that this Item will write to when the group executes. The Target Name list will populate with all the column names from the Group's target table if the Target Type is Database field.

Next...

- Expression Items

Expression Items

Expression Items are used for executing comparisons, simple math and querying additional database tables. They get their values from an expression made up of static values or other items, or from SQL Queries. They can have alarms and can be executed when the trigger is active or every time the group executes.

Expression items are items in groups that handle an expression much like an expression tag. The main difference between the two is that expression items in transaction groups only exist within the transaction group and they only update at the rate of the transaction group. You can use expression items to perform calculations and create complex triggers.

Expression Items

Watch the Video

To create an Expression Item

1. From the **Transaction Gourps** node of the **Project Browser**, select a group.
2. In the white area of **Basic OPC/Group Items**, **Run-Always** or **Triggered Expression Items**, right-click and select **New Item > New Expression Item**.
   The **Edit group tag** window is displayed and you can set the following properties:

Expression Item General Properties	
General Properties	
Name	The name of the OPC item in the group. There cannot be duplicate names within a group.
Value	The static value of this Expression item. This will be overwritten by an Expression/SQL binding.
Datatype	The datatype values are stored as.
Value Mode	
Property	Which property of the OPC item you want to use. • Value - Item value • Quality - Quality code of the expression/SQL Query (192 = GOOD_DATA) • Timestamp - The last time the item value changed • Name - The SQLBridge Item Name property of this Item.
Mode	Options for displaying values based on the Item value. • Direct Value - Item value • Hour Meter - Record the amount of time the Item value is non-zero. This accumulation will reset to zero when the item value goes to zero. The datatype should be set to integer or float when using an Hour Meter regardless of the OPC Item type. On Zero - Use a zero value to accumulate time instead of a non-zero value Retentive - Retain the Hour Meter value when it is not accumulating. Units - The time units to display. • Event Meter - Record the number or times the Item value is non-zero. The datatype should be set to integer when using an Event Meter regardless of the OPC Item type. On Zero - Use a zero value to accumulate events instead of a non-zero value
Evaluation Mode	Run-always (ignore Trigger) - When selected, this causes the group to evaluate at each group interval, before the trigger state is evaluated.
Write Target	Target Type - This is the selection for what the Item will write to when the group executes. • None, read-only item - Do not write this value to the database. • Database field - Write the Item value to the specified column in the database table. • Other tag - Write the Expression Item's value back to an OPC item or Tag Reference.

Target Name	The name of the column in the database that this Item will write to when the group executes. The Target Name list will populate with all the OPC Item and Tag Reference names from this Group, or the column names from the Group's target table depending on the Target Type selected.
Numeric **These are the Numeric properties for Expression Items. For a full description, see Tag Scaling Properties.**	
Alarming **These are the Alarming settings for the OPC items. See Alarming Properties for a full explanation.**	
Expression **These are the Expression/SQL Query options for Expression Items. See Expression/SQL Properties for a full explanation.**	

3. Select the **Expression/SQL** tab from the menu on the left of the **Edit group tag** window.
4. Select **Expression** from the dropdown menu.
   Expression items can use the same functions as other expression items in Ignition. An expression item returns in accordance with the created expression. The resulting value can be written to a transaction group Item such as a tag or it can be written to a database.
5. Select the **General** tab and configure the **Write target** to instruct the transaction group what to do after the expression is complete.

To use an Expression Item as a transaction group trigger

1. In the **Run-Always Expression Items** location of the transaction group, right-click and select **New Item > New Expression Item**.
2. On the **Edit group tag** window, set the following properties to create a new expression item that returns a boolean true or false depending on the result of the expression item:
   Data Type: **Boolean**
   Mode: **Direct value**
3. Select the **Trigger** tab of the transaction group's properties on the right of the transaction group.
4. Select the **Execute this group on a trigger** box, and from the drop-down of **Trigger on Item** select the name of the new expression item.
5. Properly configure the **Trigger conditions**.

Next...

- SQL Query Expression Items

SQL Query Expression Items

Expression items can be SQL statements that handle select, update, insert, or delete statements from a database. More complicated queries can require you to set the query type manually. If the query is simple, Ignition automatically detects the query type. SQL query expression items can serve as triggers for the transaction group or they can only execute when the transaction group is triggered.

SQL Query Expression Items

Watch the Video

To trigger a transaction group with a SQL query Expression item

1. In the Transaction Group workspace, create a New Expression item.
2. On the **Edit group tag**, go to the **Expression/SQL** tab, and select **SQL Query** from the dropdown list.
3. Enter a SQL query into the Expression item, and click **OK**.
   The SQL query must return a numerical value.
4. Go to the **Trigger** tab, from the **Trigger on item** dropdown list, select the Expression item that has the SQL query.
5. Properly configure the trigger conditions and start the transaction group.

SQL Statements in Transaction Groups

It is important to be mindful of expression items that perform SQL statements. A transaction group that executes repeatedly can also be executing multiple expression items. If these expression items are executing a SQL statement, the query count on a database can increase dramatically.

Next...

- Run Always vs. Triggered Items

Run Always vs. Triggered Items

In the Transaction Group workspace, you can set the Expression items in a group to execute in two different ways:

- **Run-Always** in which the Expression items will execute at the groups rate (based on the **Execution Scheduling** settings on the **Action** tab).
- **Triggered** in which the Expression items run when the group is triggered (based on the settings on the **Trigger** tab).

If on the **Trigger** tab, the option **Execute this group on a trigger** is not selected, the items in the **Run-Always Expression Item** area and **Triggered Expression Items** area will all execute at the same time. The advantage selecting the **Execute this group on a trigger** option is that is gives you flexibility on when the expression items are executed which may result in easing the load on a database.

Run Always vs. Triggered Items

Watch the Video

Next...

- Hour and Event Meters

Hour and Event Meters

Hour meter and **Event meter** refer to the **Value Mode** option settings on tags in Transaction Groups. The Value mode drives values that are used to create values that determine how long a value was true. While the selected Value Mode for most transactions is **Direct**, however, the **Hour meter** mode accumulates value for the duration of a condition, and the **Event meter** accumulates count in response to the condition.

Hour and Event Meters

Watch the Video

Hour Meter

It is common to write to a tag during the time when a tag's value is true. An hour meter simplifies this effort. Hour meters can be meters that accumulate the millisecond, second, minute, hour, or day.

To count the duration of a tag being true

1. From OPC Browser or Tag Browser, drag a boolean tag into the **Basic OPC/Groups Items** area of a **Standard** transaction group.
2. From OPC Browser or Tag Browser, drag a memory tag or OPC tag (must be a numeric data type) into the **Basic OPC/Groups Items** portion of the **Standard** transaction group.
3. Right-click on the boolean tag in the transaction group and select **Edit** to edit it.
   The **Edit group tag** window is displayed.
4. In the **Edit group tag** window, for **Value Mode**, select **Hour meter**.
5. In the **Edit group tag** window, for **Target Type** select **Other tag** from the dropdown menu, and in **Target Name** enter the name of the memory tag as the target, and click **OK**.
6. In the **Basic OPC/Groups Items** area where the tags are located, go to the **Target Name** colum n, left-click on each tag to get the dropdown menu, and set the following:
   a. For the boolean tag, select the memory tag from the dropdown which is set previously as the target tag to write the hour meter to.
   b. For the memory tag, select Read-only from the dropdown.
7. Click **Enabled** at the top of the page, and do a **File > Save** to start the group.
8. Make the boolean tag true to the start the Hour meter.

Event Meter

Another common scenario is to count the number of times an event occurred. For example, where there is boolean tag and you want to count the number of cycles the boolean tag has experienced.

To count in response to a tag being true

1. From OPC Browser or Tag Browser, drag a boolean tag into the **Basic OPC/Groups Items** area of a **Standard** transaction group.
2. From OPC Browser or Tag Browser, drag a memory tag or OPC tag (must be a numeric data type) into the **Basic OPC/Groups Items** portion of the **Standard** transaction group.
3. Right-click on the boolean tag in the transaction group and select **Edit** to edit it.
   The **Edit group tag** window is displayed.
4. In the **Edit group tag** window, for **Value Mode**, select **Event meter**.
5. In the **Edit group tag** window, for **Target Type** select **Other tag** from the dropdown menu, and in **Target Name** enter the name of the memory tag as the target, and click **OK**.
6. In the **Basic OPC/Groups Items** area where the tags are located, go to the **Target Name** column, left-click on each tag to get the dropdown menu, and set the following:
   a. For the boolean tag, select the memory tag from the dropdown which is set previously as the target tag to write the hour meter to.
   b. For the memory tag, select Read-only from the dropdown.
7. Click **Enabled** at the top of the page, and do a **File > Save** to start the group.
8. Make the boolean tag true to start the Event meter.

Next...

- Resetting Hour and Event Meters

Resetting Hour and Event Meters

You can set the hour or event meter based on a condition.

To reset an hour or event meter based on a condition

1. In the **Basic OPC/Group Items** section, right-click and **Edit** a tag that is serving as the Hour or Event meter.
   The **edit group tag** window is displayed.

2. In the **Value Mode** area, select the **Reset on condition** check box, click the **tag** icon to display the **Choose Tag** window, and select a **Group Tag** from the pop-up window.

3. Click **OK**.

4. Next to the **tag** icon, choose the **operator** sign, and enter a number.

5. Click **OK**.
   The target tag will now reset in response to the condition.

INDUCTIVE UNIVERSITY

Resetting Hour and Event Meters

Watch the Video

Next...

- Trigger - On Value Change

Trigger - On Value Change

It is often useful to execute a group only when the values of the group's tags have changed, or when a particular tag within the group has changed. In either case, the transaction group will execute every time the value or values change.

Trigger – On Value Change

Watch the Video

To execute only when the tag values change

1. In the **Trigger** tab, select at the very top the **Only evaluate when values have changed** checkbox.
   Now the group executes only when all the tags change. But you may want to execute the group when only one tag changes.

2. To execute when only one tag changes, from the **Tags to watch for change** dropdown, select **Custom**, click on **Select tags**, select the tag from the pop-up window, and click **OK**.
   You can select more than one tag at a time in order to monitor more than one tag for value changes.

3. From the **Trigger on item** dropdown, select the appropriate tag to execute the tag on a trigger.

4. Select the **Active on value change** radio button.

Next...

- Trigger - While Condition is True

Trigger - While Condition is True

Groups can execute while a condition is true resulting in the transaction group continuing to execute for the duration of this condition.

To continue execution while the condition is true

1. Create a transaction group, and drag a numeric or boolean tag into the **Basic OPC/Group Items** section.
2. From the **Target Name** column dropdown, select **Read Only**.
3. Go to the **Trigger** tab, and select the **Execute this group on a trigger** checkbox.
4. In the **Trigger conditions** area, set the trigger conditions which will determine under what condition the group executes.

Trigger – While Condition is True

Watch the Video

Next...

- Trigger - Rising Edge (Once)

Trigger - Rising Edge (Once)

Groups can execute only when the trigger is increasing in value. This is known as a rising edge trigger and it will only execute once and will not execute again until the trigger repeats the same cycle.

Trigger – Rising Edge (Once)

Watch the Video

To execute the group only on the rising edge

1. Create a standard transaction group with any number of tags as long as one of them is a boolean tag that will serve at the trigger for the group.
2. Set the **Write Target** for the boolean tag to **Read-only** by selecting read-only from its drop down in the Target Name column.
3. Go to the **Trigger** tab and select the check box to **Execute this group on a trigger**. Select the boolean tag from the drop down menu and select to have the group only execute once while the trigger is active.
4. Save the transaction group to start the group.

Next...

- Trigger - Reset Trigger

Trigger - Reset Trigger

Resetting a trigger after execution of a triggered transaction group will result in the transaction group writing once to the targets followed by writing back to the trigger to reset it.

To reset the trigger after execution

1. Create a transaction group with a boolean tag. The write target for this tag should be read only.
2. Select the **Trigger** tab and select the **Execute this group on a trigger** check box.
3. Select the **Reset trigger after execution** check box.
4. Press save to start the transaction group.

Trigger – Reset Trigger

Watch the Video

Next...

- Trigger - Handshakes

Trigger - Handshakes

When a group executes, it either completes successfully or an error prevents its execution. The outcome of an execution can be handled in the handshake section of the trigger section of the transaction group. When a group executes successfully or fails to execute, the handshake can write a value back to a tag to alert the user that the group executed successfully or unsuccessfully.

To set handshake values for alerting the user

1. Create a transaction group with a boolean and a numeric tag.
2. Set the boolean and the numeric tag to read only.
3. Go to the **Trigger** tab and choose to **Execute this group on a trigger**.
4. Select the boolean tag as the trigger In the **Trigger on item** drop down, and select the appropriate execution conditions.
5. In the bottom section, select **Write handshake on success**, select the numeric tag to write to, and choose a number that signifies success.
6. Likewise, in the bottom section, select **Write handshake on failure**, select the numeric tag to write to, and choose a number that signifies failure.

INDUCTIVE UNIVERSIT

Trigger – Handshakes

Watch the Video

Next...

Transaction Groups Example

There are four types of transaction groups: **Historical**, **Standard**, **Block**, and **Stored Procedure**. This examples in this section shows the different ways that you can use them.

In this section...

Basic Historical Group

A basic historical transaction group is any transaction group that logs data to a database.

To create a Historical Group

1. In the Project Browser, right-click on the **Transaction Groups**, and select **New Historical Group**.
2. Select tags from the OPC Browser and drag them into the **Basic OPC/Group Items** area of the transaction group.
   The target databases are represented in the **Target Name** column and when the group executes, the data will be stored in the columns of these names.
3. Go to the **Action** tab and select **Timer** to specify how often the group executes.

   There are other options that help configure the data as it enters the database. For example, having an additional column for the time stamp.

Basic Historical Group

Watch the Video

Next...

- Scheduled Group

Scheduled Group

You can log data to a database at specific times using the execution scheduling. You create the **Scheduled** group the same way as you create a basic **Historical** group. But instead of executing on a **Timer**, you select the **Schedule** radio button, and enter the times in the text box and the rate.

The following table shows examples of how you can enter numbers in the text box:

Times	Rate	Result
8, 11, 5pm	n/a	Group will execute 3 times at 8 in the morning, 11 in the morning, and finally at 5pm.
8	n/a	Group will execute once at 8 in the morning.
8-10	1 hour	Group will execute at 8, 9, and 10 in the morning.
8-10, 17	1 hour	Group will execute at 8, 9, and 10 in the morning and once at 5pm.

Scheduled Group

Watch the Video

Next...

- Realtime Group

Realtime Group

Using transaction groups to update specific rows in a database is an easy way to expose OPC data to systems that cannot access OPC server but can query a SQL database.

To update the first row in a database

1. Create a new Standard group and drag your OPC tags into the **Basic OPC/Group Items** area.
2. On the **Action** tab, select the appropriate Timer, Data source, Table name, and other various settings.
3. In the **Table action** area of the **Action** tab, select **update/select**, and **first** which will continue to update the first row of the database according to the timer.
   To update a specific row rather than the first row in the database, select the **custom** or the **key /value pairs** options.
4. Select the group, click **Enabled** and do a **File > Save** to start it.
   The transaction group will only write the values that have changed to the database thus reducing the overhead.

Realtime Group

Watch the Video

Next...

- Bi-directional Group

Bi-directional Group

A bidirectional transaction is any transaction that synchronizes data between sources. It is not a transaction that inserts values into a database like a historical group. Bidirectional transactions are useful in a number of cases. A bidirectional transaction can update a database table with the values from an OPC location thus making the values accessible to systems that have the ability to read from databases but not OPC locations. Likewise, bidirectional transactions allow external systems to update OPC locations by interacting with the database.

A transaction group designed for bi-direction synchronization evaluates and handles the updates at the group's rate, for example every second. In the event that both the database and the OPC locations experience a change between the executions of the transaction group the group decides to synchronize the database and the OPC location in accordance with the selected update mode. The update modes can also be individually tailored for each tag and database column by choosing the correct update mode in the Mode column of the Basic OPC/Group Items section.

Bi-directional Group

Watch the Video

To create an bidirectional transaction group

1. Create a **New Standard Group** and drag tags from the OPC server or the Tag Browser into the group's **Basic OPC/Group Items**.
2. Go to the **Action** tab, and set the following:

   Update mode: either **Bi-directional OPC wins** or **Bi-directional DB wins**
   Data source: the appropriate source
   Table name: the appropriate name
   Automatically create table: select the check box so a table is automatically created if the table name does not already exist
   Table Action area: select **update/select** and **first**. You can also, synchronize the OPC and database locations based on a condition by choosing **custom** or the **key/value pairs** options.
3. Click **Enabled** and do a **File > Save** to start the group.

Next...

- Stored Procedure Group

Stored Procedure Group

A stored procedure group will interact with a database stored procedure. A stored procedure in a database is like a function. The procedure has a name and it may or may not take parameters. Once the procedure is called and the parameters passed into the procedure, the query executes on the database server and a the resultset is returned. The actual text of the query is stored and managed on the database server. Stored procedure provide an increased level of security and maintenance that make them ideal for certain scenarios.

Suppose the following stored procedure already exists on the database server:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `tankCount`(in invar varchar
(45), out outvar integer)
BEGIN
        select count(area) into outvar from test.tanks where area =
marker1;
END
```

INDUCTIVE UNIVERSITY

Stored Procedure Group

Watch the Video

And suppose the procedure is called with the following SQL statement and returns the number of tanks that are associated with Area A.

```
call tankCount("Area A")
```

This is a very simple scenario. Often stored procedures contain very long or complicated queries. Lets say you want to use a transaction group to call this stored procedure and then to handle the result by writing to a tag location.

To use a transaction group to call a stored procedure

1. Create a new **Stored Procedure** transaction group.
2. Drag two tags into the **Basic OPC/Group Items** section of the group. The two tags must be of the same datatype as the stored procedure's parameters. In this case the two tags will be a string and an integer type.
3. Go to the **Action** tab of the transaction group, select the appropriate **Data source.** Select the procedure name from the procedure name drop down menu.
4. In the **Basic OPC/Group Items** section, from the **Target Name** column dropdown of the string tag, select the input parameter, and select None for the output.
5. In the **Basic OPC/Group Items** section, from the **Target Name** column drop down of the integer tag, select Read-only, and select the output parameter for the output.
6. Click **Enabled** and do a **File > Save** to save the project and to start the transaction group.

Note: When running a Stored Procedure from an Oracle database, you cannot use named parameters. Instead you must use the index (number) arguments. IE: 1 is the first argument, 2 is the second, etc.

Next...

- Block Group

Block Group

The Block group is a type of transaction group that stores data vertically. Whereas, a Standard group stores the information horizontally in a single row. Block groups share many of the same features as the Standard group. They can be bidirectional, insert into a database, or simply update the database. All the rows in a Block group are associated with a single database transaction therefore the process of writing to the database is very efficient.

INDUCTIVE UNIVERSITY

Block Group

Watch the Video

To create a Block group

1. Create a new **Block** transaction group.
2. Drag a tag folder into the **Block Items** section of the new transaction group.
3. Select the item in the Block group, right-click and select **Edit**.
4. Change the **Name**, enter the **Target Name** to anything appropriate, and configure the remainder of the group settings under the **Action** tab.
5. Select the group, click **Enabled** and do a **File > Save** to start it.

Next...

- Recipe Group

Recipe Group

You can use Standard or Block Transaction Groups to create recipe management system with bi-directional communication to pass database information to the PLC when requested. The main advantage of using transaction groups to design the recipe management group is that the group will write to all the OPC locations.

INDUCTIVE UNIVERSITY

Recipe Group

Watch the Video

To use transaction groups to create recipes

1. Create a new **Standard** transaction group.
2. Drag a set of tags into the group's **Basic OPC/Group Items**.
   One of these tags should be a recipe index tag and needs to be read-only.
3. Change the recipe index tag to read-only by selecting **Read-only** from the drop-down in the **Target Name** column.
4. Go to the **Action** tab, and set the following:
   a. **Update mode**: choose **OPC to DB**.
   b. **Table name**: select the table name that contains the recipes. This table should have an index.
   c. **Use custom index column**: select the check box and choose the column that is the index from the table.
   d. **Store timestamp to a column**: deselect the check box.
5. In the **Table action** area of the **Action** tab, click **update/select** and **custom** to create a custom clause that references the read-only tag. The read-only tag and the name of the index column of the database must be the same. Do not include the word "where" because it is assumed to be part of the enclosed statement.
6. Click **Enable** to enable the group and do a **File > Save** to save the project and start the transaction group.
7. Test the recipe group by entering a new value into the recipe number (for example, the tag that is mapped to the index of the recipe database).
   The recipe values will download to the appropriate tags.

Next...

- Update or Insert Group

Update or Insert Group

You can update a row or insert a new row into the database when a key pair combination does not exist. This eliminates the need to have a database that has every possible option considered in its original design. Because of the **insert row when not present** setting, the group will insert a new record whenever the designated ID doesn't exist. Afterwards, it will update the rows in the table that are associated with the key/value references as shown in this example.

To update or insert a new row into the database

1. Create a **New Standard Group**.
2. Drag a group of tags into the groups **Basic OPC/Group Items** section.
3. Change one of the tags to be read-only by selecting **Read Only** from the tag's **Target Name** column.
4. Select the appropriate settings under the groups **Action** tab.
5. In the group's **Action** tab, in the **Table action** area, select the **update/select** radio button.
6. In addition, select the **key/value pairs** radio button.
   a. For the **Column** select the database table ID column.
   b. For the **Value** column, select the read-only tag.
   c. Select **Insert row when not present** check box at the bottom of the **Table action** area.
7. Click **Enable** and do a **File > Save** to save the project and to start the group.

INDUCTIVE UNIVERSITY

Update or Insert Group

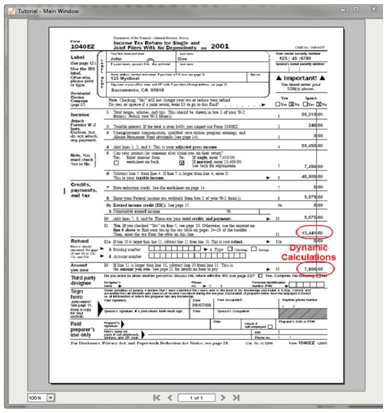
Watch the Video

Next...

Reporting

See legacy documentation for 7.7 Reporting module.

Welcome to Ignition Reporting!



Complete Reports on Existing PDFs

Ignition Reporting is a module for creating dynamic reports! These reports may be generated from existing Adobe Acrobat (pdf) files or created totally from scratch. Data is introduced through Ignition, providing access to any SQL database!

Ignition Reporting makes creating professional reports easy with a rich library including: images, graphs, tables, and a variety of basic shape tools. The new Report Scheduler allows automatic report generation at any time you'd like, and automated distribution means it gets where you need it without lifting a finger. Access to live reports available through the web based Ignition runtime, a Java application, providing authenticated users access from anywhere, all based on networking standards that your IT department can support. Reports are printer friendly and can easily be exported to a variety of formats including pdf! Here are some common uses of dynamic reports:

- Production Management
- Efficiency Monitoring
- Downtime Tracking
- Statistical Process Control
- Quality Assurance
- Overall Equipment Effectiveness (OEE) Management
- Historical Data Analysis

Benefits

Ignition Reporting enables managers to increase productivity, decrease waste, reduce costs, and increase quality with existing resources by providing a view of their manufacturing process. Managers often save time by automating reporting processes that were once done by hand. Often valuable man hours that went into creating spreadsheets or reports can be recovered! These reports are trivial to manage since they are generated on the fly from existing SQL database data.

Features

New in Ignition 7.8, with Reporting Module v3!

- The Schedule Engine allows you to schedule when reports run, and what to do with the report when it's complete. Email, FTP, Save as File, use in a Script!
- Easy to use WYSIWYG (What you see is what you get) designer that includes an intuitive layout and drawing tools
- New Charting components are easier to configure!
- Barcodes now have 2D encoders for things like QR codes, Data Matrix and PDF417!

The most noteworthy feature of the Reporting Module is the fact that is integrated into the Ignition system. This provides access to Ignition data including any SQL database, allows as many concurrent clients as the hardware can handle, and shares authentication with your existing Ignition project all through simple but secure web-based access.

Other features:

- Powerful table tool that creates new pages to fit your data. It supports a wide range of features.
- Crosstab tool lets you quickly create cross-reference tables to visualize and identify relationships in your data.
- Reports are printer friendly.
- Every report can be saved by the user in a variety of formats including pdf.
- The Reporting Module includes the Row Selector and Column Selector components. Both are very useful when working with DataSets. They work especially well with Ignition graph and table components as well as the Report Viewer.
- The Reporting Module includes the File Explorer and PDF File Viewer components. These are very useful for viewing machine maintenance manuals or any other PDFs from within your project.

In this section ...

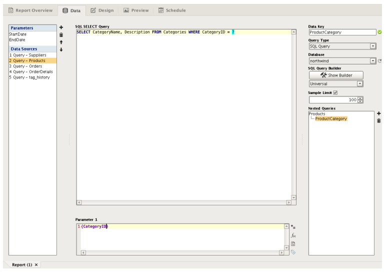
Reporting Basics

We've taken advantage of Ignition 7.8's powerful platform to bring you the Reporting Module v3. This new module is nearly a complete rewrite, with an upgraded reporting engine, completely new design interface, new components, and powerful data collection utilities. We know that Reports are increasingly important in modern automation projects and this new Module makes it easy for you to create engaging, informative reports that are easily tailored to fit your data. We look forward to seeing what you can build!

What's New?

This documentation is a great way to get familiar with all the new features in the Reporting v3. We'll cover all these and more in greater depth, this is just a partial list of what is new in the Reporting Module:

1. New Upgraded Report Engine
2. Completely New Report Workspace
3. Reports as Project-level Resources
4. Powerful Data Collection Utilities
5. Drag-and-Drop Query Builder
6. Improved Charting Components
7. 2D Barcode Generation
8. Powerful Scheduled Report generation
9. Familiar Property Editing
10. Scheduled Distribution Actions to Email, Save, FTP and More!
11. New Scripting Capabilities



Parameters: StartDate, EndDate

Data Sources:
1 Query - Suppliers
2 Query - Products
3 Query - Orders
4 Query - OrderDetails
5 Query - tag_history

SQL SELECT Query

```
SELECT CategoryName, Description FROM Categories WHERE CategoryID = ?
```

Parameter 1

```
{CategoryID}
```

Data Key: ProductCategory
Query Type: SQL Query
Database: northwind
SQL Query Builder: Universal
Sample Limit: 100
Nested Queries: Products └ ProductCategory

Easily Collect Structured Data!

Backwards Compatibility!

We've built Reporting Module v3 to be completely backwards compatible. If you have existing Report Viewers running in a project, they will continue to work as they always have without any modification. If you need to modify an existing report, you'll still have access to the same customizer and as you always did.

However, if you look at the Reporting Components in Vision, you'll notice some changes. The Report Viewer icon has changed and now creates a new version of the component.   This new Report Viewer acts as a viewer for your Report Resources, meaning you won't be able to create new Reports based on the old Component. We've also replaced the old PDF Viewing component with one based on our much more powerful IA-Labs PDF Viewer module.

We have made the upgrade seamless by maintaining your existing reports -- but old reports aren't *new* Reports! To encourage users to utilize the power of the new system, we've also provided a Report Converter. After you've gone through the documentation and have an idea of how the new Reports operate, take a look at our Conversion Notes page to see how you can port your existing Vision report to a new Project Resource to take advantage of all the new features.

In this section ...

Reporting Interface

Reporting v3 brought some significant changes to the way Reports are structured. No longer is the Reporting Module simply a Vision add-on. Of course, you retain the ability to add reports to Vision Windows, but it's also possible to create complete reports without having dependencies on other modules because Reports are now Project-Level Resources. Once you install the module, you'll be able to open the Ignition Designer and immediately see a new root resource folder. To create a new report, right-click on the root Report node in the Project Browser and a new Report Resource will be created.

Reports as Project Resources means that you are no longer reliant on Ignition Clients to generate reports. Reports now run on the Ignition Gateway, allowing completely headless generation and distribution. As detailed in the Scheduled Reports section, your reports can now run automatically!

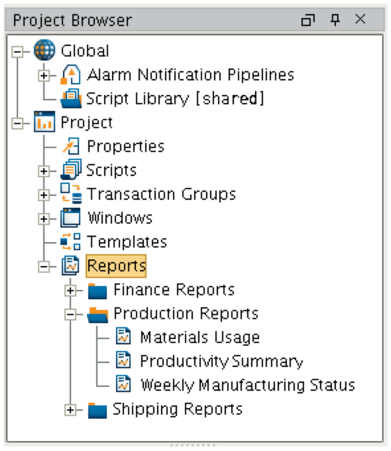
INDUCTIVE UNIVERSIT

Reporting Interface

Watch the Video

New Resource Means New Workspace

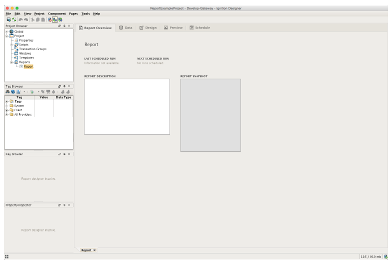
When we set out to create a new Reporting Module, we took a step back and tried to imagine what the ideal report system could provide. We thought about the different stages of report design, the features needed for powerful reports, the many types of data that we may want. We also listened to a lot of user feedback.   We were amazed in the variety and quality of reports people created with the old Report Module, but repeatedly heard customers ask for something easier, something that felt more like the rest of Ignition. Trying to distill all these requirements into a singular workflow was no small task, but we believe this new workflow will be understandable to the new developer, yet powerful enough for a seasoned analyst. Make no mistake, reports have a lot of moving parts, but fear not! You'll be able to leverage our new tools to create impressive reports faster than ever!



Reports are Project-Level Resources in Ignition 7.8

A New yet Familiar Workspace

So how do you create reports in Reporting v3? The new Report Workspace is the answer!   Built from the ground-up to be intuitive and familiar, the new workspace has a logical workflow which makes it easy for anyone to create meaningful reports.   From the first informative panel to full previews of the report you are creating, we believe this product will significantly cut the time it takes to design, edit and distribute reports. Lets take a look at the different parts of the new workspace.



The Reporting Workspace

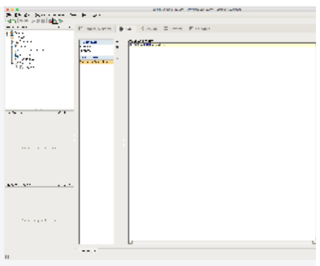
Report Overview

The first tab should be fairly self-explanatory – it provides some valuable quick-glance information about your report. There isn't much to see in a new report, but it can be a great introduction to an existing report for those new to a project, or if someone wants to quickly verify the schedule execution status of the report. There is space to add notes, which is a good place to provide background information and context about the report.   The Overview will also show a thumbnail of the most recent report, its last execution time, and the next scheduled execution. The Report Snapshot is generated per visit to the Preview Panel.

Data Panel

The Data panel is the first stop for configuring a new report. You'll notice the list on the left that has labels for Parameters and Data Sources, as well as some buttons. It is on this page where we can configure what data we want to make available to our report. The emptiness of a fresh Data Panel is *very* deceiving! We'll dive more into Parameters and Data Sources, but for now go ahead and click the + butt on. You'll see a list of built-in Data Sources and Parameters. Selecting any one of them will open the configuration screen for that Data Source type and allow you to start configuring the data you'll supply to your report.

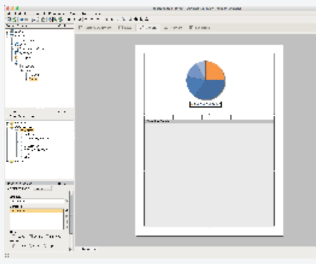
By the end of this guide, you'll be comfortable adding data sources to power your dynamic report.



The Data Panel

Design Panel

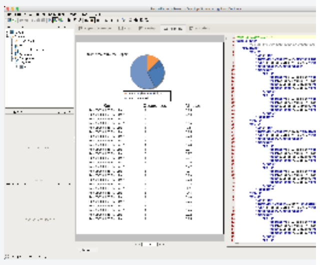
The Design Panel is something we expect to be a relief to those who felt the previous Vision Reporting Component didn't *feel* like Ignition. If you've ever designed a Vision Window, this should feel right at home. Starting from the top-left – the structure of a Report is now represented in the Project Browser tree, allowing easily visualization of the tree of elements on your report. But, we didn't stop there! Grab a component off the vision-like reporting palette and add it to the page. You'll see on the bottom left that we've completely re-written how you configure report objects. For complex components, we have built custom configuration panels to supplement the Property Inspector table that all the design objects have. In place of the Tag Browser, we now have a Datakey Browser that gives you an easy way to add the data sources you configured in the Data panel to the Report.



The Design Panel

The Preview Panel

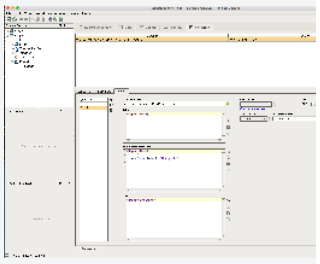
The Preview Panel, while not one of our 3 Steps, is a huge help while building your Report. By choosing to limit the amount of Preview data to return on the Data Panel, you'll be able to get fast visual feedback on your report without long-running queries. Not only does it provide you an instant example of what your report looks like, but it also gives you the ability to view the actual data that is being sent to your report! Having a snapshot of your data can be incredibly helpful when trying to figure out why your report doesn't appear as you expect it to. Whether you have a error in your Query, or a bad Data Key in a component, you'll be able to figure out the structure of the data your report is getting and quickly find ways to solve any problems.



The Preview Panel

The Schedule Panel

If you tried to schedule the execution of a Report on previous version of the Report Module, you likely found that there really wasn't a simple way to succeed. With the new Reporting module, scheduled reports are a built-in and easy to use. The Schedule panel is an incredibly powerful way to automate the execution and delivery of your Reports. Whether you have extremely long running queries that you don't want tying up a window, need to deliver to multiple file servers via FTP, or want to simply email to an existing roster, this panel provides you the means to do it. It's broken up into two major areas. On top, you find a simple table of schedules that have been established. To add a schedule, click the + button. With a schedule added, you can now choose when and how that scheduled Action will occur. Our schedules use the common Chron format, but don't worry if you aren't familiar with it. The preset options and combo-box configurability allow you to easily create just about any schedule you can imagine.



The Schedule Panel

Once you selected your schedule, you can add parameters that may affect your Actions. Actions are incredibly powerful and allow you to choose how you want to distribute or act on your finished reports. Whether saving a file locally, posting to an FTP server, or emailing (after you have configured your email connections in the gateway), you should be able to get your report where it needs to be. If our normal distribution Actions somehow fall short, you can choose to act on the finished report by triggering a script! We've also given you a convenient way of immediately testing your Action by simply clicking the button on the Action tab.

Next ...

- Report Data Basics
- Report Data
- Report Schedules

Report Data Basics

The Data Panel is an incredibly powerful tool for gathering data for your reports. In this section we'll go through some of the ways we can use the tools in the Data Panel to collect data from any source you may have.

INDUCTIVE UNIVERSIT

Report Data

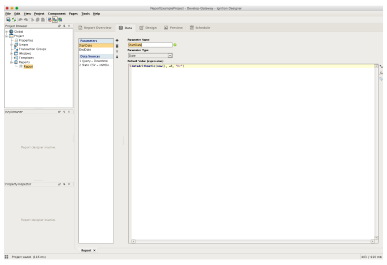
Watch the Video

Parameters

As we covered in the overview, the Data Panel is pretty simple at first glance. When creating a new report, there is an empty list of Data Sources as well as pre-made Parameters.   *Parameters* are Ignition *Expressions* . These parameters resolve at report execution time and provide a convenient way to dynamically specify content for your report. These expressions are expected to resolve to specific types.

The Parameter Types available in the ComboBox will be familiar to anyone who has used Expressions, and include:

1. **Date**
2. **String**
3. **Long**
4. **Double**
5. **Boolean**
6. **Dataset**



The Parameter Editor

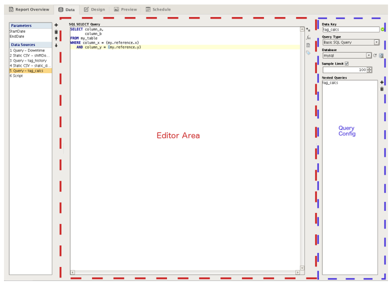
What are Parameters Useful For?

Parameters share some similarities to Custom Properties in Vision components in that they allow an Ignition user to specify dynamic data to be used in reports at runtime. For example, a manufacturer may want to generate reports for the active production lines every Monday morning, but you don't know ahead of time which production lines will be running. We can create a parameter called *activeProductionLines* a nd use an expression or tag value to determine which lines were active, and only use the parameter in a SQL Query datasource to only include data from active lines.

Parameters can be declared with an empty (null) value. In this event, it can be fed values from the Vision Report Viewer component, or in the Report Scheduling panel. Similar to how custom properties work with the old Report Component, the new Report Viewer can use parameters to bind to data through Ignition Bindings. If a parameter is supplied a default value in the Report Data panel (we call these *defaul t parameters)*, that value will be automatically be supplied to the Report Viewer component. If a parameter has no default, the Report Viewer and Preview Panel will notify the user that a parameter is undefined. It is important to note that both Scheduled Report and Vision component parameters take precedence over default parameters. So any default parameter can easily be overridden while also providing a baseline value.

Developer Tip

Give your Data Keys readable names that say something about the table or data they represent. It will make it easier to build meaningful reports!



A Query Data Source Panel

Data Sources

Data Sources are the primary means of getting data out of where it lives and into a report.   There are seven Data Sources included in the Reporting Module. There are five Query types, a Script data source and Static CSV. Each data source offers a unique method of collecting (or amending in the case of *Script* Source) data. Aside from the *Script* datasource, all will require you to specify a *Data Key.* The *D ata Key* is a unique identifier that represents the top-level of the data collected through this source – think of it as the label for the data source, or as the parent node of your source's collected 'data tree'. When your report is generated, the data collected under this identifier is passed to the reporting engine, which utilizes this identifier to appropriately place your data in the final report. We'll take a closer look at how to configure and reference this data in the *Reporting Workflow* section.

Query Datasources

Query Data Sources are the foundation on which most reports will be built. We have five query sources --

1. **SQL Query**
2. **Basic SQL Query**
3. **Tag Historian Query**
4. **Tag Calculation Query**
5. **Alarm Journal Query**

Queries share some common features in the UI, and in the fact that they all return what is essentially a Dataset.   If you take a look at the image on the left, you'll see we outlined two main areas of the Basic SQL Query data source type. By default the largest part of any Query panel is taken up by the central

Editor Area. This space is tailored for each query type. On the right, we have a "Query Configuration" area. In the screenshot, there are options for the *Data Key*, *Query Type, Database, Sample Limit* and *Ne sted Queries*. Of these, *Data Key*, *Query Type* and *Nested Queries* are shared among all Queries, while *Sample Limits* are conveniently available in SQL query types. Here is some brief information on each:

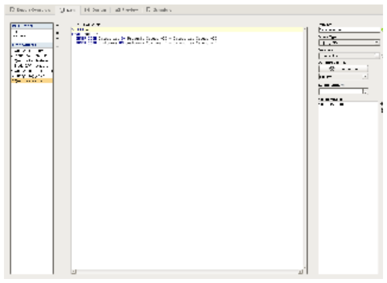
1. **Data Key** - The identifier we will reference when we design our reports. The Data Key needs to be unique, can not contain special characters (spaces, underscores, dashes are allowed), and must start with a letter. We have added a convenient validator to help detect poor data keys.
2. **Query Type** - gives you a convenient place to change the type of query
3. **Nested Queries** - This is one of the most powerful aspects of the new Reporting System. We will get into more detail on how they work in the next section, but in short: nested queries allow you to run queries that are dependent on return values from other queries. Utilizing a Parent-Child relationship, you a child query will execute for each row of its parent, and the parent's columns can then be referenced in the child.
4. **Sample Limit** - Allow you to choose to limit the number of rows returned from a query when a report is being generated in the Preview Panel, and when data is sampled for the purposes of generating Data Keys in our Designer. If supported by your database, this makes it much easier to craft and preview deep query structures without the overhead and wait-times of long-running queries.

Queries allow you to pull data from any connected database, historian and even alarm history.

Query Types

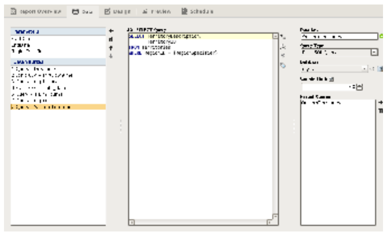
Here is a brief rundown on each query type.

SQL Query



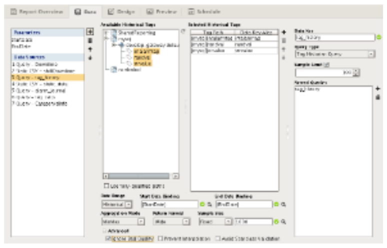
The SQL Query allows you to craft parameterized queries that run as a prepared statement. In addition, it also features the new *Query Builder*, which is a powerful Drag-and-Drop query building GUI that allows you to make complex queries from your connected databases. As Prepared Statements, these queries are more resistant to SQL injection offering additional security over basic queries. Craft your query, and for each variable you wish to evaluate, substitute a '?' and a new editor will be added below in which you can add static values, expressions, or references.

Basic SQL Query



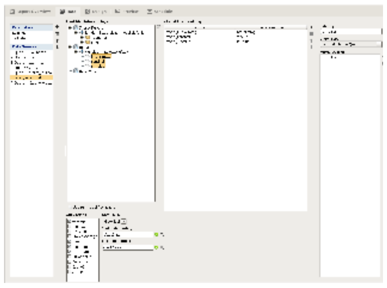
This query type is the common type historically seen through much of Ignition. You can write queries which include tag path references, expressions, or report parameters which resolve at run time.

Tag Historian Query



Provides a simple way to query data from Tag Historians. Utilizing a user interface familiar to Tag Historian users, you can collect data from Historical tags for specific date ranges, using aggregations, or for fixed sample sizes.

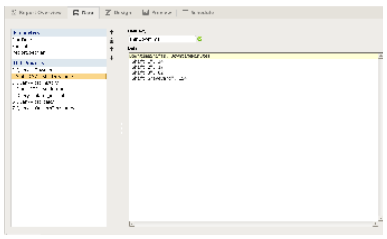
Tag Calculation Query



A new part of the Ignition Platform in 7.8, Tag Calculations is an extensible part of the API that provide values from calculations run against Tags. You can easily run several calculations on your historical tags, including Average, Sum, Maximum value, and many more.

Static CSV Datasource

The Static CSV data source is exactly what it sounds like – an opportunity to provide your report with a static set of CSV data.   We found this source to be helpful when trying to design reports without having database access. You can generate or create a simple CSV model of some of your data and test the visual layout of your report. Alternatively, it also provides a means of adding data from, for example, Excel or Access CSV exports. While not the most powerful datasource, it does make it easy to add external data.



Script Datasource

Always a popular choice for power users, a script can be used to collect data that is otherwise difficult to get through the Query types. Whether you need to pull data from files, crunch data collected from Queries using specialized Python libraries, or alter datasets from converted Reports, scripting is always a flexible way to accomplish your goals. In general, we recommend using alternatives when they exist, but if all else fails, a script will provide the flexibility you need to get data to the reporting engine.

Next...

- Report Designer
- Report Data
- Parameters
- Basic SQL Query
- SQL Query
- Tag Historian Query
- Tag Calculation Query
- Alarm Journal Query
- Static CSV
- Data Sources - Scripting

Report Designer

New Design Interface

INDUCTIVE UNIVERSITY

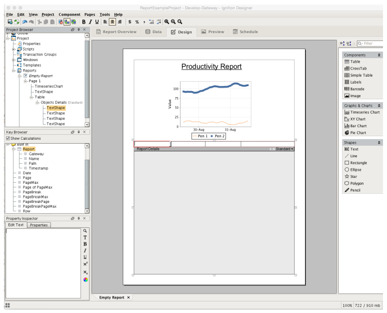
Report Designer

Watch the Video

The new Report Workspace's Design tab feels right at home in Ignition and has significant improvements in usability. Strides were taken to maximize the native feel of the workspace's Design tab while not requiring experienced Report module users to abandon their hard-earned knowledge. At the same time, our goal was to simplify the process and remove much the mystery of report design.

Looking at our screenshot of our Report Designer, you'll notice that an open Report Resource in the Project Browser can now be expanded to provide information at a glance. This tree lets you visualize the relationship and hierarchy of Report Design elements on the page. Find an element in the tree by selecting items on the page, or find an element on the page by selecting it from the tree. As you do so, you'll notice that the bottom left of the default Report Designer will change to provide configuration panels and/or property tables depending on the selected item. In between these two panels on the left is the *Key Browser* which provides the *Data Keys* and parameters we use to reference data in our report.

The right of the panel is where our design palette lives, providing a number of powerful components, charts, and shapes which we use to build the layout and visualization of our report.


The Design Panel

New Design Components in Reporting v3

While some of the components in the Report Designer are unchanged, we refreshed some and created a new configuration interface that is much simpler (and much more familiar) for Ignition users.

Barcodes

New Barcode component added support for 2D barcodes, including QR Codes, PDF 417 and Aztec and Data Matrix.

Charts

Charts were rebuilt from the ground up with simpler configuration, and more options. Plot report data in Timeseries, XY, Pie or Bar Graphs. Configure pens, axis and appearances in a familiar interface!

New Key Browser

Just below the Project Browser is the *Key Browser*. The Key Browser is home to all your Data Sources, the new Parameters, as well as a number of built-in calculation keys to speed you through the report design process. When you configure a data source and switch to the Design tab, a sample of all your queries is run on the Gateway to get information about the structure of your data. Columns in your data are represented as children in the Key Browser tree.

Object Layout

Object layout is an important aspect in creating a professional report. Ignition Reporting uses a WYSIWYG (what you see is what you get) approach.

Headers and Footers

Creating headers and footers is just like creating any other set of objects on your report. There is no explicit header or footer section. The key is sizing and positioning of components within your page around your header or footer. Each new page that the table creates will have that same header and footer.

Z Order

Z order defines relative order of objects when they overlap. Simply select the object and click "Bring to front" or "Send to back" in the Component menu

Object Grouping

Grouping makes a set of object behave as one with respect to: selection, moving, and resizing. To "drill down" to individual objects, superselect the grouped object.

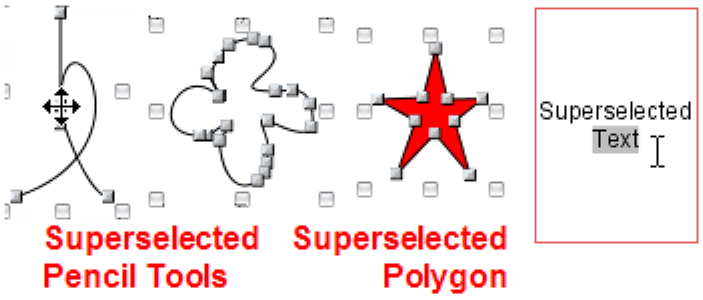
Selection and Alignment

Selection is done with the selection tool, which is the default tool in the Report Designer whenever another tool is not activated.

Reporting has a "deeper" selection model than the Ignition designer. Simple object selection is done by single clicking an object. "Selecting deep" is done by double-clicking to get into the report hierarchy. For instance, if you group two rectangles together, you can select the individual rectangles by double clicking "into" the group. With the release of Reporting v3 in Ignition 7.8, visualizing selection is simplified by viewing the Project Browser to see which node in the Report Resource tree is selected.

Superselection

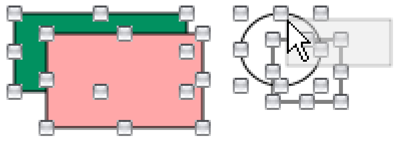
Superselection refers to an editing state that some shapes go into when double clicked. Text is the most common of these. When a text box is selected you can move and resize it. When it's super-selected, you can place the text cursor or select a range of characters and insert or delete text. The polygon and pencil are two other basic tools that support superselection.



Multiple Selection

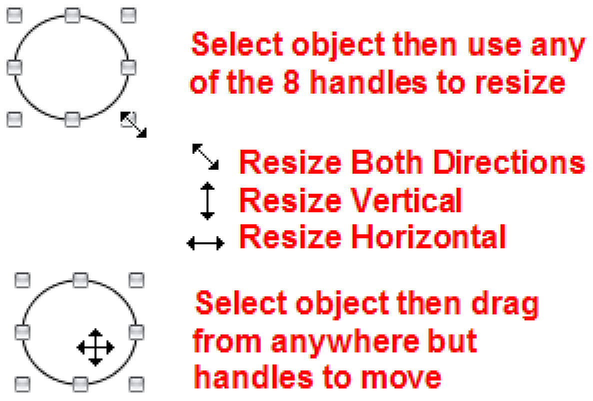
Multiple Selection can be done two ways:

- Clicking and dragging the mouse over a range of the report. Everything the selection rectangle touches becomes selected.
- Hold the shift key while making a selection or dragging a selection rectangle. Shapes hit by that action will be added or removed from the currently selected shapes.



Resizing and Moving objects

To resize or move an object first select it with a single click. To resize left click and drag one of the 8 resizing handles. To move the object, left click and drag anywhere on the object when it is selected. Both operations support shift dragging.



Alignment

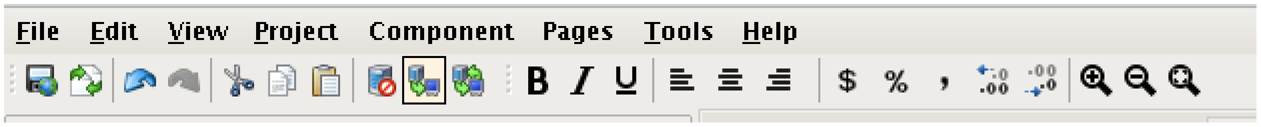
Alignment is accomplished by selecting multiple objects, then choosing "Make ..." from the Component menu or right click menu.

Menu Items	Function
Make Row Top/Center /Bottom	Quickly align several shapes in a row, either by their top, center, or bottom border. Useful when shapes are of different heights.
Make Column Left/Center /Right	Same as above, but for columns, aligning their sides or center.
Make Same Size, Width, Height	Make several shapes the same width, height or both.
Equally Space Row/Column	Equalizes the distance between shapes horizontally or vertically.

Shift Drag

Holding the shift key while you drag shapes will constrain movement to: horizontal, vertical, or 45 degrees.

Design Toolbar and Menus



The Ignition Designer menu bar provide quick access to many common Reporting Design functions. As can be seen in the screenshot above, the `Component` and `Pages` menu items are added to the when the Report Design Panel is open.

Edit

The Edit menu provides functions like cut, copy and paste.

Menu Item	Function
Undo	Undoes the last action.
Redo	Re-does the last undo (assuming nothing was changed after the last undo).
Cut/Copy/Paste	Allows you to easily duplicate or import document elements using the system clipboard.
Select All	Selects all elements at the current level of selection (or all text, if editing a text field).

Pages

The pages menu allows you to add or remove pages in the report.

Menu Item	Function
Add Page	Adds a page to the current open document, after the currently selected page.
Add Page Previous	Adds a page to the current open document, before the currently selected page.
Remove Page	Removes the currently selected/visible page in the current open document.

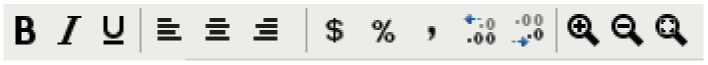
Component

This shapes menu allows you to modify the layout of objects in a report

Menu Item	Function
Group/Ungroup	Allows you to merge the currently selected shapes into a single shape for convenient management. Contained shapes are still

	accessible, via double-click super-select. Ungroup separates grouped shapes.
Bring to Front /Send to Back	All shapes have an order on the page that determines what is drawn on top when two shapes overlap. These options allow you to alter that order.
Make Row Top /Center/Bottom	Quickly align several shapes in a row, either by their top, center, or bottom border. Useful when shapes are of different heights.
Make Column Left /Center/Right	Same as above, but for columns, aligning their sides or center.
Make Same Size, Width, Height	Make several shapes the same width, height or both.
Equally Space Row/Column	Equalizes the distance between shapes horizontally or vertically.
Group in Switch /3D Shape	This feature groups selected shapes in a Switch Shape, which has the same features as Table Row Versions. It's a powerful way to conditionally provide a different look for a specific element.
Combine/Subtract Paths	Takes multiple overlapping shapes (such as a rectangle and an oval) and combines them into a single shape using the combined paths. A powerful tool to construct complex shapes.
Convert Into Image	Converts the selected shape into an image. Be sure to group shapes first if you want to convert multiple shapes into a single image.

Toolbar



The toolbar provides a variety of functions to assist in editing while designing a report.

Toolbar Item	Function
B	Toggles **bold** of the selected text
I	Toggles *Italic* the selected text
U	Toggles <u>underline</u> of the selected text
≡	Aligns the selected text left
≡	Centers selected text
≡	Aligns the selected text right
$	Adds a $ to the data key
%	Adds a % to the data key
,	Forces a decimal place on an integer value in a text shape
+.0 .00	Add the number of allowed decimal digits
.00 →.0	Subtract the number of allowed decimal digits
🔍+	Increase the size of the report in view

🔍	Decrease the size of the report in view
🔍	Have the report fill the available space

Next...

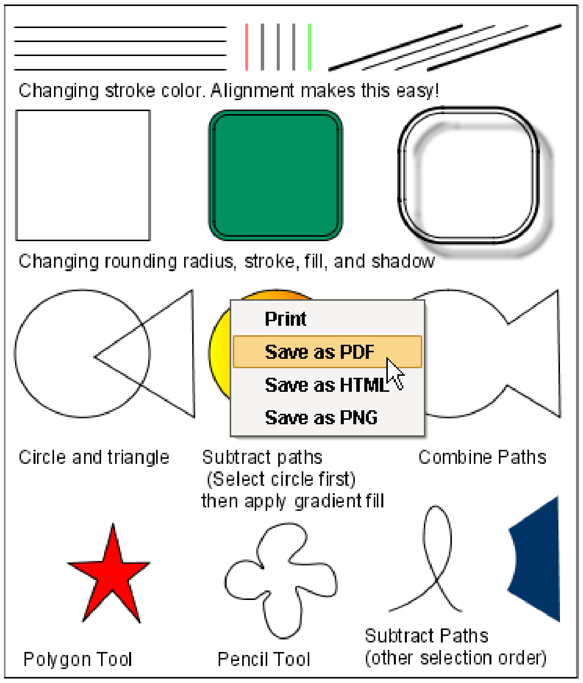
- Simple Report
- Report Charts

Simple Report

Reports in the runtime

Once created, you can view the report from the Ignition runtime or Designer preview mode with the Vision Report Viewer component.

Saving a report from the Vision Report Viewer is simply a matter of right-clicking on the report in the Ignition runtime or preview mode of the designer and selecting the format you wish to save it as. Selecting from the menu will open a Save or Print dialog in the Client window.


Changing stroke color. Alignment makes this easy!
Changing rounding radius, stroke, fill, and shadow
Print
Save as PDF
Save as HTML
Save as PNG
Circle and triangle
Subtract paths (Select circle first) then apply gradient fill
Combine Paths
Polygon Tool
Pencil Tool
Subtract Paths (other selection order)

The Vision Report Viewer component supports the following options:

- Print - send your report to a connected printer
- PDF - formatted for viewing in PDF Viewers
- HTML - an HTML document viewable in your web browser
- PNG - save as an image

Simple Report

Watch the Video

Next...

- Built-in Keys
- Other Reporting Module Components

Built-in Keys

Built-In Keys

A number of useful Data Keys are included by default. These tables lists the keys and provides a brief summary.

Built-In Report Data Keys

Key	Description
Gateway	Name of the Report's Ignition Gateway
Name	Name of the Report
Path	Path to the Report in the Project Browser
Timestamp	The Gateway's current timestamp

Built-in Keys

Watch the Video

Built-In Data Keys

Key	Description
Date	The current date/time
Row	The current row number (only in tables)
Page	The current page
PageMax	The total number of pages in the generated report
PageBreak	The number of explicit page breaks encountered
PageBreakMax	The total number of explicit page breaks in generated report
PageBreakPage	The number of pages since last explicit page break
PageBreakPageMax	The total number of pages in current explicit page break
Row	Shows the current row number. Must be used in a table

The "total2" key

An aggregate calculation will result in null if any of the individual values are null (rather than return a value that is technically incorrect). You can work around this by implementing a derived method that returns a default value if the original attribute is null and aggregating using that key/method. Also, most of the aggregates contain a second version ("total2") that assume that null is equal to zero.

The "count" and "countDeep" keys

The count keys tell us how many objects are in a given list or group. This is most commonly used for tables with one or more levels of grouping. If, for instance, you have a table of Movies grouped by their studio and you add the @count@ key to the studio details, it will display the number of movies for each studio. So it might make sense to have a text field with "@studio.name@ has released @count@ movies" (Warner Brothers has released 15 movies).

The count key only counts the next level of grouping. If you have multiple levels of grouping and want to count all the root entities use the countDeep key. Suppose you have movies grouped by their category and their studio, and want to display a top level summary. You could use: "@studio.name@ has released @countDeep@ movies in @count@ different categories" (Warner Brothers has released 36 movies in 7 categories).

Heritage Keys (Running Totals, percentage totals)

There is an additional set of keys in the Attributes Browser which are used to access upper level groups: Up, Running, Remaining. @Up.count@ would tell us how many objects are in the current level of grouping.

The text field "Row @Row@ of @Up.count@" might show "Row 1 of 5".

By doing some simple arithmetic and using the "Up" key, we can calculate a percentage total: "% Total: @revenue/Up.total.revenue@"

The running key references a virtual array containing all of the objects processed thus far in a lower level grouping. This is useful to get a running total. For example, in a ledger: "Credit/Debit: @amount@ Current balance: @Running.total.amount@"

The remaining key is conceptually the same, but results in a virtual array of remaining objects.For example: "Credit/Debit: @amount@ Remaining Activity: @Remaining.total.amount@"

Next...

- Reporting Component Properties

Reporting Component Properties

Reporting components, just like Vision components, have many different properties. Some properties are unique to certain components so it's always a good idea to look through all the properties of a component to see how they can be used to enhance your reports.

Some Report components, when selected, have an additional tab in the Property Inspector as you will see. The Reporting components that have an additional tab in the Property Inspector are shown here along with a brief description of their respective properties.

INDUCTIVE UNIVERSITY

Reporting Component Properties

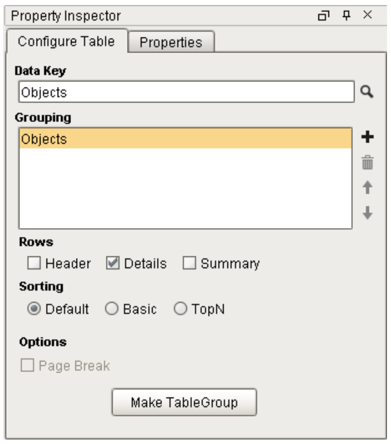
Watch the Video

Components

Table Component

The Report Table allows you to display tabular data in a variety of ways using the Table properties. Some of these same properties may also be modified from the Properties tab.

Configure Table Tab


Property Inspector
Configure Table | Properties
Data Key: Objects
Grouping: Objects
Rows: [ ] Header [✓ Details] [ ] Summary
Sorting: [● Default] ○ Basic ○ TopN
Options: [ ] Page Break
Make TableGroup

Configure Table Properties

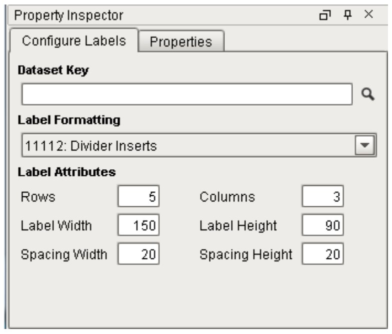
Property	Description
Data Key	Unique identifier or placeholder that points to your data. Typically a Data source is used for this property.
Grouping	Matches the name in the Data Key row that points to your data
Rows	Rows in a table. There are three types: **Header Row:** A single row inserted at the start of the Detail Rows. Commonly used to display column headers. **Details Row:** Represents the majority of the data on a table. **Summary Row:** A single row inserted after all the Detail Rows. Commonly used to display totals or summary values.
Sorting	Orders data in the table. Sorting occurs after the data has already been retrieved in the data source. There are three options: **Default:** Data sorted based on the order in which it is retrieved. When selected, the table will not sort the rows. **Basic:** Allows a list of keys to be configured. The data is sorted in the order the keys are selected.

	TopN: Uses a single key path, with a Count value that allows a limit to the number of rows that are processed
Page Break	Creates a page break between each grouping
Make Table Group	Can create 'child' and 'peer' tables.

Labels Component

Labels can be used to print out a host of different types of labels depending on how you configure the Label properties. You can create mailing labels, name tags, generic labels, and more. You can even use the Avery Label sheets. Some of these same properties may also be modified from the Properties tab.

Configure Labels Tab


Property Inspector
Configure Labels | Properties
Dataset Key
Label Formatting
11112: Divider Inserts
Label Attributes
Rows 5 Columns 3
Label Width 150 Label Height 90
Spacing Width 20 Spacing Height 20

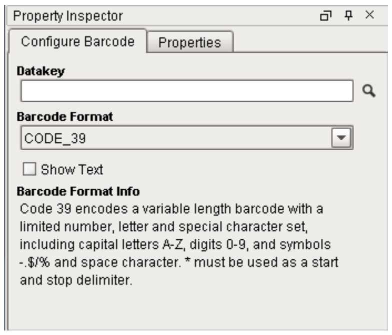
Configure Labels Properties

Property	Description
Data Key	Name of the key that will populate the Labels.
Label Formatting	Choose from a list of Avery Label Formats, or create a custom sized label.
Rows / Columns	Defines the number of rows and columns per page
Label Width / Height	Width and height of labels in pixels.
Spacing Width / Height	Distance between Labels on the page in pixels.

Barcode Component

The Barcode component allows reports to contain dynamically generated barcoded based on the how you configure the Barcode properties. Some of these same properties may also be modified from the Properties tab.

Configure Barcode Tab


Property Inspector
Configure Barcode | Properties
Datakey
Barcode Format
CODE_39
☐ Show Text
Barcode Format Info
Code 39 encodes a variable length barcode with a limited number, letter and special character set, including capital letters A-Z, digits 0-9, and symbols -.$/% and space character. * must be used as a start and stop delimiter.

Configure Barcode Properties

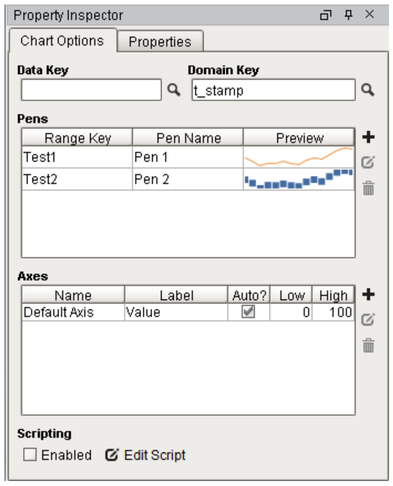
Property	Description
Data Key	Key that will populate the barcode.
Barcode Format	Barcode format to display
Show Text	Determines if the value of the Data Key should appear on the barcode.

Graph and Chart Components

The Chart Options tab contains all the same properties for both the Timeseries Chart and XY Chart. You will notice that the reporting properties for the Timeseries and XY Chart components contain some of the same properties that are in the Easy Chart Customizer. These same properties may also be modified from the Properties tab.

Timeseries Chart & XY Chart Components

Chart Options Tab



Property Inspector

Chart Options | Properties

Data Key

Domain Key t_stamp

Pens

Range Key	Pen Name	Preview
Test1	Pen 1	
Test2	Pen 2	

Axes

Name	Label	Auto?	Low	High
Default Axis	Value	✓	0	100

Scripting

☐ Enabled   Edit Script

Timeseries Chart Chart Option Properties

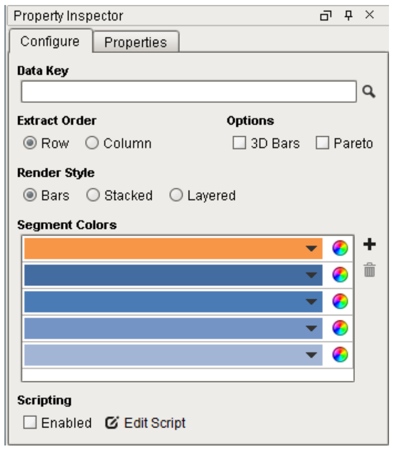
Property / Symbol	Description
Data Key	Unique identifier or placeholder to the data source that will populate the chart.
Domain Key	The data key that should represent the X axis on the chart. The Timeseries Chart expects a timestamp, while the XY chart can use any numeric value as the domain key.
Pens	Pens represent a series of data. There are several properties for each pen exposed on the Chart Options tab: **Range Key:** The key the pen should retrieve data from. **Pen Name:** The name of the pen that should appear on the chart. **Preview:** Visual representation of how the pen will be rendered. The trend is simply a preview, and does not take the data from the range key into account.
Edit Pen	Opens Pen Configuration area for editing.
Add Pen	Adds a pen.
Remove Pen	Deletes a pen.
Axes	There are several properties for each axis exposed on the Chart Options tab: **Name:** The name of the axis. **Label:** Text next to the axis. Commonly used to describe what the values of each pen represent. **Auto?:** Determines if the axis should automatically adjust the upper and lower

	bound of the axis to include all data from each pen. When enabled, the High and Low properties are ignored. When disabled, High and Low will be used. **Low:** The low range value the axis will use when 'Auto?' is disabled. **High:** The high range value the axis will use when 'Auto?' is disabled.
✚ **Add Axis**	Add an axis.
Edit Axis	Opens Axis Configuration area for editing.
🗑 **Remove Axis**	Deletes an axis.
Enable Scripting	Enables the script for the component to run.
Edit Script	Allows the chart to be modified before the report is rendered.

Bar Chart Component

The Configure tab is present when using the Bar Chart and Pie Chart components. The Configure properties are just as the name suggests; how to configure the look and feel of a chart; rows, columns, and segment colors on a chart. Some of these same properties may also be modified from the Properties tab.

Configure Tab


Property Inspector

Configure | Properties

Data Key

Extract Order: (● Row) ○ Column

Options: ☐ 3D Bars ☐ Pareto

Render Style: (● Bars) ○ Stacked ○ Layered

Segment Colors

Scripting: ☐ Enabled Edit Script

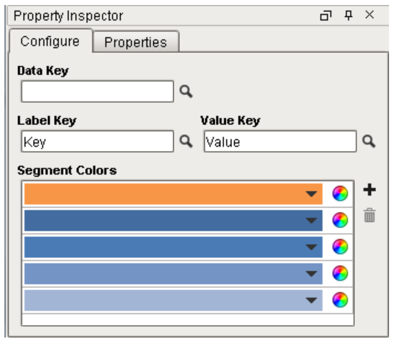
Bar Chart Configure Properties

Property /Symbol	Description
Data Key	Unique identifier or placeholder to the data source that will populate the chart.
Extract Order	The order in which the data should be extracted from the data key. Two options are available: **Row:** Data is extracted by row. Use this when columns in the data key define the series of data. **Column:** data is extracted by column. Use this when rows in the data key define the series of data; each column is a new value for the same

	series. This is sometimes referred to as data 'turned on its side'
Options	**3D:** Columns and bars with 3D appearance using angle and depth. **Pareto:** When enabled, turns the chart into a Pareto chart. Contains both lines and a bar graph where values are in descending order.
Render Style	Provides three rendering styles: **Bars:** Displays data as individual colored bars **Stacked:** Displays data as a stacked bar in different colors **Layered:** Displays data as a layered bar in different colors
Segment Colors	A unique color in the RGBA color space for each segment (bar)
Color Wheel	Tool for creating color combinations for bars
✚ **Add Segment**	Adds a color / row
Remove Segment	Deletes a color / row
Scripting Enabled	Enables the script for this component to run
Edit Script	Allows the chart to be modified before the report is rendered.

Pie Chart Component

Configure Tab


Property Inspector

Configure | Properties

Data Key

Label Key: Key

Value Key: Value

Segment Colors

Pie Chart Configure Properties

Property	Description
Data Key	Unique identifier or placeholder to the data source that will populate the chart.
Label Key	Key that will list the names of each segment in the legend on the chart.
Value Key	The key that will populate the chart with data.
Segment	A unique color in the RGBA

Colors	color space for each segment (wedge).
✚ **Add Segment**	Adds a color wedge to the chart
🗑 **Remove Segment**	Deletes a color wedge from the chart

Shapes

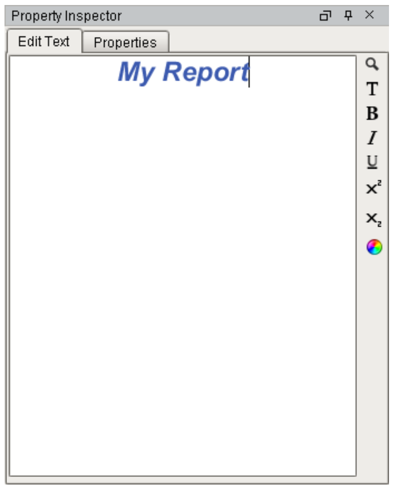
The reporting component properties for Shapes uses the Property Inspector. There are no other tabs for additional properties. The reporting component properties are used just like the ones in the Vision Module. Drag any of the properties on to your Design workspace to use in your report.

Property / Symbol	**Description**
Text	Click and drag to create a shape that will contain text.
Line	Click and drag to create a line
Rectangle	Click and drag to create rectangle
Ellipse	Click and drag to create n ellipse
Star	Click and drag to create a star
Polygon	Click points so that lines will be joined with straight lines
Pencil	Click and draw free-hand segments

Text Shape Component

The Edit Text tab is pretty straightforward to use. Text Shape properties can be modified by the buttons on the right side of the Property Inspector. These same properties may also be modified from the Properties tab. Properties that are modified will usually affect all text in that object regardless of specific text selection.

Edit Text Tab



Text Shape Component Properties

Property / Symbol	**Description**
🔍	Data Key search
T	Font, Font & Style
B	Bold
I	Italic
U	Underline
	Superscript

x^2	
x_2	Subscript
●	Font Color

Next...

- Key Calculations
- Report Tables
- Report Designer

Key Calculations

It's possible to perform calculations on data keys. This section documents the various operators and functions that are available.

Key Calculations

Watch the Video

Keychain Expressions

Keychains have their own expression language that is largely similar in syntax to Java. You can type in expressions within the "@" symbols to perform calculations on the keys. Here are the operators in order of precedence.

Operator	Function	Example
Parenthesis	(expr) Nested expressions	Any portion of a Key Chain can be enclosed with parenthesis to guarantee precedence.
Multiplicative	*, /, % Multiply, divide, modulo	These are the most common and intuitive operators. You might want to display @quantity*price@ in an invoice line-item or calculate a percent like this @profit/revenue*100@.
Additive	+, - Add, subtract	See multiplicative above
Relational	>, <, >=, <= Greater-than, less-than, greater/less-than-equal	These are most useful for conditionals: @amount>=0? "Credit" : "Debit"@ or @name=="this"? "that" : name@
Equality	==, != Equal, not-equal	See Relational above
Logical	AND &&	These operators make it possible to test multiple conditions: @revenue>100 && budget<50? "Winner!"@ or @name=="Jack" || name=="Sam"? "Good Name!"@.
Logical	OR ||	See and above
Conditional	? : If/then - with form "expr? true_expr : false_expr"	Provides IF/THEN/ELSE expressions. Note: a false expression is optional. 'null' will be evaluated to false and non-null as true. You can provide null substitutions like this: @name? name : "(None provided)"@. You can also nest conditionals for more conditions. For example, @age>=21?"Adult":(age>12?"Teen":"Child")@.
Assignments	=, +=	For the brave, you can create temporary variables for use in a report. Most of the functionality you might use this for is covered in more intuitive ways (such as the Running key), but it is possible to define a variable in a header row: @revTotal=0@ and update it in details rows @revTotal+=revenue@.

Math Functions

The following functions return floats.

Menu Item	Function
floor(float)	Round input down to the nearest whole number.
ceil(float)	Round input up to the nearest whole number.
round(float)	Round input to the nearest whole number.
abs(float)	Returns the absolute value of the input (if number < 0 return number * -1).

min(float, float)	Returns the input number with the least value.
max(float, float)	Returns the input number with the greatest value.
pow(float, float)	Returnsfirstnumber to the second number power.

String Functions

The following functions return strings.

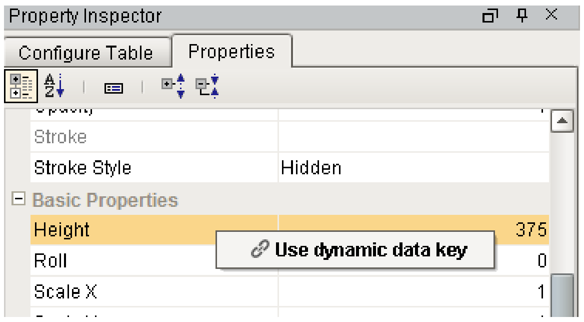
Menu Item	Function
startsWith(String, String)	Returns true if the first string starts with the second.
endsWith(String, String)	Returns true if the first string ends with the second.
substring(String, int start)	Returns a substring of String beginning at position start.
join(List aList, String aKeyChain, String aDelimeter)	Used to display an individual attribute of individual objects as a single String. Suppose you have a list of movies and want to show their titles in a comma separated list: @join(getMovies, "getTitle", ", ")@
substring(Object aString, int start, int end)	Obtain a subset of a given string. This could be useful if you wanted to restrict a text field to a certain number of chars:@substring(title, 0, 10)@

Next...

- Use Dynamic Data Key

Use Dynamic Data Key

It's possible to bind properties on Reporting components to parameters. It's as simple as dragging-and-dropping the parameter to the property, or right clicking the property and clicking on 'use dynamic data key'.



INDUCTIVE UNIVERSITY

Use Dynamic Data Key

Watch the Video

Next...

- Converting Legacy Reports
- Report Data

Converting Legacy Reports

The complete rebuild of the Reporting Module in Ignition 7.8 brought many improvements that would have been impossible to add to the legacy module. To preserve report functionality and prevent problems with backward compatibility, any existing reports will function as they always have. To get the most out of your reports and enable new functionality in such as Scheduled Reports, you'll have to convert your old Vision Report Panel components to new Report Resources. In an effort to minimize barriers, we created a Report Conversion process that will attempt to convert Ignition 7.7- reports into 7.8 + reports. We encourage people to convert their reports to new resources if they feel they would benefit from the added functionality, but in doing so, it's important to keep some things in mind:

Converting Legacy Reports

Watch the Video

Some Components have Imperfect Conversion

There are a number of components that have been upgraded or completely rebuilt. Due to the changes, some components and/or configurations may not convert perfectly to the new module. Specifically, the upgrade to the Barcode component and addition of 2D barcodes utilizes a new encoding system that does not have perfect parity with the legacy encoder. The new component does not explicitly support encoders for some of the Narrow and Extended codes, as well as MSI. Reports which require the Narrow or Extended Code 39/Codabar or MSI barcodes will not convert perfectly. Lastly, the changes to the Charts brought many improvements, but will look a little different and may require some configuration.

Data Sources will need Configuration

Data Sources are a huge improvement in the new module. It's now far easier to collect and use data from nearly anywhere in Ignition. Unfortunately, the custom properties in the legacy Report module do not directly map over to the new Data Sources. When a report is converted, its custom properties will be converted to Parameters. Parameters are great in that they allow for a quick conversion and enable things like Scheduled Reports, but Data Sources will need to be manually configured if desired.

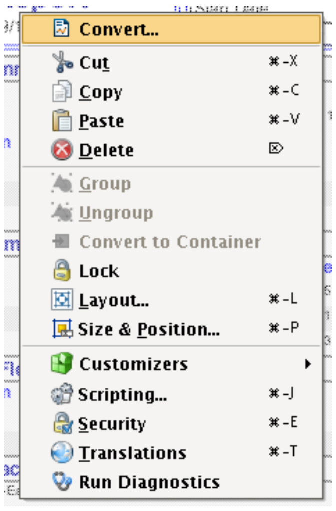
These are the two major caveats to be aware of when converting. The conversion tool has been tested with a variety of legacy reports, but there may be additional factors preventing conversion. We encourage users to visit the Inductive Automation forum and let us know – improvements to the conversion tool can only occur if we are made aware of errors!

Conversions are Non-Destructive

With the potential issues covered, it's important to note that report conversions do not destroy or alter the original report. If the conversion isn't successful for some reason, the new report can simply be deleted and the old one exists as it always did. If the conversion is successful, the old report can be saved to an unused window or exported as a backup before being deleted. Conversions may not always be perfect, but there is no risk in trying.

How to Convert

Converting a legacy report is quite simple.   In the Ignition Designer, open the window where the Report Viewer component exists. If you right click on the component, the new "Convert..." option is at the top of the popup menu. Selecting it will start the conversion process which first prompts for the "OK" to proceed, then asks what the new Report Resource should be called. Upon conversion, a new Report Viewer will be added to the existing window in addition to the old report. If you check the Project Browser, you'll see the new report under the Reports tree node. Open the newly created Report Resource for any edits, to add data sources, or add Scheduled Actions!



Next...

- Sample Legacy Report Conversion
- Report Data
- Parameters

Sample Legacy Report Conversion

Title

Sample Legacy Report Conversion

Watch the Video

Next...

Report Data

Data Keys

The most critical part of any report is data. Without data, there is nothing to report! In the Reporting Module, we use *Data Keys* and (optionally) *Keychain Expressions* to feed the report from the data sources. The simplest reference to data is a simple Data Key. A Data Key is a reference that points to the data it represents. At report generation time, these keys resolve to the values (or sets of values) provided by the data source.   In simple terms, Data Keys are placeholders for your data.

In this section ...

Data Keys and Keychain Expressions

Data Keys

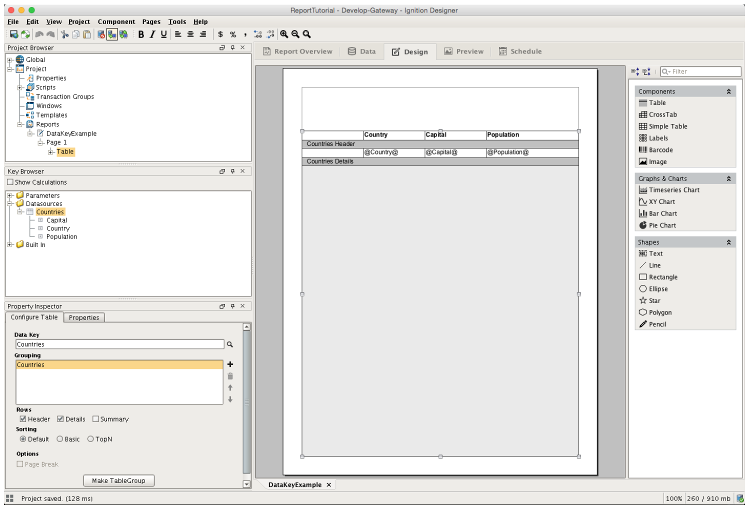
The most critical part of any report is data. Without data, there is nothing to report! In the Reporting Module, we use *Data Keys* and (optionally) *Keychain Expressions* to feed the report from the data sources. The simplest reference to data is a simple Data Key. A Data Key is a reference that points to the data it represents. At report generation time, these keys resolve to the values (or sets of values) provided by the data source.   In simple terms, Data Keys are placeholders for your data.

Example

If we created a data source named `Countries`, we can experiment with Data Keys to see how they resolve. Create a table in the Report Designer. Drag the Countries data source from the Key Browser to the Data Key field in the Table Configuration Panel as seen below. In the image, you can also see that we enabled the Table's Header, and typed header descriptions for each column we wanted to display. Then we dragged and dropped keys from the Key Browser into the Details columns. We can tell which items in the Report Designer reference data keys because they are surrounded by "@" symbols.

"Countries" data source

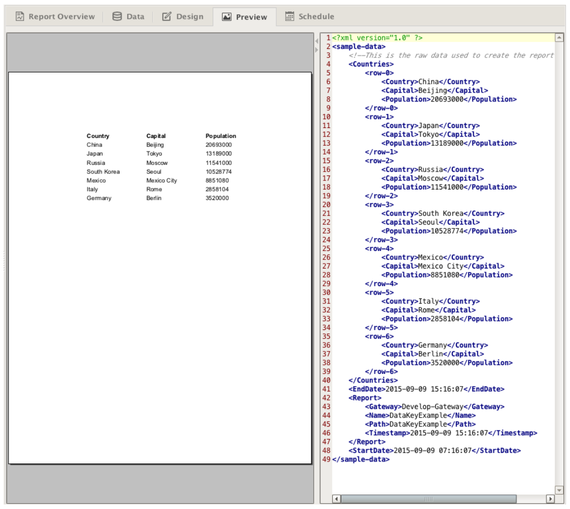
```
Country, Capital, Population
"China", "Beijing", 20693000
"Japan", "Tokyo", 13189000
"Russia", "Moscow", 11541000
"South Korea", "Seoul", 10528774
"Mexico", "Mexico City", 8851080
"Italy", "Rome", 2858104
"Germany", "Berlin", 3520000
```



	Country	Capital	Population
Countries Header			
	@Country@	@Capital@	@Population@
Countries Details			

Data Key: Countries
Grouping: Countries
Rows: [✓ Header] [✓ Details] ☐ Summary
Sorting: [✓ Default] Basic TopN
Options: ☐ Page Break

If we click to the Preview tab, you will see a simple table that neatly displays a row for each entry in Countries data source. Looking at the XML in this tab, you can see that the Countries data source sent a single row of data for each entry. You'll notice that the headers are printed exactly as they were in the Design panel. As an experiment, you can surround these header values with "@" symbols to see what happens.



Country	Capital	Population
China	Beijing	20693000
Japan	Tokyo	13189000
Russia	Moscow	11541000
South Korea	Seoul	10528774
Mexico	Mexico City	8851080
Italy	Rome	2858104
Germany	Berlin	3520000

```
<?xml version="1.0" ?>
<sample-data>
    <!--This is the raw data used to create the report
    <Countries>
        <row-0>
            <Country>China</Country>
            <Capital>Beijing</Capital>
            <Population>20693000</Population>
        </row-0>
        <row-1>
            <Country>Japan</Country>
            <Capital>Tokyo</Capital>
            <Population>13189000</Population>
        </row-1>
        <row-2>
            <Country>Russia</Country>
            <Capital>Moscow</Capital>
            <Population>11541000</Population>
        </row-2>
        <row-3>
            <Country>South Korea</Country>
            <Capital>Seoul</Capital>
            <Population>10528774</Population>
        </row-3>
        <row-4>
            <Country>Mexico</Country>
            <Capital>Mexico City</Capital>
            <Population>8851080</Population>
        </row-4>
        <row-5>
            <Country>Italy</Country>
            <Capital>Rome</Capital>
            <Population>2858104</Population>
        </row-5>
        <row-6>
            <Country>Germany</Country>
            <Capital>Berlin</Capital>
            <Population>3520000</Population>
        </row-6>
    </Countries>
    <EndDate>2015-09-09 15:16:07</EndDate>
    <Report>
        <Gateway>Develop-Gateway</Gateway>
        <Name>DataKeyExample</Name>
        <Path>DataKeyExample</Path>
        <Timestamp>2015-09-09 15:16:07</Timestamp>
    </Report>
    <StartDate>2015-09-09 07:16:07</StartDate>
</sample-data>
```

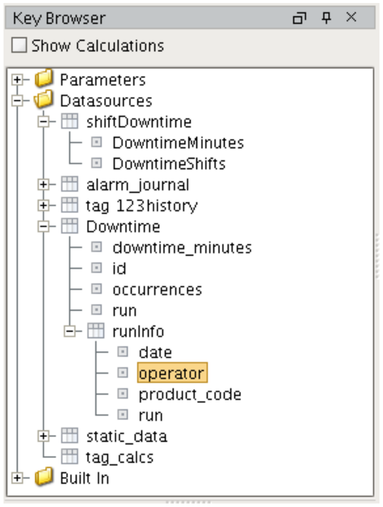
Data Key Usage

All text fields in Report Design elements are able to resolve Data Keys. In fact, you can freely mix plain text and data keys anywhere. In our table example above, we could substitute "@Population@" with "@Population@ people" and the data key portion would be swapped out for the appropriate data. This can be a very powerful feature when trying to create reports. For example, you could have a text field at the bottom of a report that displays "@Page@ of @PageMax@".   If for some reason your Data Key can't be found, your report will display a Substitution String (which defaults to "<N/A>" but can be configured in the Report's Property Table by selecting the Report in the Project Browser).

For more information on Data Key syntax, please see the Key Calculations page

Data Keys as Paths

Data Keys are *relative,* and use 'dot notation' to reference children.   Meaning, if we have a nested data structure, we can use Data Key paths (also known as *Keychains*) to reference the nested data. In the key browser image below, we have a nested data source called Downtime. Downtime contains a number of columns, and then contains a reference to additional data called `runInfo`. If we wanted to access the highlighted `operator` data, we could use the keychain dot notation in the designer - *@Downtime.runInfo.operator@* . Nested data sources are outside the scope of this page, but you can learn about Data Source nesting in the Nested Queries section.



Array Index of Data

You can reference an individual object in a list using standard array indexing syntax (brackets) like this: `@Data[0].firstName@`, where Data is a data source that contains a child data key named `firstName.` In this case, we would be accessing the first `firstName` in the set of data.

Next ...

- Parameters
- Static CSV
- Table Header, Details and Summary Rows

Parameters

Adding Parameters function similar to custom properties on Vision windows and components. Their value can be shown directly on your report, or be referenced by data sources and other parameters. Parameters are given a default value when first created, but this value can be overridden once the report runs.

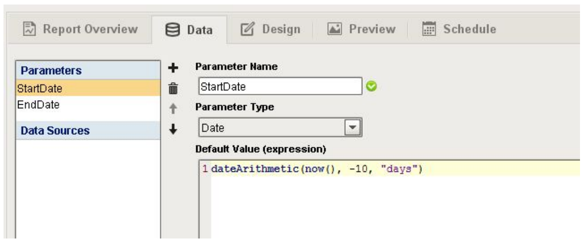
INDUCTIVE UNIVERSITY

Parameters

Watch the Video

Default Starting Parameters

When you first create a report, you start with two parameters: **StartDate** and **EndDate**. EndDate will have a default value of 'now()', while StartDate will have a default value of 'dateArithmetic(now(), -8, "hr")'. Notice how I have changed the default value of the StartDate to be a difference of 10 days instead of 8 hours by modifying the Default Value expression. You can also delete both StartDate and EndDate parameters if you do not need date parameters in your report.



Parameter Name: StartDate

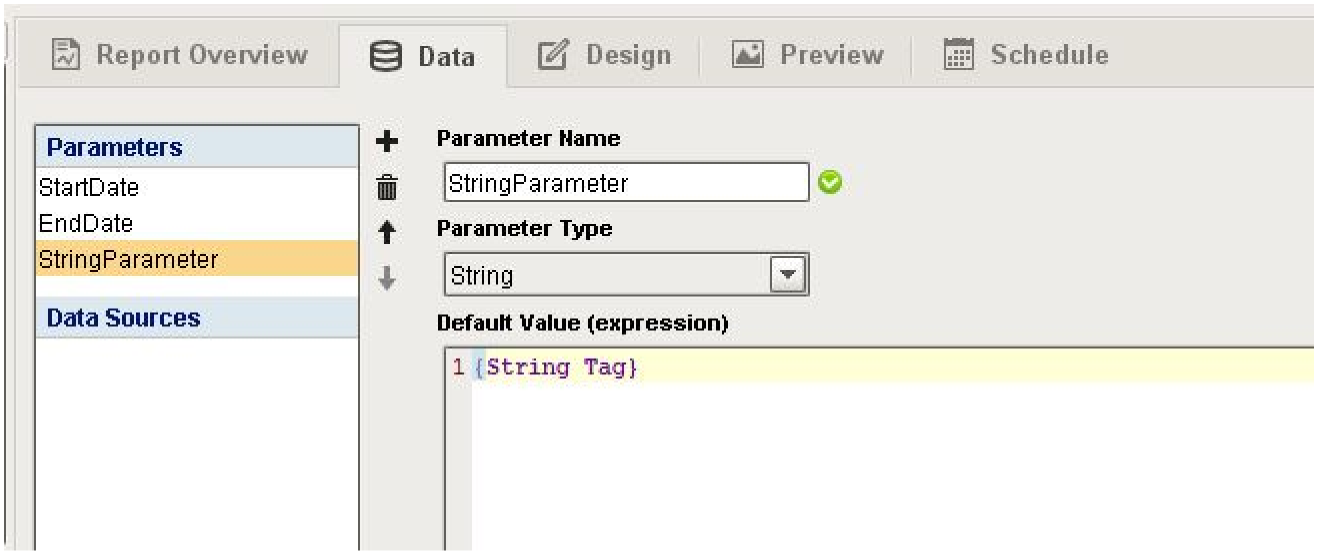
Parameter Type: Date

Default Value (expression)

```
dateArithmetic(now(), -10, "days")
```

Adding New Parameters

In addition to expressions, you can also have your default value reference a tag, like I have done with this StringParameter.



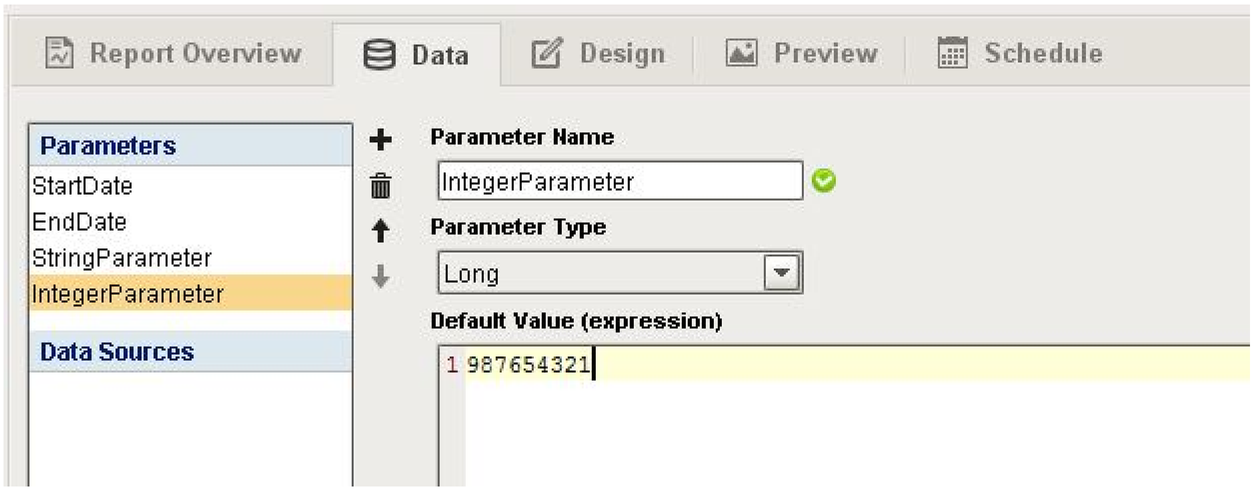
Parameter Name: StringParameter

Parameter Type: String

Default Value (expression)

```
{String Tag}
```

You can also use a literal value, like I have done for this IntegerParameter.



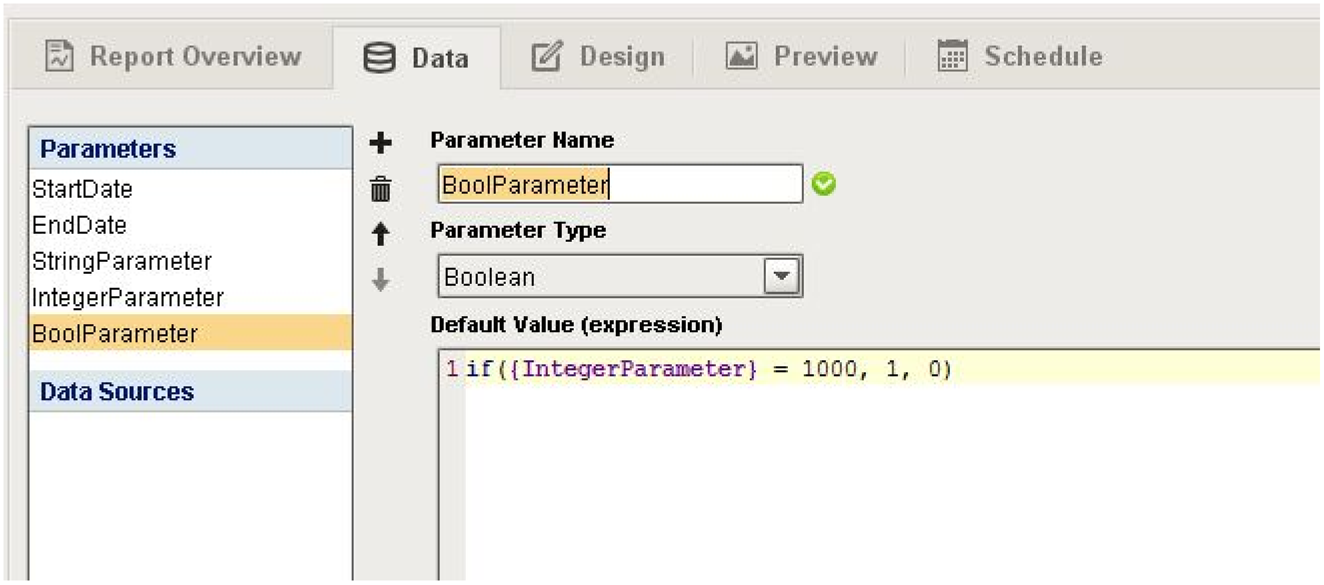
Parameter Name
IntegerParameter

Parameter Type
Long

Default Value (expression)

```
987654321
```

Finally, you can reference other parameters, like I have in this BoolParameter. It is important to note that when referencing other parameters, you must type the name of the parameter exactly within a set of curly braces { }. Also, you can only reference parameters that are above the current parameter. So while I can reference the IntegerParameter in the BoolParameter, I can not reference the Bool Parameter in the String Parameter.



Parameter Name
BoolParameter

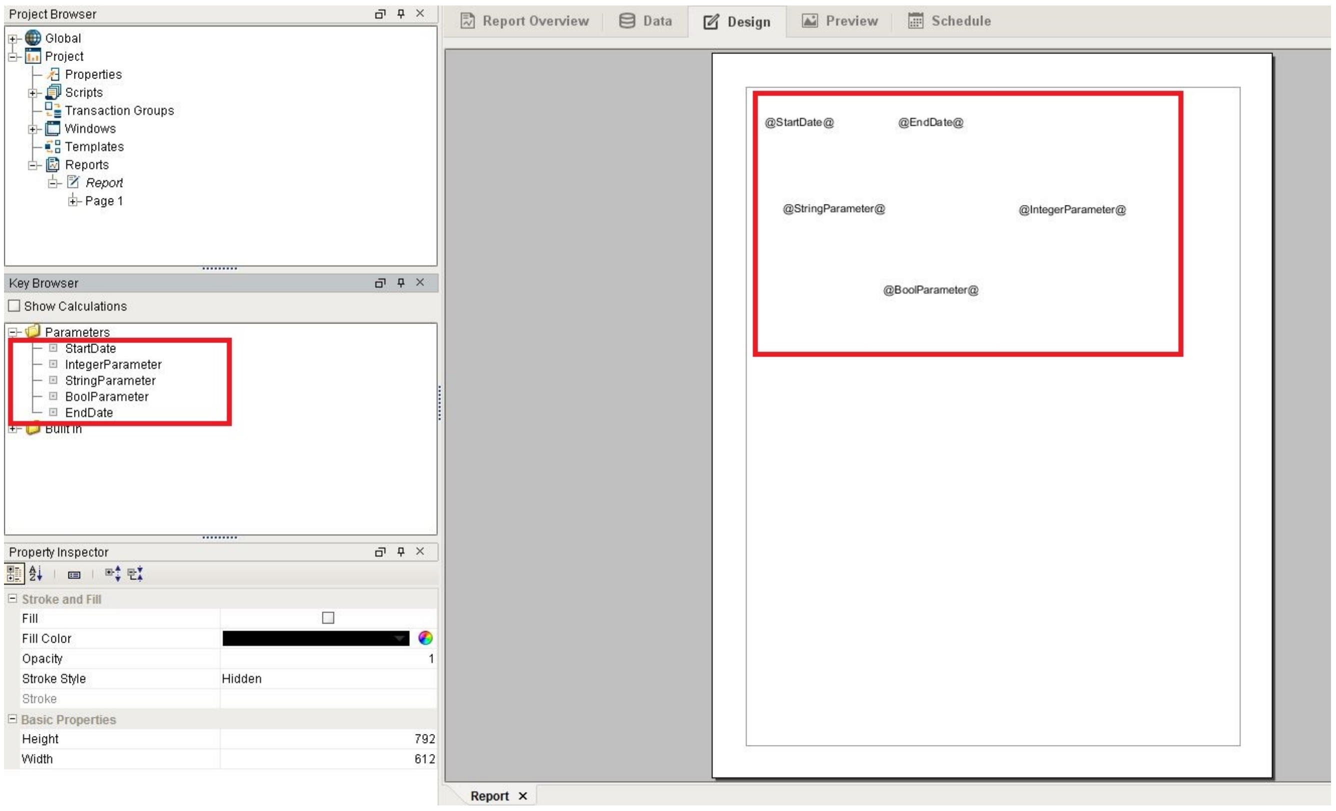
Parameter Type
Boolean

Default Value (expression)

```
if({IntegerParameter} = 1000, 1, 0)
```

Designing with Parameters

Now that we have created our parameters, we are ready to put together a report using those parameters. I have made a report and have dragged all of the parameters we created, including the start and end date onto the design section of the report. This way, we can see how all of our parameters behave within the report itself.


Project Browser
- Global
- Project
  - Properties
  - Scripts
  - Transaction Groups
  - Windows
  - Templates
  - Reports
    - Report
      - Page 1

Report Overview | Data | Design | Preview | Schedule

@StartDate@ @EndDate@

@StringParameter@ @IntegerParameter@

@BoolParameter@

Key Browser

Show Calculations

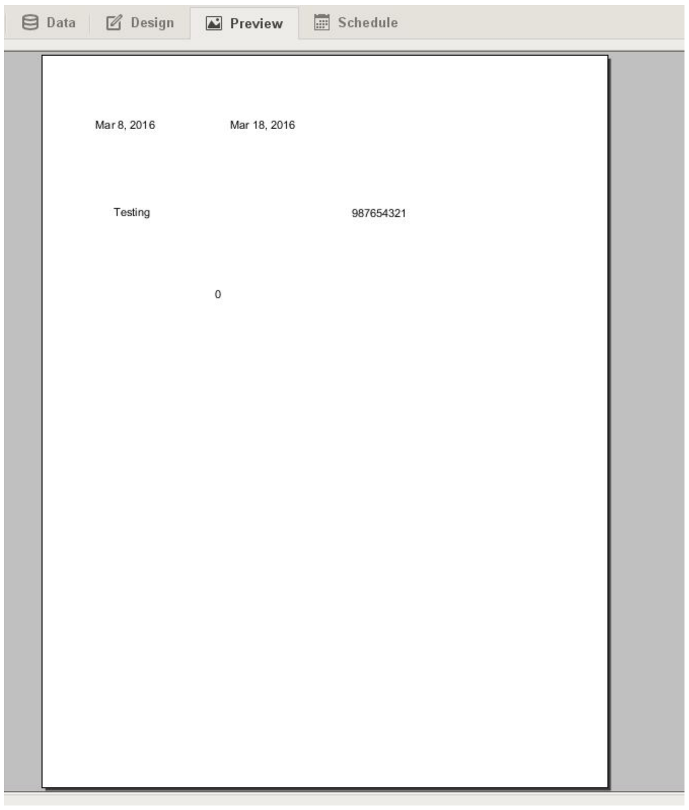
- Parameters
  - StartDate
  - IntegerParameter
  - StringParameter
  - BoolParameter
  - EndDate
- Built In

Property Inspector

Stroke and Fill	
Fill	☐
Fill Color	■
Opacity	1
Stroke Style	Hidden
Stroke	
Basic Properties	
Height	792
Width	612

Report ×

Switching over to the preview tab, we can see that each one of our parameters is now displaying its default value. The end date is showing March 18th, which was the current date at the time, and the start date is showing March 8th, which is 10 days before the current date. Our string parameter is displaying the word "Testing" which is the word that is stored in my string tag, the integer parameter is displaying the static number we placed in as a default value, and the boolean parameter has a value of 0 or false, because our integer parameter is not equal to 1000.


Data | Design | Preview | Schedule

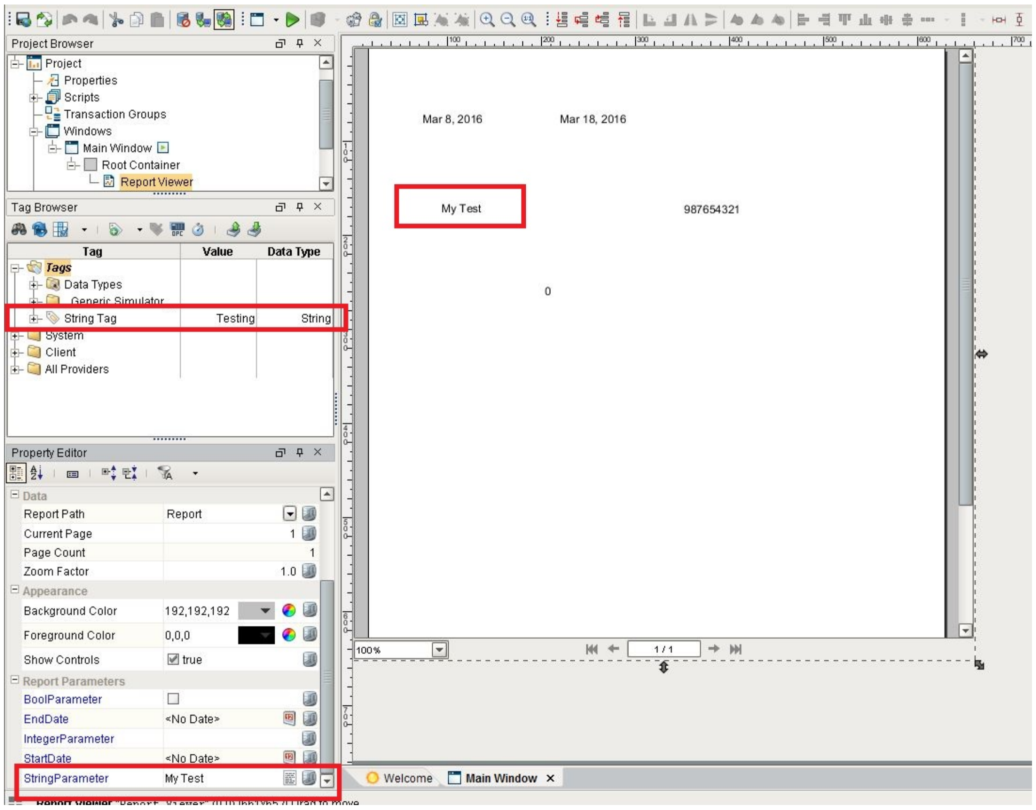
Mar 8, 2016 Mar 18, 2016

Testing 987654321

0

Manipulating the Parameters in the Report Viewer

When we view our report in the Report Viewer component on a vision window, we will at first see our default parameter values on the report. All of our parameters are listed in the property editor under the Report Parameters section. You will notice that I have manually entered in a value for the string parameter, and that parameter on our report updates to reflect the new value. You can also see that our string tag has not changed, nor will changing the string tag change our parameter. It is important to distinguish that setting the default value of a parameter is not the same as binding it. The default value will be used if there is no value in that parameter. Since I have placed a static value of "My Test into that string parameter, the report viewer on this page will always display "My Test" as the value of that parameter, regardless of what the value of the tag is.



Tag	Value	Data Type
Tags		
Data Types		
Generic Simulator		
String Tag	Testing	String
System		
Client		
All Providers		

Data	
Report Path	Report
Current Page	1
Page Count	1
Zoom Factor	1.0
Appearance	
Background Color	192,192,192
Foreground Color	0,0,0
Show Controls	☑ true
Report Parameters	
BoolParameter	☐
EndDate	<No Date>
IntegerParameter	
StartDate	<No Date>
StringParameter	My Test

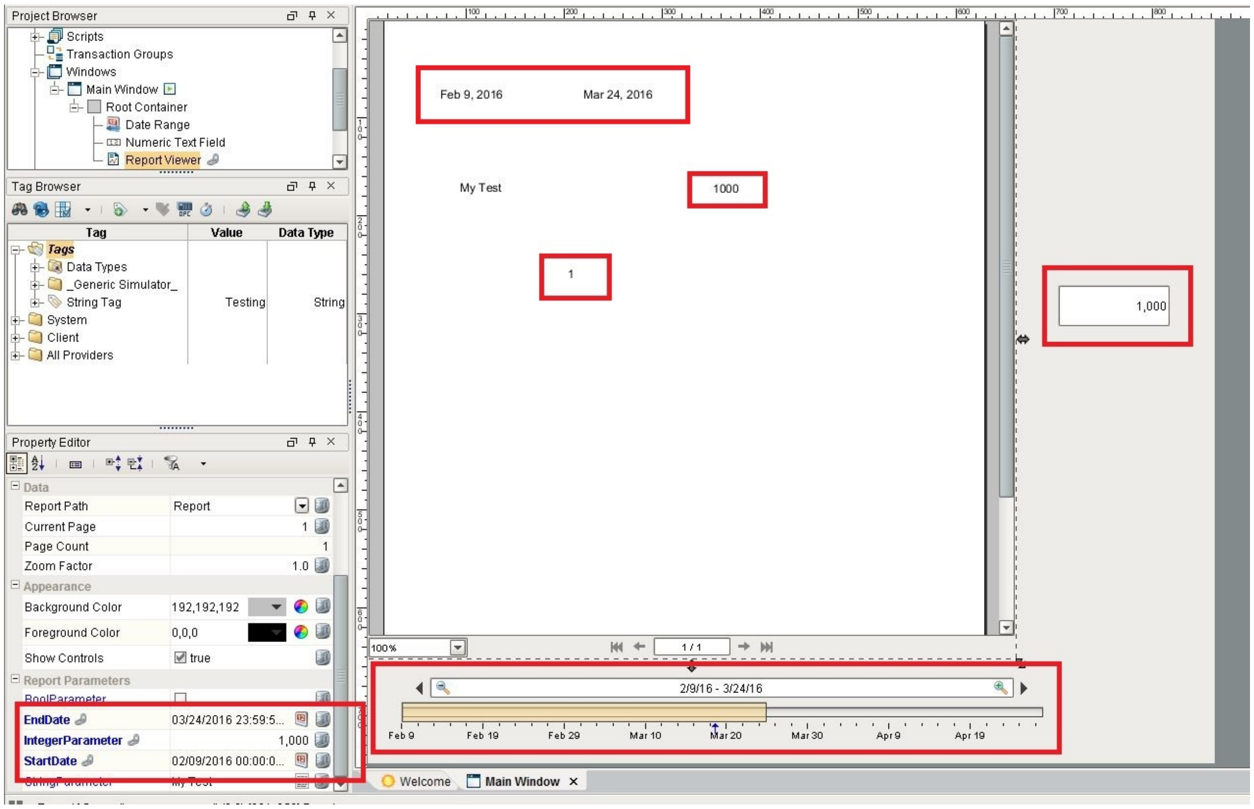
Mar 8, 2016 Mar 18, 2016

My Test 987654321

0

We can also bind values to the report parameters to make them dynamic. This allows us to use other on screen components to manipulate the reports parameters. I have bound the start date and end date parameters to the start and end date properties of a Date Range component. You can see that by moving the date range slider, we can easily manipulate the start and end date parameters of our report.

I have also bound the integer parameter to the numeric text field. You will notice that the value of the parameter reflects the value of the numeric text field. We can also see that because we have not put in a value for our bool parameter, it is still using the default value, which is true when the integer parameter is 1000. You can see that the bool parameter has updated in the report to reflect that.



Data	
Report Path	Report
Current Page	1
Page Count	1
Zoom Factor	1.0
Appearance	
Background Color	192,192,192
Foreground Color	0,0,0
Show Controls	☑ true
Report Parameters	
BoolParameter	☐
EndDate	03/24/2016 23:59:5...
IntegerParameter	1,000
StartDate	02/09/2016 00:00:0...
StringParameter	My Test

Feb 9, 2016 Mar 24, 2016

My Test 1000

1

1,000

100% 1 / 1

2/9/16 - 3/24/16

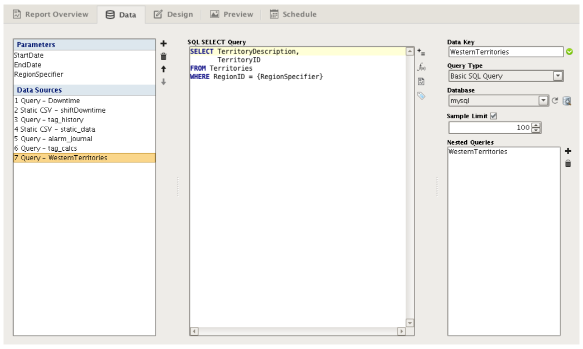
Feb 9 Feb 19 Feb 29 Mar 10 Mar 20 Mar 30 Apr 9 Apr 19

Next...

- Basic SQL Query

Basic SQL Query

This query type is the common type typically seen through much of Ignition before version 7.8. You can write queries which include tag path references, expressions, or report parameters which resolve at run time.



Parameters
StartDate
EndDate
RegionSpecifier

Data Sources
1 Query – Downtime
2 Static CSV – shiftDowntime
3 Query – tag_history
4 Static CSV – static_data
5 Query – alarm_journal
6 Query – tag_calcs
7 Query – WesternTerritories

SQL SELECT Query

```
SELECT TerritoryDescription,
       TerritoryID
FROM Territories
WHERE RegionID = {RegionSpecifier}
```

Data Key: WesternTerritories
Query Type: Basic SQL Query
Database: mysql
Sample Limit ✓: 100
Nested Queries: WesternTerritories

Basic SQL Query

Watch the Video

Next...

- SQL Query
- Report Tables
- Report Charts

SQL Query

The SQL Query Data Source allows you to craft parameterized queries that run as a prepared statement. In addition, it also features the new *Query Builder*, which is a powerful Drag-and-Drop query building GUI that allows you to make complex queries from your connected databases. As Prepared Statements, these queries are more resistant to SQL injection offering additional security over basic queries. Craft your query, and for each variable you wish to evaluate, substitute a '?' and a new editor will be added below in which you can add static values, expressions, or references.

INDUCTIVE UNIVERSITY

SQL Query

Watch the Video

Using the SQL Query Datasource

SQL Query Data Sources are created by selecting it from the + button menu on the Data tab of our Report Workspace. As with all Query data sources, at minimum you will want to verify the *Database* and give the Query a meaningful name in the right column's *Data Key* field.

Examples

Using Parameters in Queries

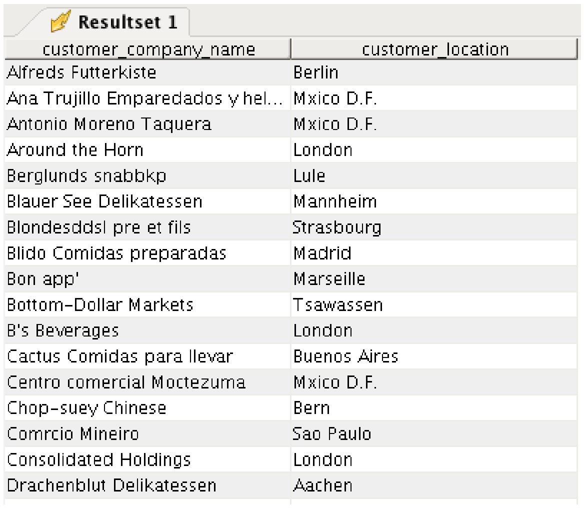
Parameterized Queries are an easy way to create Queries which can be safely configured to utilized dynamic values when a Report is executed in a schedule or Vision Report Viewer. When we write a Query of this type, we can substitute '?' anywhere we want to utilize a parameter value. As we add ? placeholders, the configuration panel will add editors for each placeholder. These editors allow you a ton of options. They can reference report Parameters, get values from Tag paths or utilize Ignition Expressions.

In this example, we'll use a simple parameter to alter the results of a Query at runtime in a Vision Report Viewer. Specifically, we are going to use Parameterized Query to alter the results displayed in a Table in the Vision component.

We'll be using a query that returns two columns of data:

1. The name of Customer
2. The city in which they are located.

The results from the base query look just like the image below.



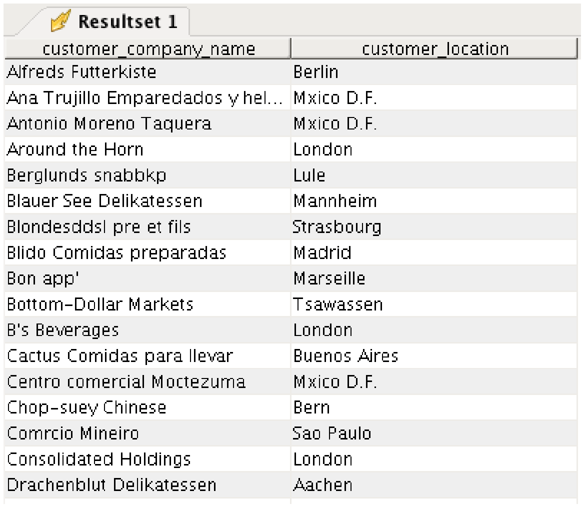
Resultset 1

customer_company_name	customer_location
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y hel...	Mxico D.F.
Antonio Moreno Taquera	Mxico D.F.
Around the Horn	London
Berglunds snabbkp	Lule
Blauer See Delikatessen	Mannheim
Blondesddsl pre et fils	Strasbourg
Blido Comidas preparadas	Madrid
Bon app'	Marseille
Bottom-Dollar Markets	Tsawassen
B's Beverages	London
Cactus Comidas para llevar	Buenos Aires
Centro comercial Moctezuma	Mxico D.F.
Chop-suey Chinese	Bern
Comrcio Mineiro	Sao Paulo
Consolidated Holdings	London
Drachenblut Delikatessen	Aachen

The first thing we will need is a Parameter for the report, so we start by creating a parameter through our Data panel and name it *CustomerLocation.* N ext, we'll create the actual SQL Query datasource and add our query.

```
SELECT
        CompanyName AS customer_company_name, City AS customer_location FROM Customers
WHERE
        City = ?
```

Now in the *Parameter 1* editor below the query box, we want to add a reference to our Parameter, which we can do using curly braces like `{Customer Location}`. Here is what the finished data source looks like.

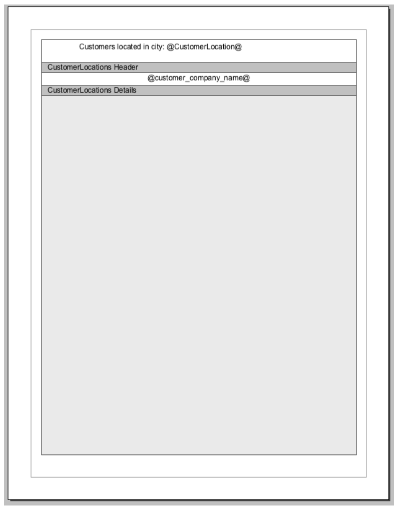


Resultset 1

customer_company_name	customer_location
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y hel...	Mxico D.F.
Antonio Moreno Taquera	Mxico D.F.
Around the Horn	London
Berglunds snabbkp	Lule
Blauer See Delikatessen	Mannheim
Blondesddsl pre et fils	Strasbourg
Blido Comidas preparadas	Madrid
Bon app'	Marseille
Bottom-Dollar Markets	Tsawassen
B's Beverages	London
Cactus Comidas para llevar	Buenos Aires
Centro comercial Moctezuma	Mxico D.F.
Chop-suey Chinese	Bern
Comrcio Mineiro	Sao Paulo
Consolidated Holdings	London
Drachenblut Delikatessen	Aachen

When the Report executes, it is going to substitute the value of the ? for whatever value is used for *CustomerLocation.* Remember that parameter values can be entered in the Parameter configuration of the Report Data panel, as a property in the Vision Report Viewer component, or in the Report Schedule panel, or any mixture of the three. If a value is not selected as a default in the Report Data panel, one will need to be specified in the Vision component to generate a report view in the Viewer, or in the Report Schedule for successful generation via schedule Action. If a default is provided, any values entered in those respective areas with override the default. In our case, we intend to use the Report only in a Vision component, so we will forgo the default and let the operator enter a value to see the result.

Now we'll create a simple table in the design panel by dragging the table component onto the page and dropping our CustomerLocations data source key into the DataKey config field. We'll set Header and a Details rows to visible and unstructured and add and then add some simple text. The header will let us know which city the parameter is set to, and we will simple print a line for each customer name we find that match the city in the details row. If we click over to the Preview panel now, we'll find the Report blank. This is expected since we have no default parameter and have yet to specify one.



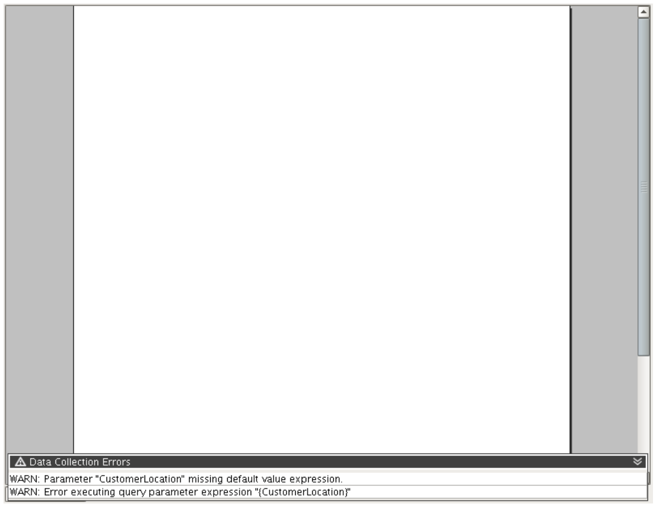
Customers located in city: @CustomerLocation@

CustomerLocations Header

@customer_company_name@

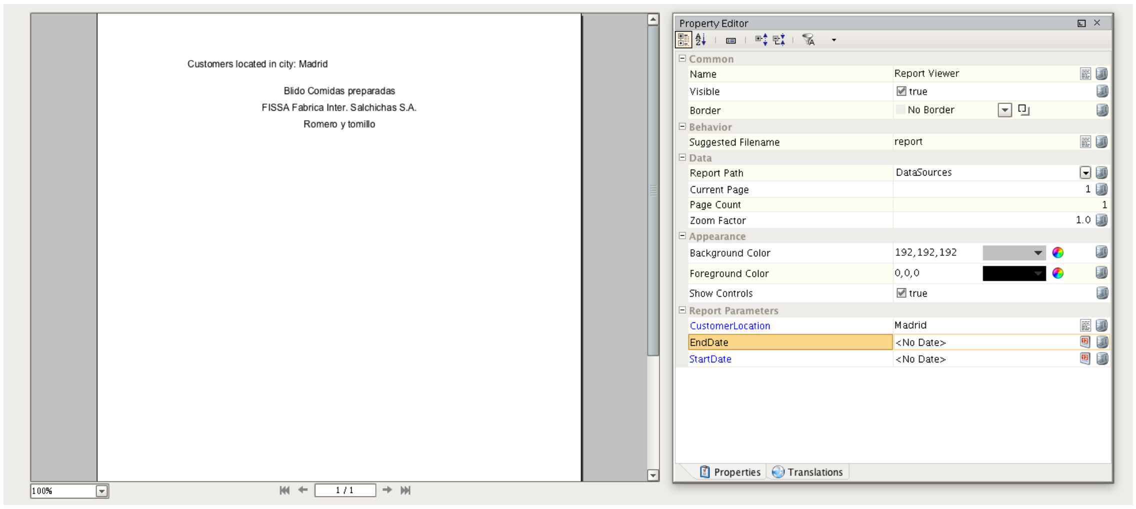
CustomerLocations Details

Now we shift to the Vision workspace to create a new Window and add a Report Viewer component. If we select the Report we created in the Property Inspector of the Report Viewer component, we should see a similar Data Collection Error to the one in the Reporting Workspace Preview panel.


Data Collection Errors

WARN: Parameter "CustomerLocation" missing default value expression.
WARN: Error executing query parameter expression "{CustomerLocation}"

To correct the error, we simple need to add a value to the parameter field for CustomerLocation. We know there are some Customers in Madrid, so we try that and we get the following result.


Customers located in city: Madrid

Blido Comidas preparadas
FISSA Fabrica Inter. Salchichas S.A.
Romero y tomillo

Property Editor

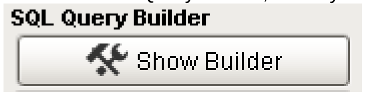
Common	
Name	Report Viewer
Visible	☑ true
Border	No Border
Behavior	
Suggested Filename	report
Data	
Report Path	DataSources
Current Page	1
Page Count	1
Zoom Factor	1.0
Appearance	
Background Color	192,192,192
Foreground Color	0,0,0
Show Controls	☑ true
Report Parameters	
CustomerLocation	Madrid
EndDate	<No Date>
StartDate	<No Date>

These parameter properties and bindable, so you can use text fields, tag values, or any other Ignition binding to set the value. By using parameters in this way, we allow a lot of flexibility while at the same time significantly reducing the risk of SQL injection. This is a simple example, but using this same process you can create complex reports in a similarly simple process!

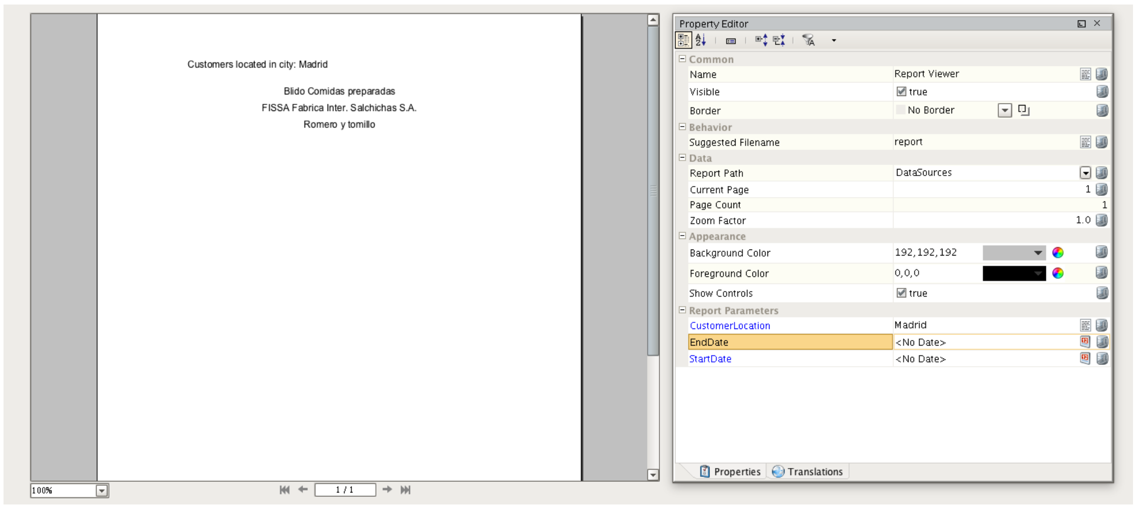
Crafting Queries with the Query Builder

The SQL Query data source includes our first release of the powerful SQL *Query Builder* tool. The Query Builder is a graphic tool for building queries. While a basic understanding of SQL helps make the most of Query Builder powerful tool, most people will have no problem creating effective queries after a brief tutorial.

To activate the Query Builder, start by selecting the SQL Syntax version from the drop down menu beneath the



button. If your Database type isn't available (or you aren't sure), you can likely get most of the general functionality selecting the *Universal* option. Then push the button to show the *Query Builder.* The Query builder has a lot of functionality and can consume a lot of space in the Designer. As a result, we recommend using it on larger monitors if possible.



Customers located in city: Madrid

Bldo Comidas preparadas
FISSA Fabrica Inter. Salchichas S.A.
Romero y tomillo

Common	
Name	Report Viewer
Visible	✓ true
Border	No Border
Behavior	
Suggested Filename	report
Data	
Report Path	DataSources
Current Page	1
Page Count	1
Zoom Factor	1.0
Appearance	
Background Color	192,192,192
Foreground Color	0,0,0
Show Controls	✓ true
Report Parameters	
CustomerLocation	Madrid
EndDate	<No Date>
StartDate	<No Date>

Next...

- Tag Historian Query
- Parameters

Tag Historian Query

The Tag Historian Query provides a simple way to query data from Tag Historians. Utilizing a user interface familiar to Tag Historian users, you can collect data from Historical tags for specific date ranges, using aggregations, or for fixed sample sizes.

Tag Historian Query

Watch the Video

Next...

- Tag Calculation Query
- Report Charts

Tag Calculation Query

Tag Calculations are executed by the Tag Historian, providing multiple calculated values for each tag path. Calculations such as *Sum, Duration On /Off, Average* and more are available.

Tag Calculation Query

Watch the Video

Next...

- Alarm Journal Query
- SQL Bridge Module

Alarm Journal Query

The Alarm Journal Query data source is a simple way to access Alarming data within a report. It provides a straight-forward interface to select StartDate and EndDate, Event types, Filters and Priority.

Alarm Journal Query

Watch the Video

Next...

- Static CSV
- Report Tables

Static CSV

The Static CSV data source allows you to quickly craft a data source to use in your report. This data source is ideal when you need some test data to begin creating a new report.

Static CSV

Watch the Video

Example

1) In the Data Panel of a report, click on the ✚ and select Static CSV

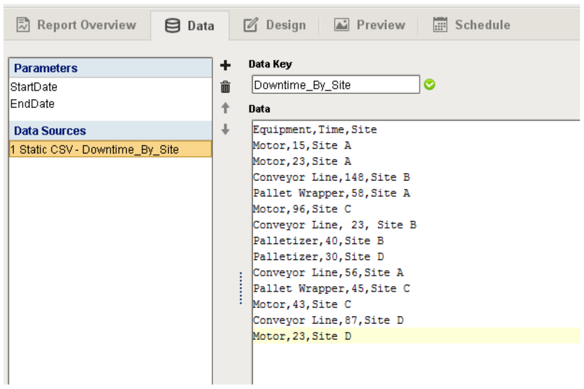
2) Give the data source a name by filling in the Data Key property with the following: Downtime_By_Site

3) Copy the following data and paste it into the Data field:

Downtime_By_Site

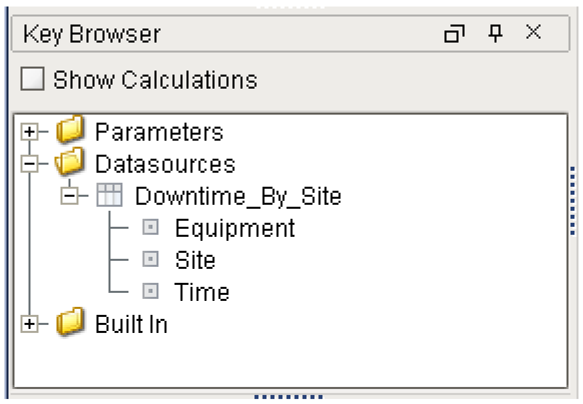
```
Equipment,Time,Site
Motor,15,Site A
Motor,23,Site A
Conveyor Line,148,Site B
Pallet Wrapper,58,Site A
Motor,96,Site C
Conveyor Line, 23, Site B
Palletizer,40,Site B
Palletizer,30,Site D
Conveyor Line,56,Site A
Pallet Wrapper,45,Site C
Motor,43,Site C
Conveyor Line,87,Site D
Motor,23,Site D
```

4) The data source should look like the picture below:



```
Equipment,Time,Site
Motor,15,Site A
Motor,23,Site A
Conveyor Line,148,Site B
Pallet Wrapper,58,Site A
Motor,96,Site C
Conveyor Line, 23, Site B
Palletizer,40,Site B
Palletizer,30,Site D
Conveyor Line,56,Site A
Pallet Wrapper,45,Site C
Motor,43,Site C
Conveyor Line,87,Site D
Motor,23,Site D
```

5) Switch to the Design panel, expand Datasources in the Key Browser, and you'll see the Static CSV available for use



Next...

- Nested Queries

Nested Queries

Query Nesting turns complex data relationships into easily reportable structures and will undoubtedly be one of the most useful Reporting v3 features for. So what is a Nested Data Source? The simple definition is that a *Nested Query* uses the results of a previously executed query to collect data. The general structure of a Nested Data Source is one in which you have a Parent queries and child queries. Those well-versed in SQL are probably thinking that this sounds like a `JOIN` and in fact, there are some similarities. There are also some major differences which allow Nested Data to be both easier and more powerful to accomplish:

1. Nesting relationships are not restricted to data in a single schema, database or even source! Nesting is easy to configure across tables, between different databases, or even with sources like the Tag Historian!
2. Writing queries for nested query sources can be far simpler and easier to maintain than writing complex JOIN operations
3. Nested structures allow more control in how data is collected, allowing data structures and relationships that are more expressive

Nested Queries

Watch the Video

How Nesting Works

Let's use a simple data relationship to help illustrate how nesting occurs.

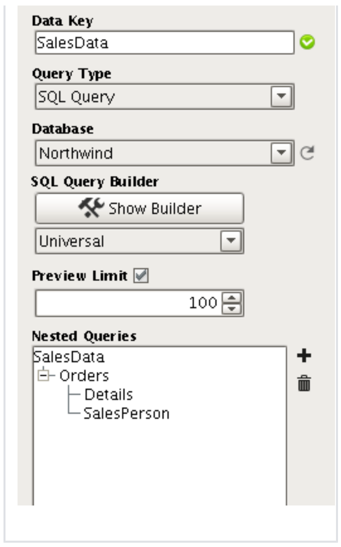
Imagine we have data collected from two unrelated sources that look the ones seen in this table.

Codes		Frequency		
CodePK	**Code**	**FrequencyID**	**CodeID**	**Frequency**
1	ZG	1	6	11
2	GB	2	7	41
3	DC	3	5	13
4	PD	4	4	26
5	JE	5	3	13
6	FO	6	2	32
7	QT	7	1	11
...	...	...	...	...
5436	POE	3456	5345	34

We want to create a data source connecting all these things for reporting using nesting. The process to accomplish this is:

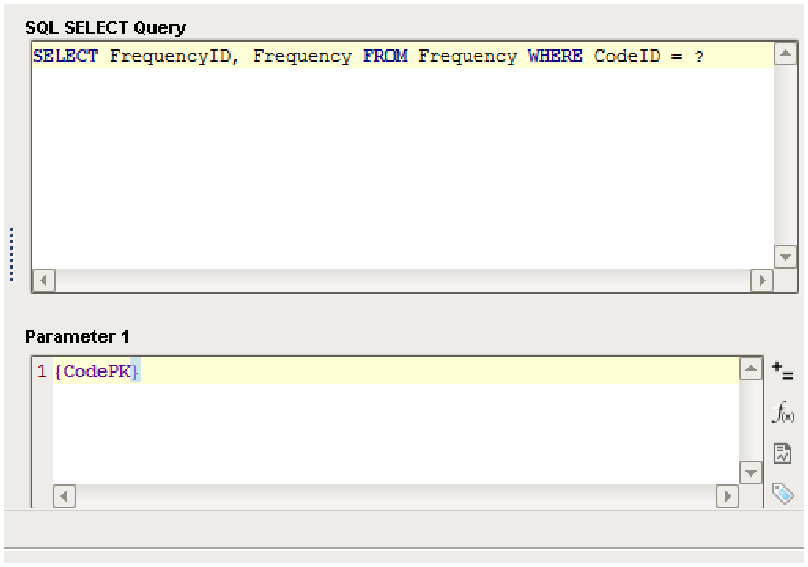
1. Create a new data source to be the **Parent** query. The parent query will likely depend on what your data looks like. For this example, we'll just use the `Codes` data.
2. Give your data source a meaningful name, `CodeFrequency` will work for us.
3. Create a **Child Query** by clicking the + button in right configuration pane of your data source, also giving it a name. We will call this nested data `NEST`.
4. In the Child query, reference one or more column values from the parent.

Nest Query Configuration


Data Key
SalesData
Query Type
SQL Query
Database
Northwind
SQL Query Builder
Show Builder
Universal
Preview Limit ✓
100
Nested Queries
```
SalesData
└ Orders
    ├ Details
    └ SalesPerson
```

Example of what these queries could look like:

```
-- Parent Query

SELECT * FROM Codes

-- In the Child Query, accessed by clicking on the NEST leaf of the Nested Queries Tree
-- This assumes the value of Parameter 1 would be equal to {CodePK}
SELECT FrequencyID, Frequency FROM Frequency WHERE CodeID = ?
```


SQL SELECT Query
```
SELECT FrequencyID, Frequency FROM Frequency WHERE CodeID = ?
```
Parameter 1
```
{CodePK}
```

How Nesting Works

With this simple example, what happens is that first the parent query is called, and a set of results is returned to the reporting system with the name of our parent `CodeFrequency` . Only after this data has returned, the Child query will execute, once for each row of the parent, substituting the value of `CodePK` into the child query where we have the `{CodePK}` reference.

The resulting first two rows of data will have a structure like this:

```
CodeFrequency
           |
           |-Row 1
           |     |_ CodePK - 1
           |     |_ Code   - ZG
           |     |_ NEST
```

```
            |            |_ FrequencyID - 1
            |            |_ Frequency   - 11
            |
            |-Row 2
            |     |_ CodePK - 2
            |     |_ Code   - GB
            |     |_ NEST
            |            |_ FrequencyID - 6
            |            |_ Frequency   - 32
```

Each row returns the CodePK and Code from the parent query, but also the results of the child query, that apply to the `CodeID` we get from the parent PK.

It may be easier to see the data returned in a flatter form like a table that relates the values by row, ignoring the tree structure:

Parent Query Row	CodePK	Code	FrequencyID	Frequency
1	1	ZG	1	11
2	2	GB	6	32

Now, in our Report Design, we will have access to our datasource `FrequencyData` that has linked these two sets of data together through a shared value.

Special Considerations

Nested Queries are powerful and easy to use, but users should be aware of runtime implications. Imagine the scenario above, where we have two sets of data, each with 5000+ rows. When our child query executes, each row of its query is going to require a lookup from the parent. For most common sets of data and database sizes, this won't be an issue, but it's possible to imagine that instead of just one child query, we have a dozen. In addition, some of those children *also* have many children.   It's very easy to see in this scenario how exponential growth occurs and our system performance may suffer. Most report designers will limit query sizing as oversized data structures are simply not as easy to work with. However, if you feel an urge to generate massive complex trees of million line queries, you may be waiting a while.

Next...

- Table Grouping - Child Tables
- Charts Inside of Tables

Report Charts

Report charts allow for the visual representation of reportable data. Charts can be driven by any data source provided to a report.

In this section ...

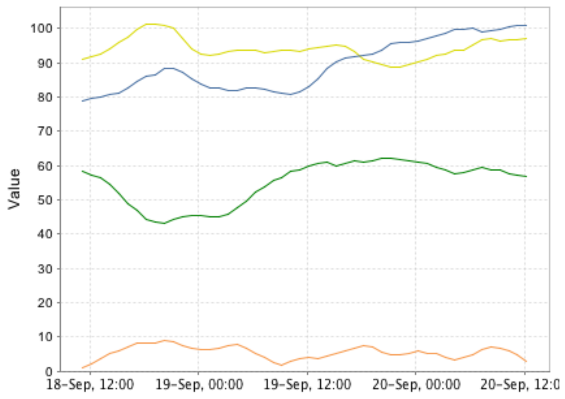
Timeseries Chart

The Timeseries Chart is a type of XY chart whose domain or X-Axis represents time series data and range can be one or more pens. The Timeseries Chart is a great way to display visual from Tag History, or similar time related data sources.

INDUCTIVE UNIVERSITY

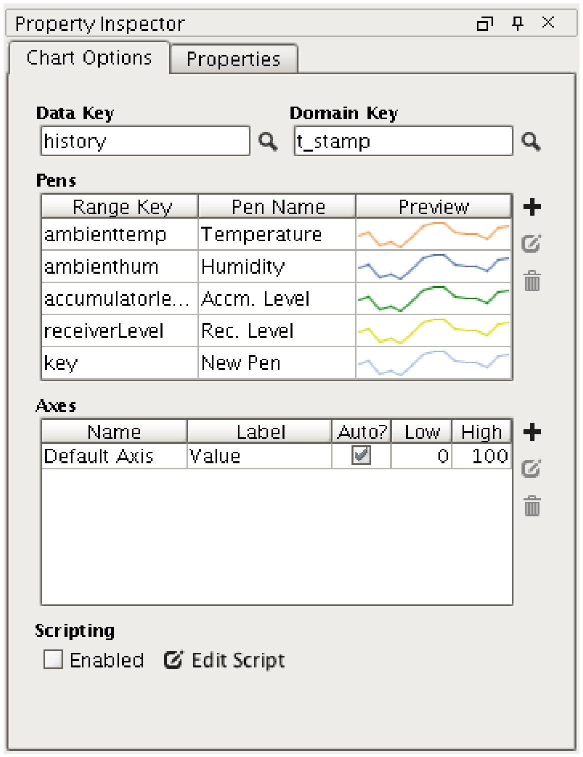
Timeseries Chart

Watch the Video



Usage

To use the chart, drag the component from the Report Palette to your report. Type or drag a data key from the key browser into the **Data Key** field of the Chart Options Config. Select a time series domain (for example a `t_series` column of your query). The Y axis (value) can be modified in the property inspector but the X axis (time range) is based on the data in the 'data key' that powers the chart dataset.



Data Key: history

Domain Key: t_stamp

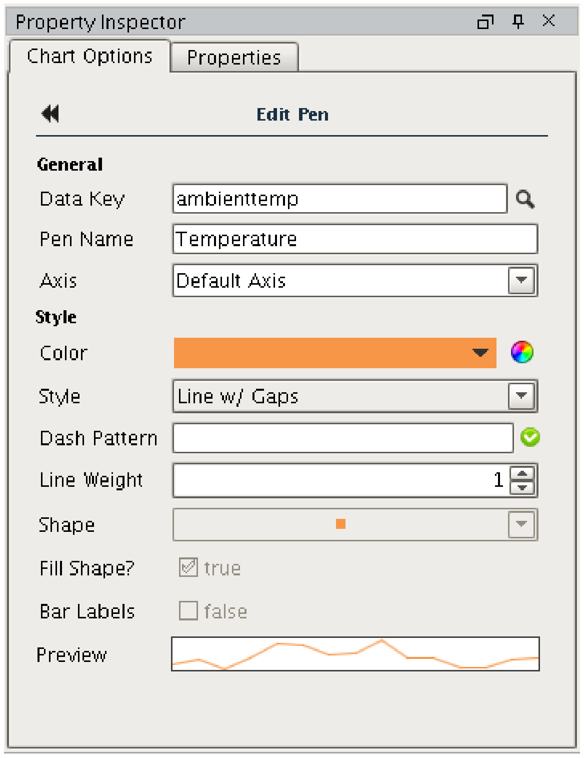
Pens

Range Key	Pen Name	Preview
ambienttemp	Temperature	
ambienthum	Humidity	
accumulatorle...	Accm. Level	
receiverLevel	Rec. Level	
key	New Pen	

Axes

Name	Label	Auto?	Low	High
Default Axis	Value	✓	0	100

To add pens to your query, simply click the ✚ button on the Configure Chart panel of the property inspector. Double clicking or selecting a pen then clicking the button will toggle the Pen Configuration area.



Property Inspector

Chart Options | Properties

Edit Pen

General

Data Key	ambienttemp
Pen Name	Temperature
Axis	Default Axis

Style

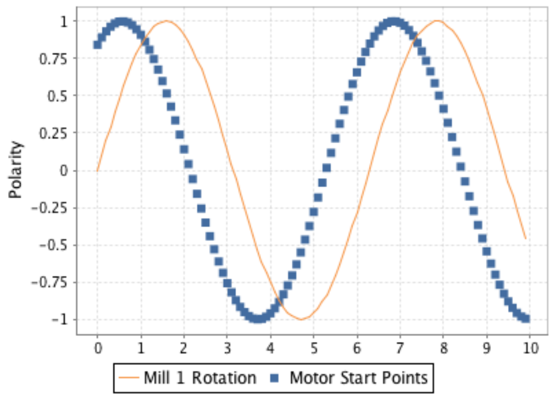
Color	🟧
Style	Line w/ Gaps
Dash Pattern	
Line Weight	1
Shape	■
Fill Shape?	☑ true
Bar Labels	☐ false
Preview	

Next...

- XY Chart

XY Chart

The XY Chart is will generate an X vs Y plot of your data. XY charts can have multiple Pens and Axes per data source, each easily configurable in the custom configuration panel for the component.



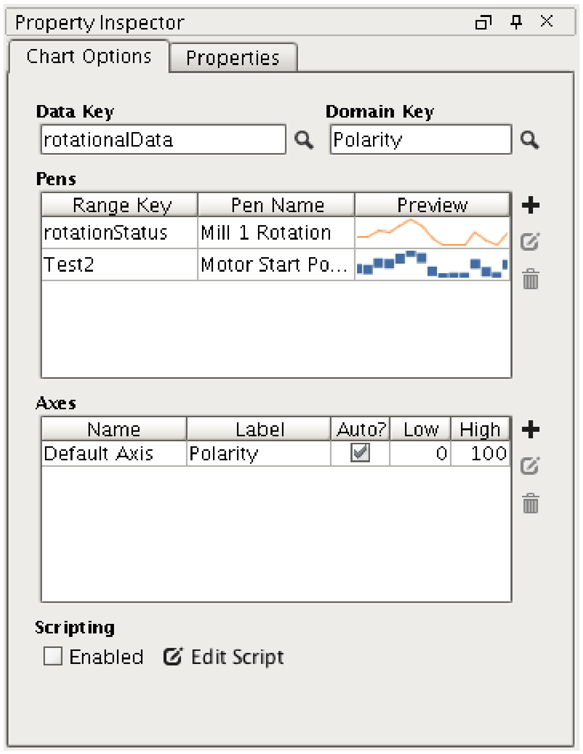
INDUCTIVE UNIVERSITY

XY Chart

Watch the Video

Usage

To use the chart, drag the component from the Report Palette to your report. Type or drag a data key from the key browser into the **Data Key** field of the Chart Options Config. Select a time series domain (for example a `t_series` column of your query).



Data Key: rotationalData
Domain Key: Polarity

Pens

Range Key	Pen Name	Preview
rotationStatus	Mill 1 Rotation	
Test2	Motor Start Po...	

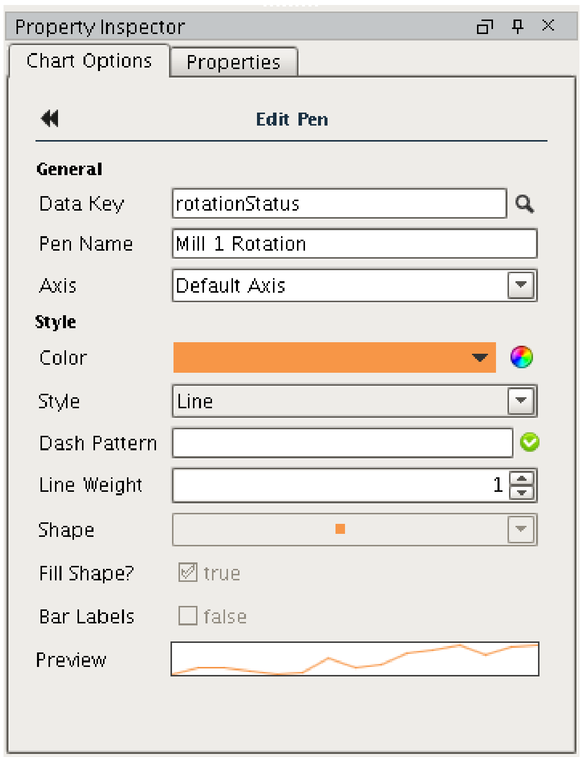
Axes

Name	Label	Auto?	Low	High
Default Axis	Polarity	✓	0	100

Scripting

☐ Enabled Edit Script

To add pens to your query, simply click the ✚ button on the Configure Chart panel of the property inspector. Double clicking or selecting a pen then clicking the button will toggle the Pen Configuration area.



General

Data Key: rotationStatus
Pen Name: Mill 1 Rotation
Axis: Default Axis

Style

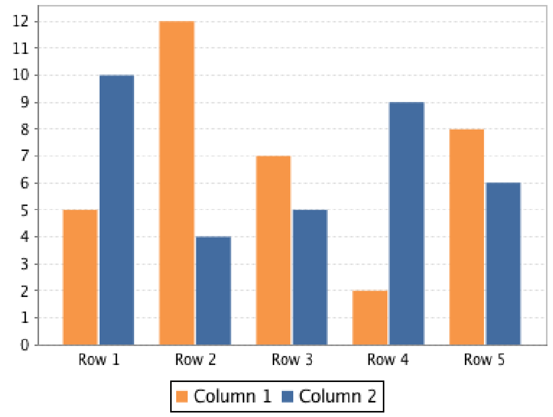
Color
Style: Line
Dash Pattern
Line Weight: 1
Shape
Fill Shape?: true
Bar Labels: false
Preview

Next...

- Report Bar Chart

Report Bar Chart

The Bar Chart component can be used to add bar charts to a report.



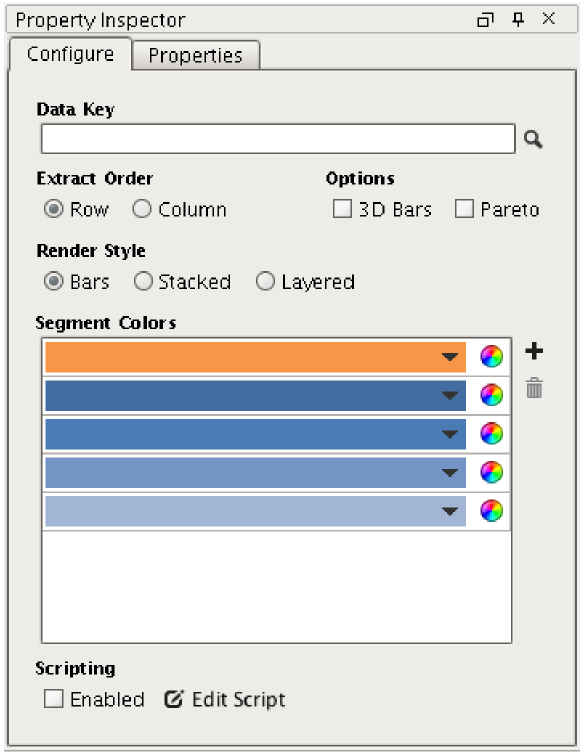
INDUCTIVE UNIVERSITY

Bar Chart

Watch the Video

Usage

To create a bar chart, simply drag the button from the Report Palette and drop onto your page, Table Row, or other shape capable of containing children. Bar charts are quite easy to configure, requiring a data source whose first column generally contains the name or identifier the bar represents, and the following one or more columns represent some numerical value to be plotted.



Property Inspector

Configure | Properties

Data Key

Extract Order: [✓ Row] Column

Options: 3D Bars, Pareto

Render Style: [✓ Bars] Stacked Layered

Segment Colors

Scripting: Enabled, Edit Script

Examples

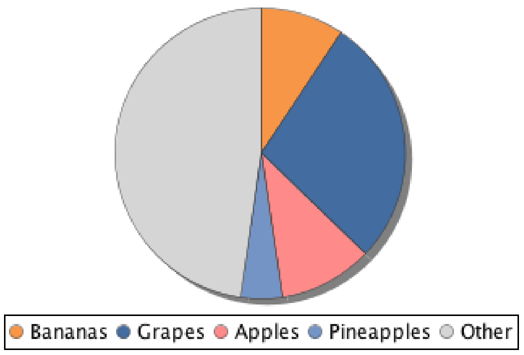
A Bar Chart example can be found in the Report Workflow Tutorial.

Next...

- Report Pie Chart

Report Pie Chart

The Pie Chart adds a Pie Chart based on a simple two column data structure.



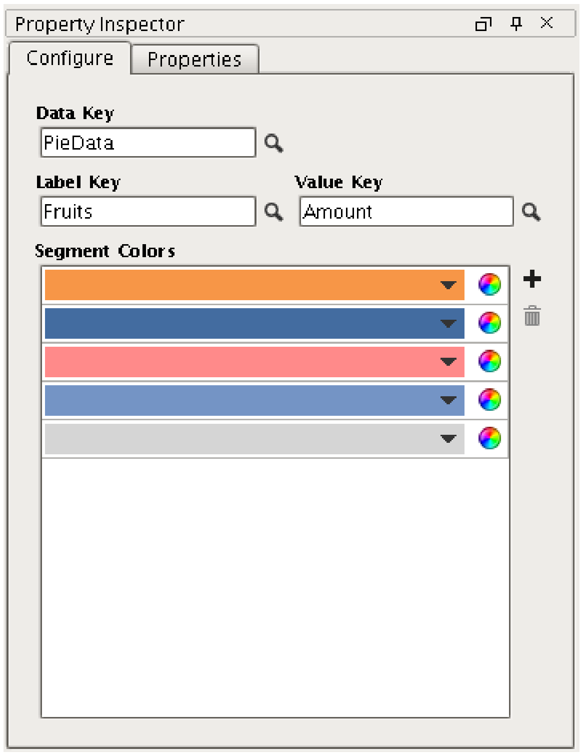
INDUCTIVE UNIVERSITY

Pie Chart

Watch the Video

Usage

To create a new Pie Chart, drag the pie chart palette item from the Report Design Palette onto your Report Page or Parent Shape. A pie chart has a simple configuration consisting of label values (generally Strings) and numeric quantity values. Color segments can be added or removed by clicking the ✚ button on the Configuration panel.


Property Inspector

Configure | Properties

Data Key: PieData

Label Key: Fruits

Value Key: Amount

Segment Colors

Example

The Pie chart in the images above was created using a simple data source which had just two columns, the first which represented our Label, the second being a column of values.

```
Fruits, Amount
"Bananas", 52
"Grapes", 154
"Apples", 58
"Pineapples", 25
"Other", 265
```

While this example may seem simple, any data source containing values can be used in similar way by simply using two relevant data keys from a more complex source.

Next...

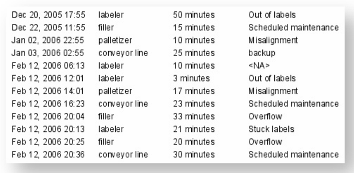
- Report Tables

Report Tables

Tables are a major part of Ignition Reporting. They are composable and highly flexible, giving Reporting users the ability to flexibly layout and organize tabular data.

Overview

Basic Table Example



Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Dec 22, 2005 11:55	filler	15 minutes	Scheduled maintenance
Jan 02, 2006 22:55	palletizer	10 minutes	Misalignment
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 06:13	labeler	10 minutes	<NA>
Feb 12, 2006 12:01	labeler	3 minutes	Out of labels
Feb 12, 2006 14:01	palletizer	17 minutes	Misalignment
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels
Feb 12, 2006 20:25	filler	20 minutes	Overflow
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance

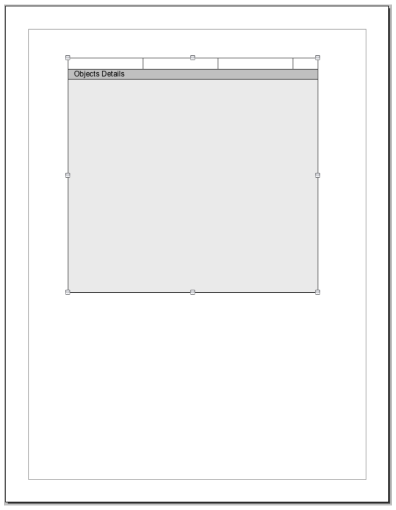
Tables are objects that display data in a structured, repetitious format. Their complexity can range from trivially simple to complicated, but fundamentally they do the same thing - take the data given to them and repeat structured row for each row or set of data provided. The Reporting engine will automatically create new pages to fit all data within the table's boundaries while applying your formatting preferences (such as font, size, layout and alignment) each step along the way. Combine that feature with powerful data manipulation and expressive layout tools, and you get an object that often forms the basis of your reports.

We'll start by taking a look at the various parts of the base table component and move into more features and example.

The Table Component

The Table component can be created by dragging it from the Report Design Palette, or by simply dragging a Data Source from the Key Browser to a page in your report. A new Table with no bound Data Key is fairly simple looking, but hides a wealth of functionality. To best understand how to use a table, you'll want to have some familiarity with the Report Design Concepts (Version 7.7) we cover in the design introduction. Selection and Super Selection are particularly important when dealing with Tables.

When you create a table, by default you will see something like this on the page.

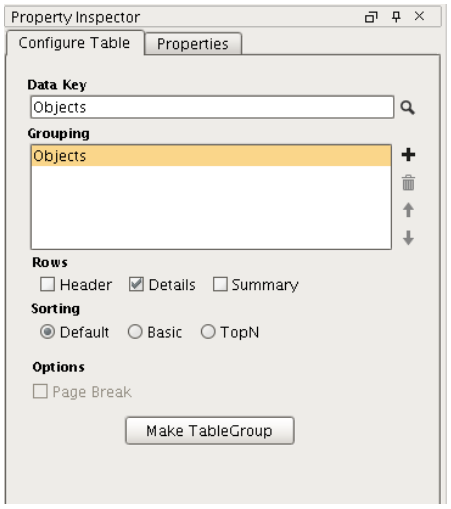


Anatomy of a Table

Get Familar with the Table's Features!

While not exactly complex, it's important to understand the different parts of a table and how they interact. First, the size of the component on the design page dictates how large the table can be on the generated report. Second, in this table, we have a single dark bar with the text "Object Details" on it, with four fields we can super select above. These dark bars are called *Rows*, we will talk about them soon.

If you look at the Table Configuration panel, you will see that it has a default data key of "Objects", which matches the Details row on the component. Changing the Data Key will also change the name on the row making it easier to determine what you are looking at when editing multiple tables or Grouping.


Property Inspector

Configure Table | Properties

Data Key: Objects

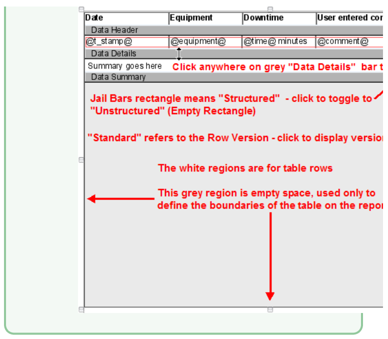
Grouping: Objects

Rows: [ ] Header [✓ Details] [ ] Summary

Sorting: [✓ Default] Basic TopN

Options: [ ] Page Break

Make TableGroup


Date	Equipment	Downtime	User entered co
Data Header			
@t_stamp@	@equipment@	@time@ minutes	@comment@
Data Details			
Summary goes here			
Data Summary			

Click anywhere on grey "Data Details" bar t

Jail Bars rectangle means "Structured" - click to toggle to "Unstructured" (Empty Rectangle)

"Standard" refers to the Row Version - click to display versio

The white regions are for table rows

This grey region is empty space, used only to define the boundaries of the table on the repo

Sorting

Sorting orders your data by a key or list of keys. There are three types of Sorting in Tables.

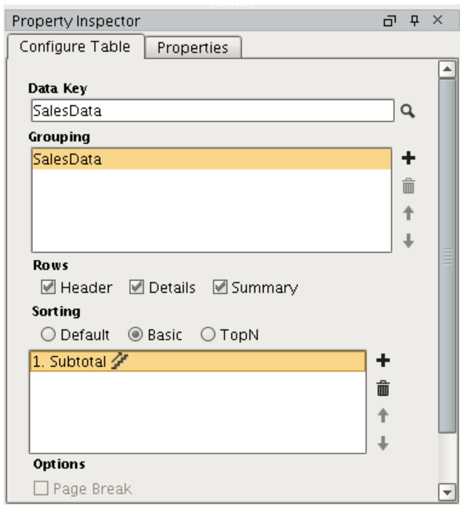
- **Default** - data is sorted based on the order in which it is retrieved.
- **Basic** - takes a list of keys and sorts by the first one. If the sort results in a tie, the tie will be resolved by the next key in the list, and so on.
- **TopN** - uses a single key path, with a *Count* value that allows a limit to the number of rows that are processed.

Basic and TopN sorts can be configured for either ascending ( ) or descending ( ) sorts. They can also utilize aggregate (calculation) keys.

The TopN option *Include Others* ☐ Include "Others" will include all values outside of the specified *Count* range by compressing them into a single row.

Basic Sort Example

We will look at the same table using different sorts. Both have the same exact layout (we made the first, and simply copy/paste to replicate). The first uses Default sort, the second will use a Basic Sort as configured in the image.


Property Inspector

Configure Table | Properties

Data Key: SalesData

Grouping: SalesData

Rows: [✓ Header] [✓ Details] [✓ Summary]

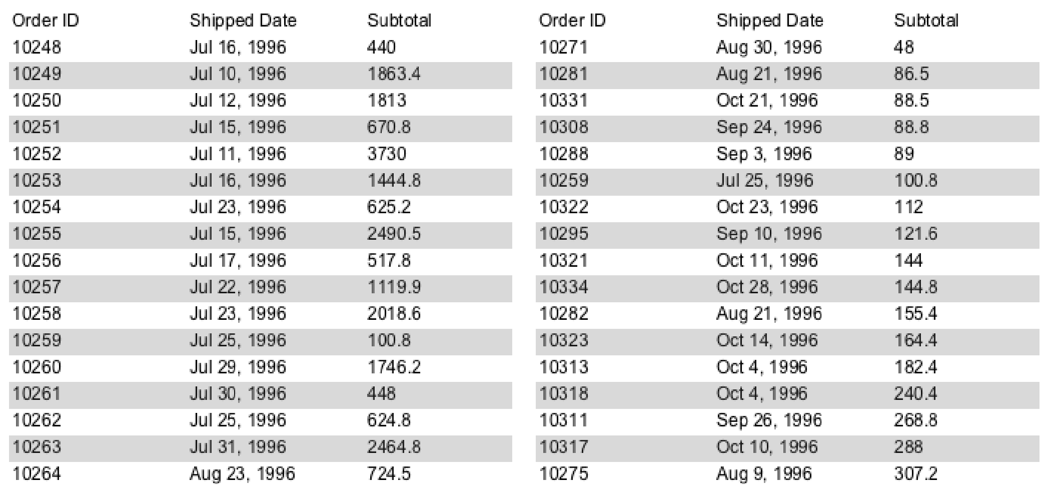
Sorting: Default [✓ Basic] TopN

1. Subtotal

Options: [ ] Page Break

To alternate to descending, we could click the icon and it would toggle to a descending sort.

These are our resulting tables, showing the left table sorted by its primary key (OrderID) and the right sorted by Ascending Subtotal.



Order ID	Shipped Date	Subtotal	Order ID	Shipped Date	Subtotal
10248	Jul 16, 1996	440	10271	Aug 30, 1996	48
10249	Jul 10, 1996	1863.4	10281	Aug 21, 1996	86.5
10250	Jul 12, 1996	1813	10331	Oct 21, 1996	88.5
10251	Jul 15, 1996	670.8	10308	Sep 24, 1996	88.8
10252	Jul 11, 1996	3730	10288	Sep 3, 1996	89
10253	Jul 16, 1996	1444.8	10259	Jul 25, 1996	100.8
10254	Jul 23, 1996	625.2	10322	Oct 23, 1996	112
10255	Jul 15, 1996	2490.5	10295	Sep 10, 1996	121.6
10256	Jul 17, 1996	517.8	10321	Oct 11, 1996	144
10257	Jul 22, 1996	1119.9	10334	Oct 28, 1996	144.8
10258	Jul 23, 1996	2018.6	10282	Aug 21, 1996	155.4
10259	Jul 25, 1996	100.8	10323	Oct 14, 1996	164.4
10260	Jul 29, 1996	1746.2	10313	Oct 4, 1996	182.4
10261	Jul 30, 1996	448	10318	Oct 4, 1996	240.4
10262	Jul 25, 1996	624.8	10311	Sep 26, 1996	268.8
10263	Jul 31, 1996	2464.8	10317	Oct 10, 1996	288
10264	Aug 23, 1996	724.5	10275	Aug 9, 1996	307.2

Filtering

Filtering gives the option of processing data based on an expression. The Filtering property can be found in the Property Inspector under the **Properti es** tab.

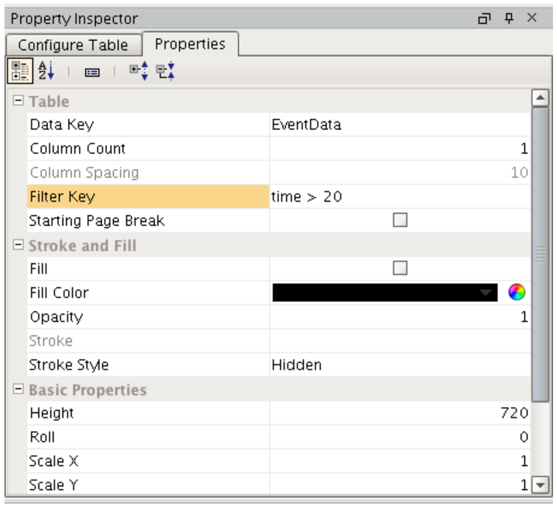
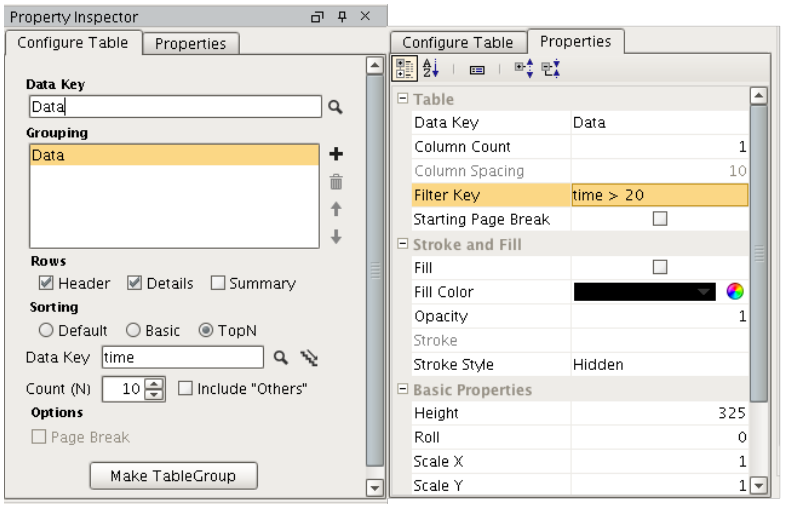


Table	
Data Key	EventData
Column Count	1
Column Spacing	10
Filter Key	time > 20
Starting Page Break	☐
Stroke and Fill	
Fill	☐
Fill Color	■
Opacity	1
Stroke	
Stroke Style	Hidden
Basic Properties	
Height	720
Roll	0
Scale X	1
Scale Y	1

If the expression resolves false, the row will be skipped. Note: you do not need @ symbols to reference keys.

Filtered TopN Sort Example

This example is sorted descending by downtime and filtered by downtime greater than 20 minutes.



Data Key: Data

Grouping: Data

Rows: [✓ Header] [✓ Details] Summary

Sorting: Default Basic [✓ TopN]

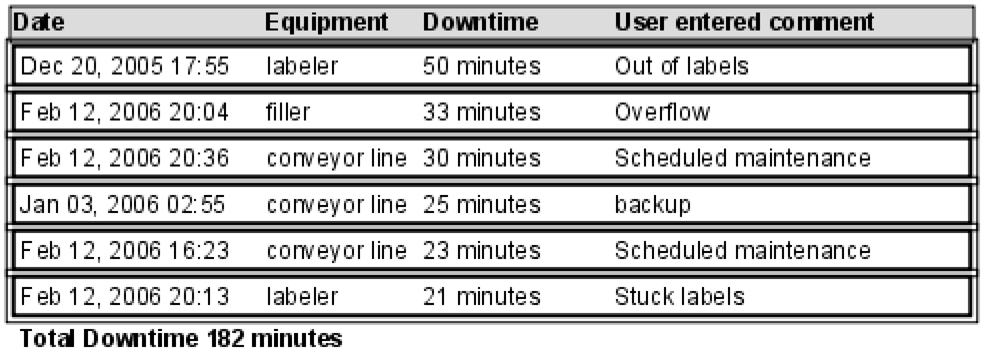
Data Key: time

Count (N): 10 Include "Others"

Options: Page Break

Make TableGroup

Table	
Data Key	Data
Column Count	1
Column Spacing	10
Filter Key	time > 20
Starting Page Break	☐
Stroke and Fill	
Fill	☐
Fill Color	■
Opacity	1
Stroke	
Stroke Style	Hidden
Basic Properties	
Height	325
Roll	0
Scale X	1
Scale Y	1



Date	Equipment	Downtime	User entered comment
Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels

Total Downtime 182 minutes

In this section ...

Table Header, Details and Summary Rows

Rows are an important fundamental aspect of tables. The different types of rows can be independently enabled for each level of Grouping, and for each table in a Table Group. Adding or removing rows is as simple as selecting the Row boxes in the Table Configuration tab of the Property Inspector. Additionally, Table Row Versioning gives you the option of conditionally displaying rows with a different format

Table Header, Details and Summary Rows

Watch the Video

Row Types

Header Row

The header row is used to add such report features as column labels. An interesting feature of the header is Reprint When Wrapped, which allows a different header on every page after the first. The main data in a table has one header row. Each subgroup of data can have its own row header.

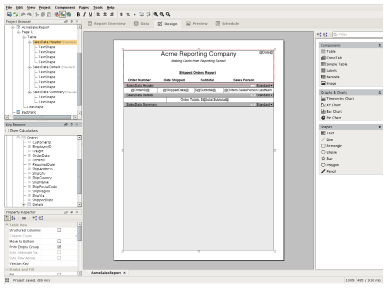
With grouping, the "top" level Header is the first row for the entire report. Lower level Headers fall immediately below higher level Details. In many cases where one is used, the other could be used equivalently in its case.

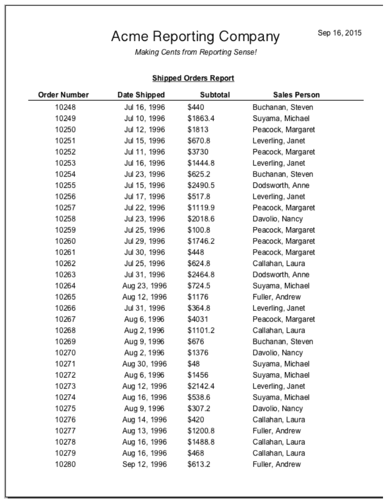
Detail Rows

The detail rows typically represent the majority of the data on a table or the "middle" rows. You might disable detail rows in unusual situations such as only displaying aggregate summaries in a grouped report. With grouping, the Detail rows appear below the same level Header and above the Header of the next level.

Summary Row

The summary row works like the header row. It prints at the bottom of the table. With grouping, Summary rows are always last, always in the opposite order of the Headers.





Acme Reporting Company — Sep 16, 2015

Making Cents from Reporting Sense!

Shipped Orders Report

Order Number	Date Shipped	Subtotal	Sales Person
10248	Jul 16, 1996	$440	Buchanan, Steven
10249	Jul 10, 1996	$1863.4	Suyama, Michael
10250	Jul 12, 1996	$1813	Peacock, Margaret
10251	Jul 15, 1996	$670.8	Leverling, Janet
10252	Jul 11, 1996	$3730	Peacock, Margaret
10253	Jul 16, 1996	$1444.8	Leverling, Janet
10254	Jul 23, 1996	$625.2	Buchanan, Steven
10255	Jul 15, 1996	$2490.5	Dodsworth, Anne
10256	Jul 17, 1996	$517.8	Leverling, Janet
10257	Jul 22, 1996	$1119.9	Peacock, Margaret
10258	Jul 23, 1996	$2018.6	Davolio, Nancy
10259	Jul 25, 1996	$100.8	Peacock, Margaret
10260	Jul 29, 1996	$1746.2	Peacock, Margaret
10261	Jul 30, 1996	$448	Peacock, Margaret
10262	Jul 25, 1996	$624.8	Callahan, Laura
10263	Jul 31, 1996	$2464.8	Dodsworth, Anne
10264	Aug 23, 1996	$724.5	Suyama, Michael
10265	Aug 12, 1996	$1176	Fuller, Andrew
10266	Jul 31, 1996	$364.8	Leverling, Janet
10267	Aug 6, 1996	$4031	Peacock, Margaret
10268	Aug 2, 1996	$1101.2	Callahan, Laura
10269	Aug 9, 1996	$676	Buchanan, Steven
10270	Aug 2, 1996	$1376	Davolio, Nancy
10271	Aug 30, 1996	$48	Suyama, Michael
10272	Aug 6, 1996	$1456	Suyama, Michael
10273	Aug 12, 1996	$2142.4	Leverling, Janet
10274	Aug 16, 1996	$538.6	Suyama, Michael
10275	Aug 9, 1996	$307.2	Davolio, Nancy
10276	Aug 14, 1996	$420	Callahan, Laura
10277	Aug 13, 1996	$1200.8	Fuller, Andrew
10278	Aug 16, 1996	$1488.8	Callahan, Laura
10279	Aug 16, 1996	$468	Callahan, Laura
10280	Sep 12, 1996	$613.2	Fuller, Andrew

Row Configuration

Configuring rows is easy with just a little know-how. Rows are best understood with a simple example, so we'll reference the design table on the right as an example.

To begin making changes to a row, you first need to select the row you'd like to modify by clicking the row on the component or the selecting the row's node in the Project Browser tree. In the image, the Header row has been selected. We can tell by looking at the Project Browser tree, or by looking at the red outline of the fields above where it says *SalesData Header* (the row's content is laid out above its respective row bar).

The four text fields that are above the *Details* row comprise what we call a *Structured Row*. A structured row can have a variable number of fields, and can be configured in the **Properties** tab of the component's Property Inspector panel. To view the properties of our Details row (or any row), we will first need to super-select the row by clicking on the dark bar in the component or the "Details" node in the Project Browser. We can confirm that this row is structured by looking at the Property table for the "Structured Columns" checkbox, or by looking for the Jail Bars icon next to the row version label for the *Standard* row version.

Structured rows can provide more control over the layout, wrapping, and organization of your text based table data. Since structured rows are intended for text layouts, they can not contain things like images and barcodes. To embed things like images and charts, we'll want to use *Unstructured Rows.* An unstructured row is functionally very similar to structured rows, except that it doesn't offer the same column based text constraints. The header of the image shows an example of using an unstructured row. In this case, we have added multiple Text shapes to add our logo, a title, and the a date *Key* in the corner to add the date every time a report is executed. A Line shape was used to underline our column headers, which are also in a Text shape.

We can alter between Structured and Unstructured rows in the Properties tab when the desired row is selected, or by clicking the white box on the row we'd like to toggle.

Next...

- Table Row Versioning

Table Row Versioning

Row versions allow you to conditionally display rows of data in different format. They are used to make certain data stand out or to make your report more legible.

Row versions are either builtin or user defined and may be specified with a version key expression. They are applicable to header, detail, and summary rows

Table Row Versioning

Watch the Video

Builtin Row Versions

By default reports use the Standard row version. Here are the builtin row versions:

Row Versions	Description
Standard	Default row version
First Only	Applies only to the first instance of the row. Good for showing header information without using an upper level detail row.
Reprint	Applies every page after the first. Good for one time headers or (continued) indications to save space.
Alternate	Applies every other row. Good for grey-bar reports by changing the background color.
TopN Others	Applies to count number of rows in a TopN sort. Using "include others" will then distinguish between TopN and non-TopN rows.
Split Header	Applies to headers that has been split due to excessive height. Good for providing "Continued" type indicators.
Custom	Create your own. Must be selected manually or in conjunction with the row's Version Key property

User Defined Row Versions

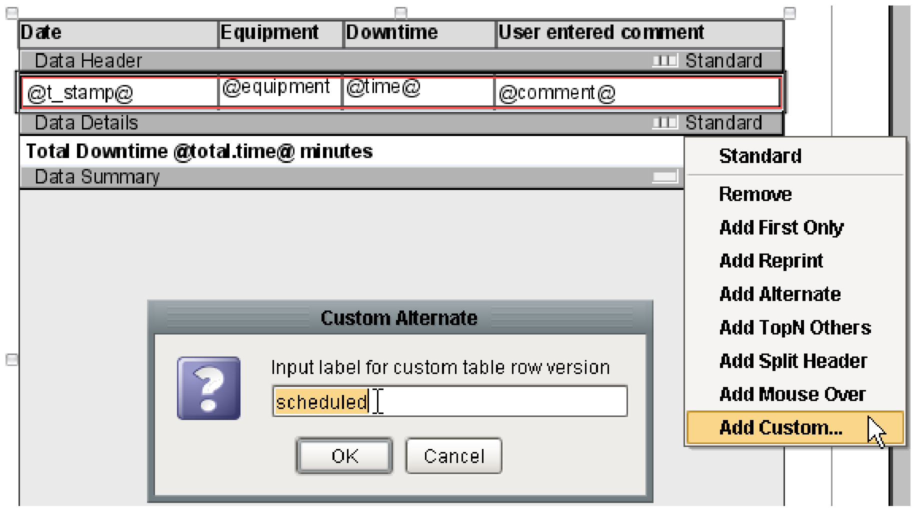
User defined row versions are identified by a string based name. They will be used when the Row Version Key expression is a string that matches the row version name.

Row Version Key

The Row Version Key is an expression that must return a string. If that string equals the name of a row version, either builtin or user defined, that version will be used. An invalid string will default back to normal builtin row version behavior.

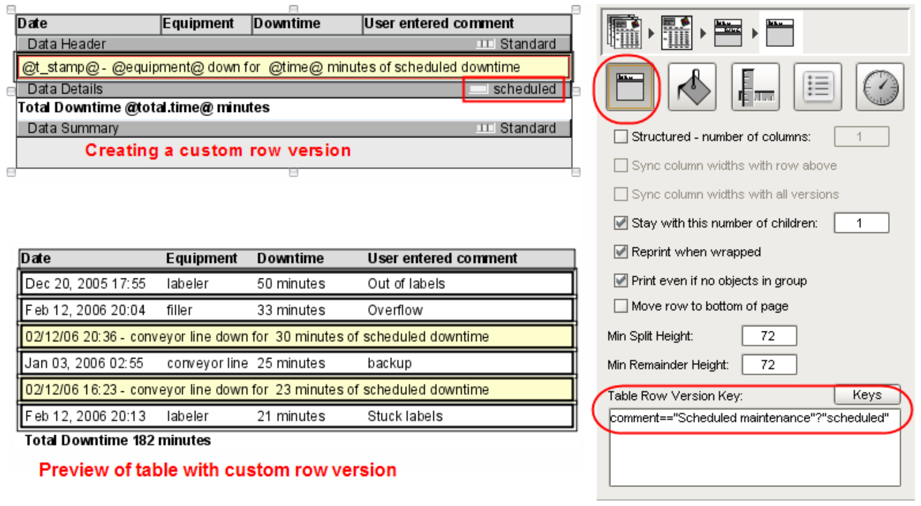
Row Version Example

1. Add a custom row version. scheduled, in this case.



2. Select your row and customize it

3. Specify Table Row Version Key. Tip: start with the expression "scheduled" to try out your custom row version before using more complex expressions. In this case we use: IF comment = "Scheduled maintenance" THEN use our custom row version.



Date	Equipment	Downtime	User entered comment
Data Header			Standard
@t_stamp@ - @equipment@ down for @time@ minutes of scheduled downtime			
Data Details			scheduled
Total Downtime @total.time@ minutes			
Data Summary			Standard

Creating a custom row version

Date	Equipment	Downtime	User entered comment
Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Feb 12, 2006 20:04	filler	33 minutes	Overflow
02/12/06 20:36 - conveyor line down for 30 minutes of scheduled downtime			
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
02/12/06 16:23 - conveyor line down for 23 minutes of scheduled downtime			
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels
Total Downtime 182 minutes			

Preview of table with custom row version

- [ ] Structured - number of columns: 1
- [ ] Sync column widths with row above
- [ ] Sync column widths with all versions
- [x] Stay with this number of children: 1
- [x] Reprint when wrapped
- [x] Print even if no objects in group
- [ ] Move row to bottom of page

Min Split Height: 72

Min Remainder Height: 72

Table Row Version Key: Keys

comment=="Scheduled maintenance"?"scheduled"

```
comment=="Scheduled maintenance"?"scheduled"
```

Tip

When using an IF condition for row versioning leave out the ELSE. Your table will then still respect builtin row versions. If you defaulted the ELSE to "Standard", none of the builtin versions such as Alternate would ever appear.

Tip

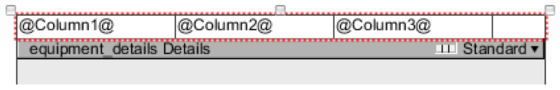
Make sure that you're happy with the Standard row version before you create other row versions. This will save you time as other versions begin as a copy of Standard.

Next...

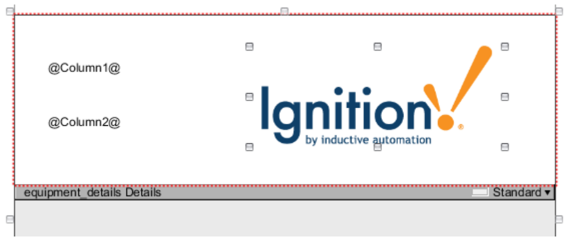
- Table Row - Structured vs Unstructured

Table Row - Structured vs Unstructured

Structured rows are by far the most commonly used table row setting. It makes it convenient to add Data Keys, and resizing one column is automatically compensated for by the adjoining column. However structured rows only allow for text shapes.



Unstructured rows allow for much more freedom on what can be placed in the table row. It is possible to have images and charts placed into an unstructured row, as well as place text shapes anywhere within the row.



Next...

- Dataset Grouping

INDUCTIVE UNIVERSITY

Table Row - Structured vs Unstructured

Watch the Video

Dataset Grouping

Dataset Grouping breaks tables down by keys that share a common value. Tables support an arbitrary level of groups. Each can have its own Header, Detail, and Summary rows. Additionally, totals and other keychain functions are supported for any level of grouping.

INDUCTIVE UNIVERSITY

Dataset Grouping

Watch the Video

Separating Groups with Page Breaks

Selecting the
Options
Page Break

option will create breaks between each Grouping. Each new instance of that level of grouping will create a new page in the report.

In the example above, separating the equipment level of grouping by page would create separate report pages for the following: `labeler`, `filler`, `palletizer`, and `converyor line`.

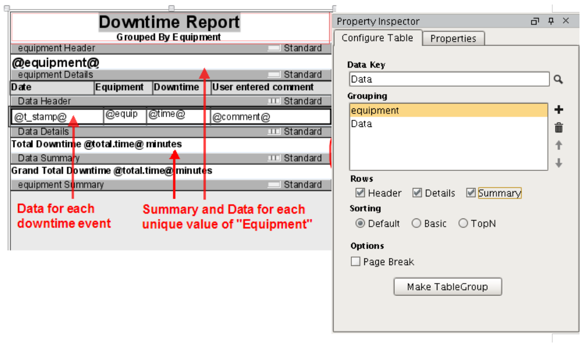
Example

This example begins with a table similar to that created in the Table Basics tutorial. We'll group our existing downtime report table by equipment.

Drag and drop a key from the Key Browser to the selected Table's Grouping configuration list to add it to the Group.

1. Drag the equipment data key from the Key Browser to the Grouping field of the Table Configuration panel, or select the
Make TableGroup
button.
2. Enable Header, Detail and Summary rows.
3. Add your header and details.
4. Use `@total.time@` for both summary rows. Notice that the total respects grouping.

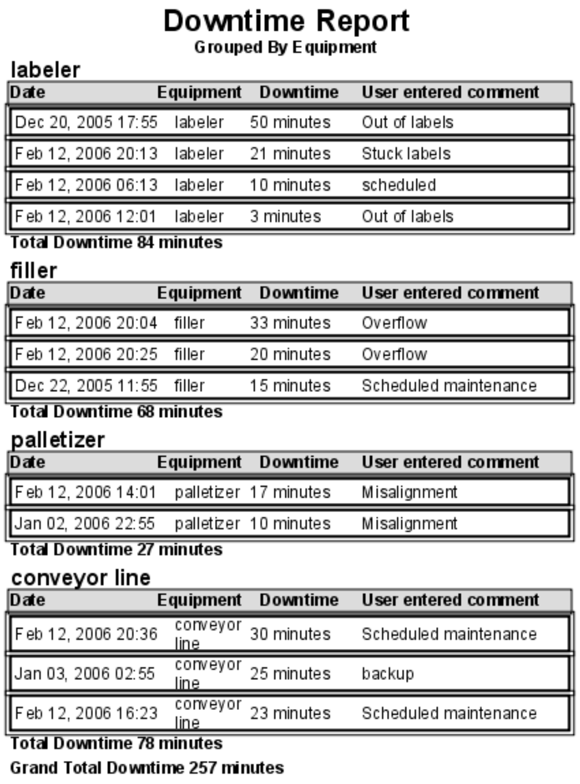
In the equipment Summary row total.time is a sum of all time at that level of grouping, which includes all downtime events. In the Data Summary row total.time is a sum of all downtime at that level of grouping, total time that has already been grouped by equipment, equivalently, total downtime by equipment.


Downtime Report
Grouped By Equipment
equipment Header — Standard
@equipment@
equipment Details — Standard
Date | Equipment | Downtime | User entered comment
Data Header — Standard
@t_stamp@ | @equip | @time@ | @comment@
Data Details — Standard
Total Downtime @total.time@ minutes
Data Summary — Standard
Grand Total Downtime @total.time@ minutes
equipment Summary — Standard

Data for each downtime event

Summary and Data for each unique value of "Equipment"

Property Inspector
Configure Table | Properties
Data Key: Data
Grouping: equipment, Data
Rows: [✓ Header] [✓ Details] [✓ Summary]
Sorting: [✓ Default] Basic TopN
Options: Page Break
Make TableGroup



Downtime Report

Grouped By Equipment

labeler

Date	Equipment	Downtime	User entered comment
Dec 20, 2005 17:55	labeler	50 minutes	Out of labels
Feb 12, 2006 20:13	labeler	21 minutes	Stuck labels
Feb 12, 2006 06:13	labeler	10 minutes	scheduled
Feb 12, 2006 12:01	labeler	3 minutes	Out of labels

Total Downtime 84 minutes

filler

Date	Equipment	Downtime	User entered comment
Feb 12, 2006 20:04	filler	33 minutes	Overflow
Feb 12, 2006 20:25	filler	20 minutes	Overflow
Dec 22, 2005 11:55	filler	15 minutes	Scheduled maintenance

Total Downtime 68 minutes

palletizer

Date	Equipment	Downtime	User entered comment
Feb 12, 2006 14:01	palletizer	17 minutes	Misalignment
Jan 02, 2006 22:55	palletizer	10 minutes	Misalignment

Total Downtime 27 minutes

conveyor line

Date	Equipment	Downtime	User entered comment
Feb 12, 2006 20:36	conveyor line	30 minutes	Scheduled maintenance
Jan 03, 2006 02:55	conveyor line	25 minutes	backup
Feb 12, 2006 16:23	conveyor line	23 minutes	Scheduled maintenance

Total Downtime 78 minutes

Grand Total Downtime 257 minutes

Next...

- Charts Inside of Tables

Charts Inside of Tables

When a table row is made unstructured it is possible to place a chart inside the row.

Example

1) Create a Static CSV data source named "Fruit Data" with the following data:

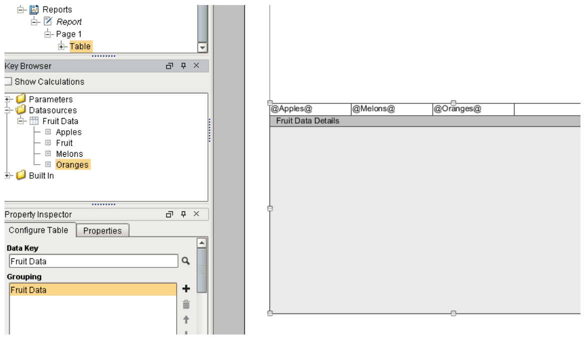
```
Fruit, Apples,Oranges,Melons
Quantity,15,30,23
```

INDUCTIVE UNIVERSITY

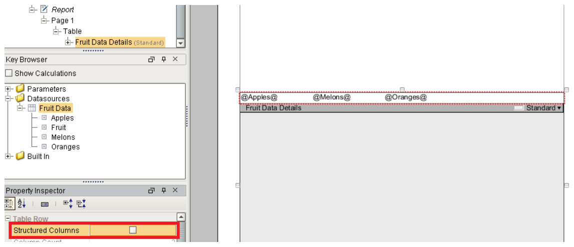
Charts Inside of Tables

Watch the Video

2) Place a Table on the report. Set the Data Key property to "Fruit Data", and place the @Apples@, @Melons@, and @Oranges@ keys ineach data source key in a separate column on the table.

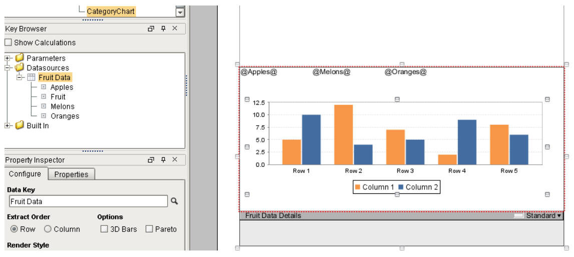


3) Select the Fruit Data Details row and uncheck the Structured Columns property

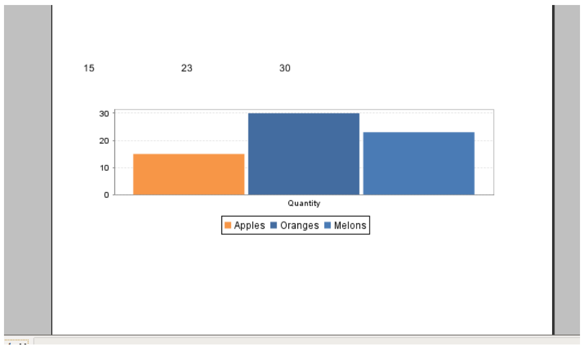


4) Click and drag the Fruit Data Details row to make some room for the chart.

5) With Fruit Data Details still selected, drop a Bar Chart component onto the window. Set the Bar Chart's Data Key property to the "Fruit Data" data source.



6) Switch to the Preview Panel. You'll see the chart in the same unstructured row.


15 23 30

Next...

- Table Groups
- Dataset Grouping

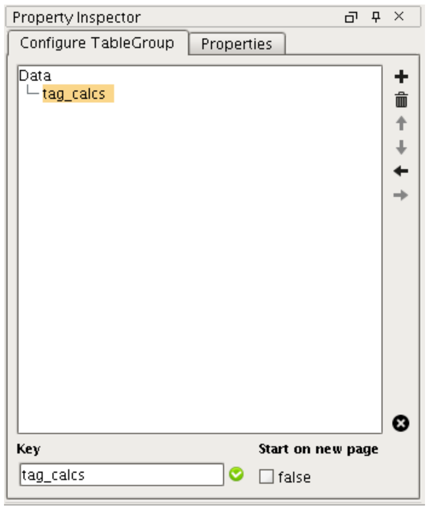
Table Groups

Table groups allow you to specify child tables for each object in the master list (using a list key found in each of those objects). It also allows you to specify additional "peer" tables that pick-up exactly where the first table ends (note: multiple tables can also be configured as multiple- page templates, providing a page break between tables).

Table groups and Nested Data Sources in Ignition 7.8 are particularly well suited for each other. We've made it easier than ever before to create things like Summary Tables for categories of items, or drill-down charts.

Usage

- To turn a table into a table group, simply select the table and click the Make TableGroup button in the Table Configuration tab of the selected Table. The table is actually a child of a "Table Group" element, which has its own configuration panel in the Property Inspector.



Key: tag_calcs
Start on new page: false

To add a Child or Peer table, click the button. This will add a whole new table for this "child" list key. You can use the arrow buttons to change the hierarchy of the tables in the group, or remove them with the trash icon. To ungroup, select the button.

You can edit each of the different tables in the table hierarchy by clicking their node in the table tree, and then clicking the body component on the page to activate its configuration. Double-click a node to get its table inspector (or double click on the table template in the open document).

You can get back to the table group inspector by clicking on the "Table Group" button at the bottom left corner of the table template, or by selecting the table group node in the Project Browser.

Parent Reference

To reference the parent row object from a child table, you can simply use the key prefix "Parent". So if a row in a movie role child table wanted to display the movie title, it could use the key "@Parent. getTitle@".

Next...

- Table Grouping - Peer Tables

Table Grouping - Peer Tables

Title

Table Grouping - Peer Tables

Watch the Video

Next...

- Table Grouping - Child Tables

Table Grouping - Child Tables

Child tables are inserted in between each row of the parent table. One table could potentially show details for each piece of equipment, with the child table could show all downtime occurrences for the row on the parent table.

Table Grouping - Child Tables

Watch the Video

Next...

- Simple Table

Simple Table

The Simple Table Simple Table is a grid-like structure that dynamically creates new rows and columns for row returned by the Data Keys on the component.

INDUCTIVE UNIVERSITY

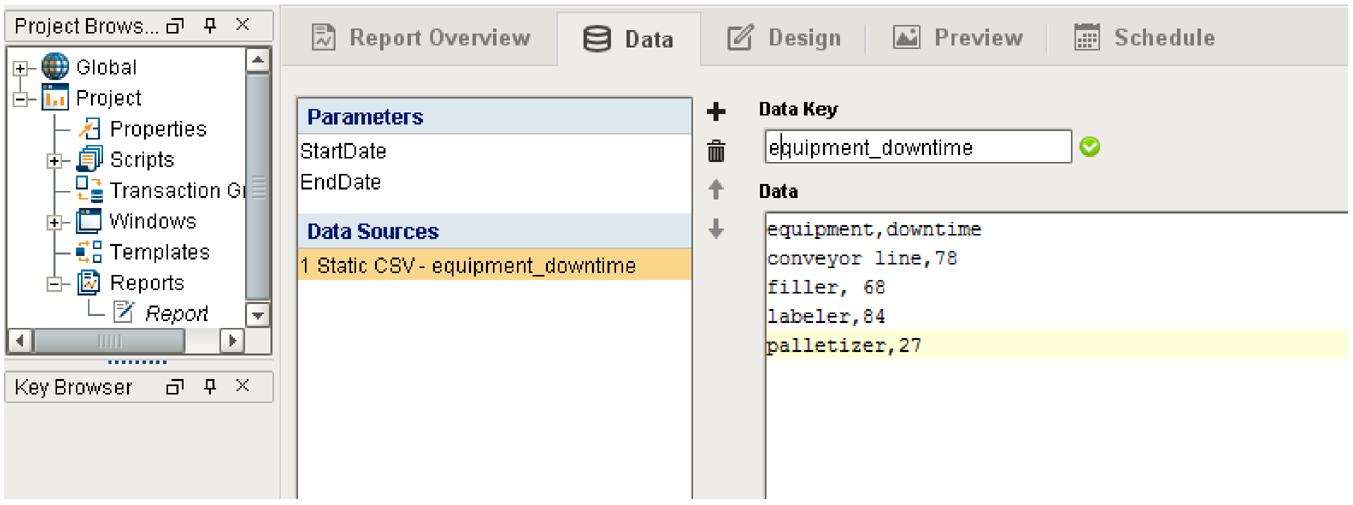
Simple Table

Watch the Video

Example

First we'll create a Static CSV data source named equipment_downtime. This data source will contain the following data:

```
equipment,downtime
conveyor line,78
filler, 68
labeler,84
palletizer,27
```

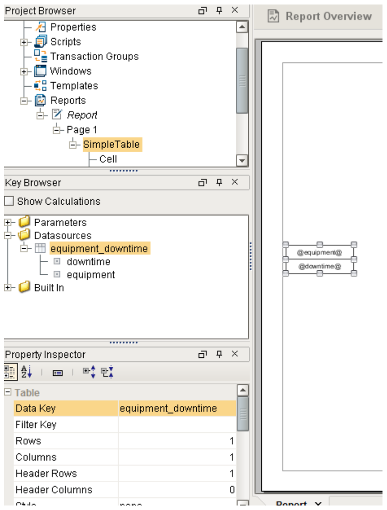


Data Key: equipment_downtime

```
equipment,downtime
conveyor line,78
filler, 68
labeler,84
palletizer,27
```

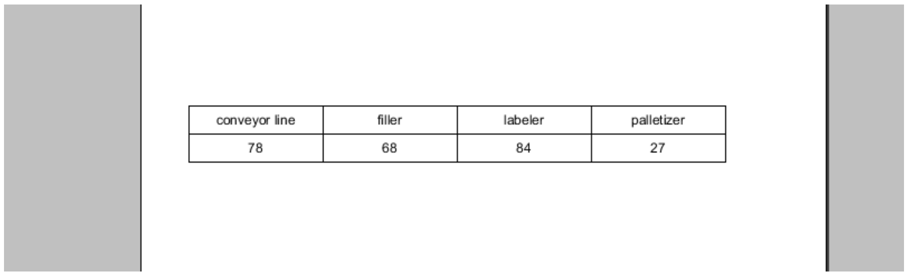
Next, head over to the Design panel. Add a new Simple Table to your report. With the Simple Table selected, set the Data Key property to the equipment_downtime data source. Then change the number of Rows and Columns to 1.

Lastly, drag the @equipment@ Data Key to the top row of the Simple Table, and the @downtime@ Data Key to the 2nd row. The table should look like the following:



Data Key	equipment_downtime
Filter Key	
Rows	1
Columns	1
Header Rows	1
Header Columns	0

Switch to the Preview panel and the table should look like the following:



conveyor line	filler	labeler	palletizer
78	68	84	27

Next...

- CrossTab Table

CrossTab Table

The Cross-Tabulation or CrossTab CrossTab component is a tabular data element like the Table and Charts. Also known as *Contingency Tables,* they are commonly used to summarize the relationship between two categories of data by showing summaries of cross sections of the data source. To be useful, crosstab data should have the following:

- Lots of repetitious data. Sums, Averages, and other Aggregating functions are well represented by crosstabs.
- A data source that provides at least two columns of data which are repetitious compared to the number of rows.
- One or more columns that represent a value that requires calculation. Examples are: summing money, displaying average response times, counting occurrences, etc. These calculations may be provided as columns or calculations of the data source, or as Keychain Expressions.

CrossTab Table

Watch the Video

Usage

The CrossTab template is much simpler than the table template. By default it just shows one cell of a simple table. This is usually configured with an aggregate key, like "@total.getAmount@". After that, grouping keys are dragged to the horizontal and vertical axis.

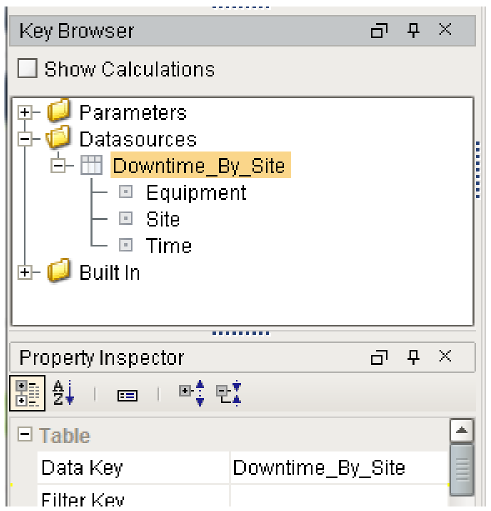
Example

First, create a Static CSV data source with the following data. Name the data source "Downtime_By_Site"

Downtime_By_Site

```
Equipment,Time,Site
Motor,15,Site A
Motor,23,Site A
Conveyor Line,148,Site B
Pallet Wrapper,58,Site A
Motor,96,Site C
Conveyor Line, 23, Site B
Palletizer,40,Site B
Palletizer,30,Site D
Conveyor Line,56,Site A
Pallet Wrapper,45,Site C
Motor,43,Site C
Conveyor Line,87,Site D
Motor,23,Site D
```

Next, switch to the Design panel and create a new CrossTab component. With the CrossTab component still selected, drag-and-drop Downtime_By_Site from the Key Browser to the Data Key property in the Property Inspector



Key Browser

Show Calculations

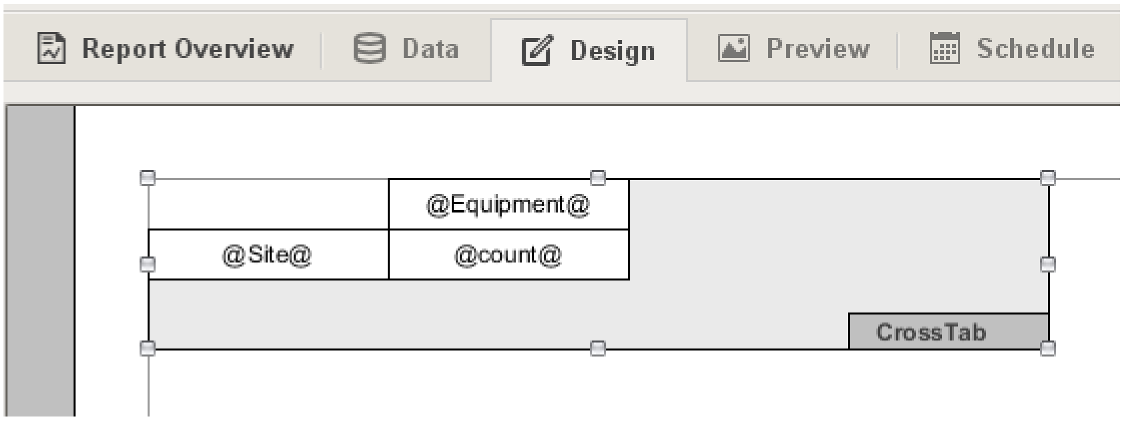
- Parameters
- Datasources
  - Downtime_By_Site
    - Equipment
    - Site
    - Time
- Built In

Property Inspector

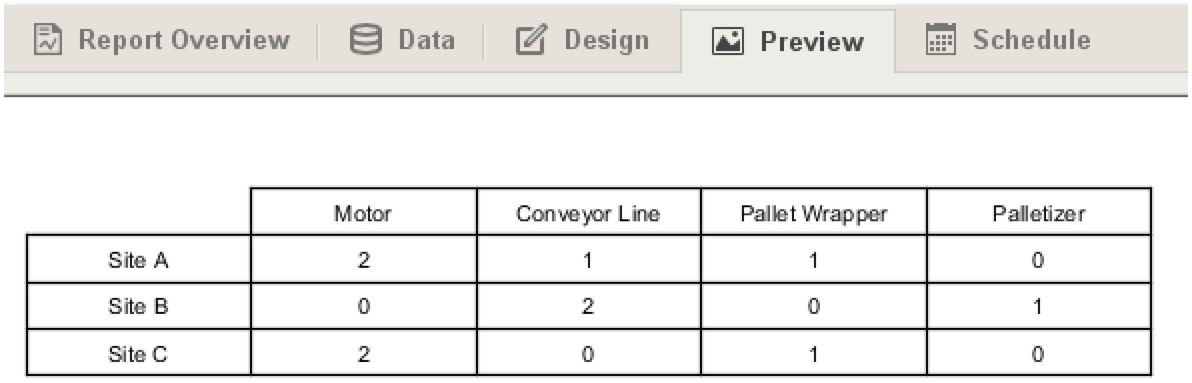
Table

Data Key	Downtime_By_Site
Filter Key	

Now we need to add some data keys to the cells in the CrossTab component. We'll add @Equipment@ to the top cell, @Site@ to the leftmost cell, and @count@ to the remaining cell. This will show a count of each piece of equipment by site. The CrossTab should look like the following:



Switch to the Preview panel and your cross tab will display the results



	Motor	Conveyor Line	Pallet Wrapper	Palletizer
Site A	2	1	1	0
Site B	0	2	0	1
Site C	2	0	1	0

Next...

- Report Schedules

Report Schedules

Run and Deliver Reports Automatically

Perhaps one of the most compelling additions of the Reporting v3 module is the new Scheduling system. It gives you total flexibility on when reports should run, and how they should be delivered.

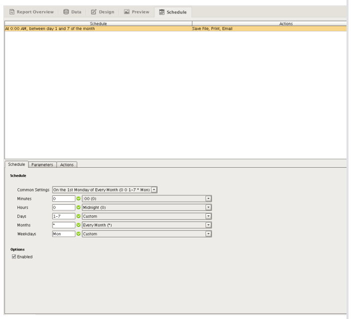
How it works

For any Scheduled Report, there are two settings which require configuration:

1. When the Report executes
2. The Action which occurs following Report Generation

Additionally, parameters can be optionally overridden by scheduled reports. Reports can have multiple Schedules, and multple Actions per Schedule. Schedules run with minute resolution, meaning that we can schedule a report down to a specific minute (if second resolution is required, scripting can fill that void). It is important to note that Schedules execute on the Ignition Gateway. This means that schedules should be created to Gateway local time, NOT Client/Designer time.

Configuring Report Schedules



Schedule	Actions
At 0:00 AM, between day 1 and 7 of the month	Save File, Print, Email

Schedule

Common Settings	On the 1st Monday of Every Month (0 0 1-7 * Mon)	
Minutes	0	:00 (0)
Hours	0	Midnight (0)
Days	1-7	Custom
Months	*	Every Month (*)
Weekdays	Mon	Custom

Options

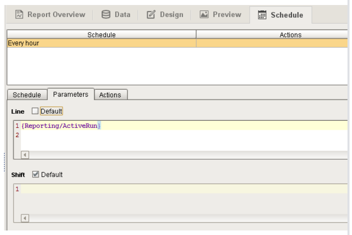
[✓] Enabled

Schedules

Creating a scheduled report is easy. On the Schedule tab, you'll notice the UI is split. On the top is a table which contains a list of all your schedules and a description of the schedule, as well as the Actions that will occur following report generation. To add a new schedule click on the ✚ button on the top-right corner of the Schedule tab. In doing so, you've created a new Schedule which can now configure.

Schedules are driven via Chron formatted strings, a popular scheduling format used in computing. If you aren't familiar with Chron, don't worry, we've created an intuitive UI that will allow you to set schedules easily without having to learn anything new. The Schedule UI provides some convenient pre-made Common Settings. If there isn't a setting for you in the Common Settings combo box, choose one that is close and then simply customize it using any of the selection boxes below. You can also enable or disable the schedule by checking the Enabled option.

Overriding Parameters in a Scheduled Report



Schedule	Actions
Every hour	

Line [ ] Default

```
{Reporting/ActiveRun}
```

Shift [✓] Default

The Parameter "Line" is being overridden by the value of the "Reporting/ActiveRun" tag.

Parameters

The Scheduler presents an additional opportunity to set parameter values. Alter default values to tailor scheduled reports without having to change the Report's design or configuration. For instance, imagine a report which summarizes how many Widgets a factory produced during a given shift. Rather than create a separate report for each shift, we can create multiple schedules (one for each shift) and simply alter a shift parameter. Using Parameters and Schedules, users can avoid creating multiple reports while keeping projects more maintainable.

Actions

Actions are triggered following the generation of a scheduled report. There can be any number of Actions associated with a Schedule, covering virtually any requirement for automatically storing, distributing or notifying upon completion of the report. Each action has its own custom configuration interface to make adding and editing Actions simple. Here is a brief summary of each Action:

Print File

Configure print settings which execute when the report is generated.

Email

Your report can be emailed to a list of emails, Rosters, or Roles of a roster.

FTP

Send your report to a file server for backups or storage.

Save File

An easy way to save a report to a location on your local gateway.

Run Script

A Run Script Action provides the ability to fully customize how a finished report is handled. The Run script Action provides the report's data as well as the bytes generated according to the *Format* option in the configuration panel.

In this section ...

Scheduling Basics

Schedules Table

Schedule Tab

Scheduling Basics

Watch the Video

Parameter Tab

Action Tab

Next...

- Scheduling Actions - Save

Scheduling Actions - Save

Scheduling Actions - Save

Watch the Video

FTP

Print File

Schedule a Print Job to a connected printer when report is generated.

Save

If a report is saved as HTML, it will output both <report>.html and an images folder containing a .png of the report. Both parts need to be present to view the report properly.

Next...

- Scheduling Actions - Email

Scheduling Actions - Email

Email Action

The Scheduled Email Action will distribute a Report via email when it is finished. Emails can be provided either by adding a list of email addresses in the Action interface, or by choosing from configured Rosters and User Roles. The sending address, email subject, email body and file name are all configurable. Subject, Filename and Body editors can utilize Expressions to dynamically add content or change names.

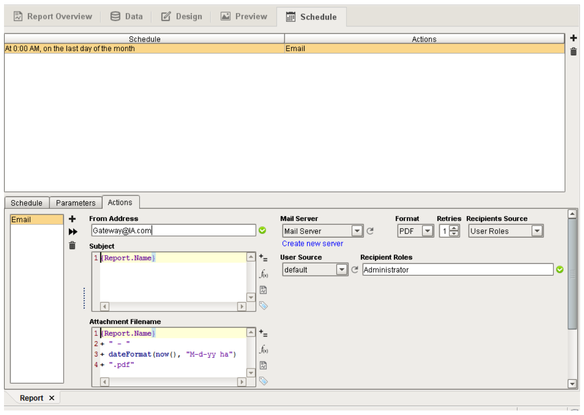
Scheduling Actions - Email

Watch the Video

Usage

Email Server settings must first be established in the Gateway by going to **Configuration > Email Settings**. Once you have created and saved an SMTP profile you can test your email settings on the Email settings page.

With an SMTP profile functional, simply fill out the fields of your Email Action for scheduled reports to be emailed on completion.



Schedule	Actions
At 0:00 AM, on the last day of the month	Email

Email

From Address: Gateway@IA.com

Subject:
```
{Report.Name}
```

Attachment Filename:
```
{Report.Name}
+ " - "
+ dateFormat(now(), "M-d-yy ha")
+ ".pdf"
```

Mail Server: Mail Server

Create new server

Format: PDF

Retries: 1

Recipients Source: User Roles

User Source: default

Recipient Roles: Administrator

Next...

- Other Reporting Module Components

Other Reporting Module Components

The Reporting Components for Vision are fairly similar to the previous Reporting module but do have some notable differences. The most obvious difference is that with Reports, now being Project-level resources, the *Report Component* has been replaced with a *Report Viewer*. We've maintained 100% backward compatibility for those who already have Reports through the previous version. If you have a report component on a project, don't worry, it will be there exactly as it was before upgrading!

Vision Reporting Components

Report Viewer

The new *Report Viewer* lets you visualize Reports in much the same way the old version did. Similarly, it also allows you to configure bindings to report parameters in a way similar to the old module's Report Component could have bound custom properties.

Row Selector

The row selector is a component that acts like a visual filter for datasets. It takes one dataset, chops it up into various ranges based on its configuration, and lets the user choose the splices. Then it creates a virtual dataset that only contains the rows that match the selected splices.

The most common way to splice the data is time. You could feed the row selector an input dataset that represents a large time range, and have it break it up by Month, Day, and then Shift, for example. Then you could power a report with the output dataset, and that would let the user dynamically create reports for any time range via an intuitive interface.

To configure the row selector, first you set up the appropriate bindings for its input dataset. Then you use its Customizer to alter the levels that it uses to break up the data. In the customizer, you add various filters that act upon columns in the input dataset, sorting them by various criteria. For example, you could choose a date column, and have it break that up by quarter. Then below that, you could have it use a discrete filter on a product code. This would let the user choose quarterly results for each product. Each level of filter you create in the customizer becomes a level in the selection hierarchy. Note, that the output data is completely unchanged other than the fact that rows that don't match the current user selection aren't present.

Column Selector

The column selector component is conceptually similar to the Row Selector, except that instead of filtering rows, it filters columns from its output dataset. Each column from the input dataset is shown as a checkbox. As the user checks and un-checks columns, the output dataset has those column s added or removed. This is very handy for driving the Table and Classic Chart components.

File Explorer

The File Explorer component displays a file system tree to the user. It can be rooted at any folder, even network folders. It can also filter the types of file s that are displayed by their file extension (For example, "pdf"). The path to the file that the user selects in the tree is exposed in the bindable property Selected Path.

This component is typically used in conjunction with the IA Labs PDF Viewer, in order to create a PDF viewing window. This is very useful for viewing things like maintenance manuals from within your project. To create a window like the one shown below follow these steps:

PDF Viewer

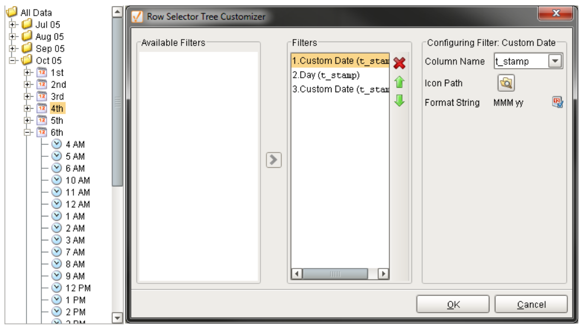
The PDF Viewer has been retooled for Ignition 7.8. Built on the same architecture our popular IA-Labs PDF Viewer module, the new PDF Viewer is much more capable of viewing pdfs from a wider variety of sources, and includes improved controls.

Component Reference

Reference documentation for Vision components is available in the Reporting Components section of the User Manual Appendix.

In this section...

Row Selector Component



1.Custom Date (t_stam
2.Day (t_stamp)
3.Custom Date (t_stam

Configuring Filter: Custom Date
Column Name t_stamp
Icon Path
Format String MMM yy

Row Selector

Watch the Video

The Row Selector is a component that allows users to filter a DataSet based on unique values of one or more columns. Each level in the sorting tree is based on these properties.

Additional information on the Row Selector can be found on the Row Selector page in the appendix.

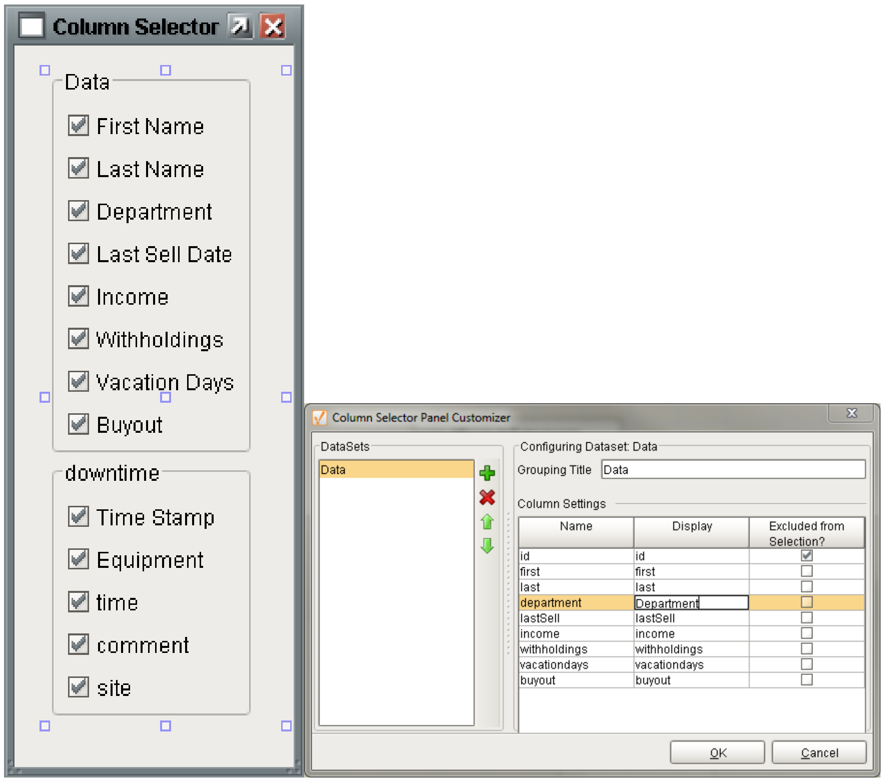
The user will see a dynamically generated expandable tree that groups their data by any number of choices. As they click down the tree, objects bound to the DataSet will indicate the filtered data. Here are a few examples.

- A line graph bound to a Row Selector. Set up grouping to be first by month and year, then day, then hour, like the top left illustration. Clicking on a month and year will dynamically update the graph for that time period. Further clicking to a specific day or hour will re-filter the graph for that period.
- A Report Viewer bound to a Row Selector. Grouping by department (String) would allow selection by department, automatically regenerating the Report on selection.
- An "alarm history" table bound to a Row Selector. This could first be broken down severity level (Integer), then broken into "Alarm Acknowledged" / "Not Acknowledged" (Boolean based). Clicking "Severity 3" would filter the table to all Severity 3 alarms. Selecting "Unacknowledged" would then filter the table to unacknowledged alarms of severity 3

Next...

- Column Selector Component
- Row Selector

Column Selector Component



Configuring Dataset: Data

Grouping Title: Data

Column Settings

Name	Display	Excluded from Selection?
id	id	✓
first	first	☐
last	last	☐
department	Department	☐
lastSell	lastSell	☐
income	income	☐
withholdings	withholdings	☐
vacationdays	vacationdays	☐
buyout	buyout	☐

Column Selector

Watch the Video

The Column Selector is a component that takes DataSets in, allows users to show or hide variables in the DataSets (Columns) via checkboxes, then outputs the resulting DataSet.

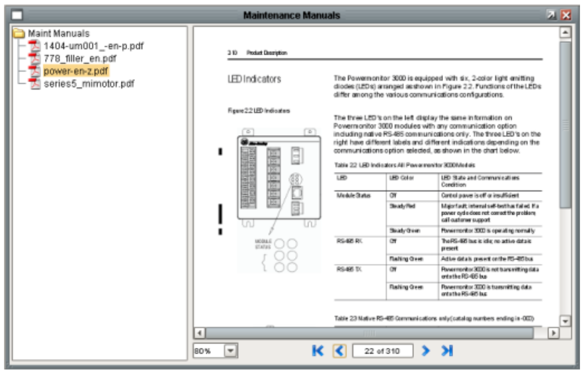
The Column Selector allows users to choose which columns in a DataSet that they wish to use. If an object is bound to the Column Selector it will update itself whenever a user checks or unchecks a column. This allows users to dynamically show/hide: Table columns, "pens" on a graph, data in a Report Viewer, or any other component set up to use a DataSet.

Additional information on the Column Selector can be found on the Column Selector page in the appendix.

Next...

- File Explorer and PDF Viewer
- Column Selector

File Explorer and PDF Viewer



INDUCTIVE UNIVERSITY

File Explorer and PDF Viewer

Watch the Video

The File Explorer component displays a filesystem tree to the user. It can be rooted at any folder, even network folders. It can also filter the types of files that are displayed by their file extension (For example, " pdf"). The path to the file that the user selects in the tree is exposed in the bindable property Selected Path.

This component is typically used in conjuction with the PDF Viewer component, in order to create a PDF viewing window. This is very useful for viewing things like maintenance manuals from within your project. To create a window like the one shown below follow these steps:

1. Bind the PDF Viewer's Filename property to the File Explorer's Selected Path property
2. Set the File Explorer's File extension filter to "pdf"
3. Set the File Explorer's Root Directory to a network folder that has your maintenance manuals in it. (Use a network folder so that all clients will be able to access the manuals).

Next...

- Scripting in Reporting

Scripting in Reporting

In this Section ...

Data Sources - Scripting

The Script data source allows you to add additional data into a report. Script data sources are able to reference data keys from other data sources. This allows you to modify keys from one data source, and return a new altered data source.

INDUCTIVE UNIVERSITY

Data Sources - Scripting

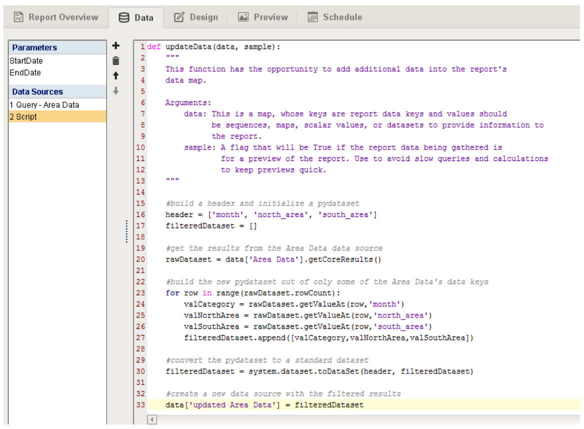
Watch the Video

Example

Say we have a query data source named "Area Data" which contains four columns: month, north_area, south_area, and t_stamp. If we need to build a new data source without the t_stamp column we can use the following code:

Script data source

```
	#build a header and initialize a pydataset
	header = ['month', 'north_area', 'south_area']
	filteredDataset = []

	#get the results from the Area Data data source
	rawDataset = data['Area Data'].getCoreResults()

	#build the new pydataset out of only some of the Area Data's data
keys
	for row in range(rawDataset.rowCount):
		valCategory = rawDataset.getValueAt(row,'month')
		valNorthArea = rawDataset.getValueAt(row,'north_area')
		valSouthArea = rawDataset.getValueAt(row,'south_area')
		filteredDataset.append([valCategory,valNorthArea,
valSouthArea])

	#convert the pydataset to a standard dataset
	filteredDataset = system.dataset.toDataSet(header, filteredDataset)

	#create a new data source with the filtered results
	data['updated Area Data'] = filteredDataset
```



```
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.
	"""

	#build a header and initialize a pydataset
	header = ['month', 'north_area', 'south_area']
	filteredDataset = []

	#get the results from the Area Data data source
	rawDataset = data['Area Data'].getCoreResults()

	#build the new pydataset out of only some of the Area Data's data keys
	for row in range(rawDataset.rowCount):
		valCategory = rawDataset.getValueAt(row,'month')
		valNorthArea = rawDataset.getValueAt(row,'north_area')
		valSouthArea = rawDataset.getValueAt(row,'south_area')
		filteredDataset.append([valCategory,valNorthArea,valSouthArea])

	#convert the pydataset to a standard dataset
	filteredDataset = system.dataset.toDataSet(header, filteredDataset)

	#create a new data source with the filtered results
	data['updated Area Data'] = filteredDataset
```

The above code would create a new data source in the Design panel named 'updated Area Data'. Note that .getCoreResults() is only applicable when the raw data source is a query. If 'Area Data' in the example above was a static CSV then you could get the data into a 'new data source' with the following:

Static CSV example

```
data['new data source'] = data['Area Data']
```

What if our 'Area Data' query has a nested query called 'Area Details' that we would like to manipulate in a script?

Script data source for nested query

```
nested = data['Area Data'].getNestedQueryResults()    # Gets results from
our parent query
subQuery = nested['Area Details']    #Gets the subquery we want -- there
can be more than one
header = ['productName', 'cost', 'triple']
alteredDataset = []
for child in subQuery:
    children = child.getCoreResults()    # children is a dataset
    for row in range(children.rowCount):
        valProductName = children.getValueAt(row,'productName')
        valCost = children.getValueAt(row,'cost')
        valTimesThree =  None
        if valCost != None:
            valTimesThree = 3 * valCost
        alteredDataset.append([valProductName,valCost,valTimesThree])


#convert the pydataset to a standard dataset
alteredDataset = system.dataset.toDataSet(header, alteredDataset)

#create a new data source with the altered results
data['Updated Area Details'] = alteredDataset
```

Next...

- Scheduling - Run Script

Scheduling - Run Script

The Run Script action is a scripting event that executes based on the schedule. This action does not inherently email, save, or otherwise distribute a report unless you do so in the script. However you can easily use system.report.executeAndDistribute() to send out the report.

Example

The following would insert a copy of the report into a database table, and then save the report on the gateway's desktop as a PNG

Run Script example

```
#save a copy of the report
query = "INSERT INTO report_log (name, data) VALUES (?,?)"
system.db.runPrepUpdate(query, [reportName,reportBytes])

#distribute the report
system.report.executeAndDistribute(path='my report', project='my project',
action='save', actionSettings = {'path':'C:
\\Users\\User\\Desktop','fileName':'SavedReport.png','format':''png'})
```

Scheduling - Run Script

Watch the Video

Next...

- Report Module Scripting Functions

Report Module Scripting Functions

The Reporting Module 7.8+ provides convenient scripting functions in `system.reporting` that enable report generation from scripting.

`system.report.executeReport(path, project, [parameters], fileType)`

Will immediately execute a report on use and return a `byte`[] array of the results.

See executeReport() in the appendix.

`system.report.executeAndDistribute(path, project, [parameters], action, [actionSettings])`

Will generate a report and distribute it by the configured Report Schedule Actions.

See executeAndDistribute() in the appendix.

Scripting Functions

Watch the Video

Next...

- Charts in Reporting - Scripting

Charts in Reporting - Scripting

The most common configuration options for charts are found in the properties panel, but almost every part of a chart can be configured through scripting. To turn on configuration scripting in a chart, select the chart, scroll to the bottom of the Chart Options tab, enable scripting and click on Edit Script.

Data Sources - Scripting

Watch the Video

Example

Say we want to change the font used in a charts labels. We can use the following code:

Configure Chart

```
def configureChart(chart):
   # This configures the legend on a report chart to use a large, bold, 
monospaced font
   from java.awt import Font

   legend = chart.getLegend()   # http://www.jfree.org/jfreechart/api
/javadoc/org/jfree/chart/JFreeChart.html
   font = Font(Font.MONOSPACED, Font.BOLD, 48)   # https://docs.oracle.com
/javase/7/docs/api/java/awt/Font.html
   legend.setItemFont(font)   # http://www.jfree.org/jfreechart/api/javadoc
/org/jfree/chart/title/LegendTitle.html
```

The above code would change the font used in the chart's legend to a 48 point bold monospaced font. The comments give links to the APIs used in each line of the code.

Next...

- Converting Legacy Reports

Reporting Tutorials

This section contains some Tutorials that will help get you started in achieving some common tasks in the Reporting Module.

Tutorial: The Report Workflow

Steps to Amazing Reports

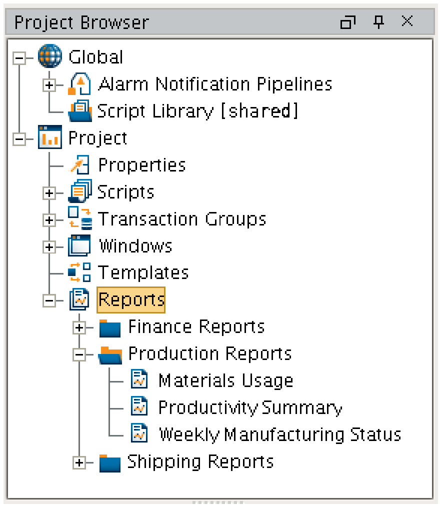
There are 3 fundamental steps to creating a Report in Ignition 7.8:

1. Gather your Data
2. Design your Report
3. Schedule Delivery

Simple, right? This workflow is really the gist of what it means to create a report in the Reporting Module. Conveniently, the Report Workspace interface makes it simple to complete each step and view your result. This tutorial will provide a walkthrough of creating a simple report, and reinforce some of the concepts and terminology involved. If you are already familiar with the basics of how to create a Report, you may be interested in the Rep orting Technical References. If you are in the right spot, lets go ahead and get started.

Your First Report

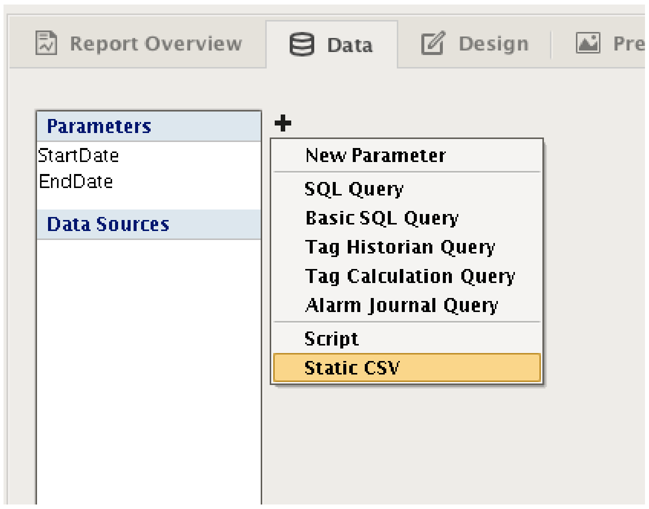
To create your first report, you will need an Ignition 7.8 installation, as well as the Reporting Module. We recommend following along with the core modules installed. If your installation is configured and Gateway running, launch the Ignition Designer and right-click on the `Reports` node in the Project Browser to create a new Report and open the Report Workspace.



With a new report open, you can take a look at the new Report Workspace. The top of the workspace has a series of stylized tabs which guide you through the creation of a new report. It's no coincidence that our Report Workspace is laid out in a way that lets you intuitively follow our 3 Steps with a little help along the way. There isn't much to see on the *Overview* of a new Report, so click over to the *Data* tab to get started.

Set Up Data Sources

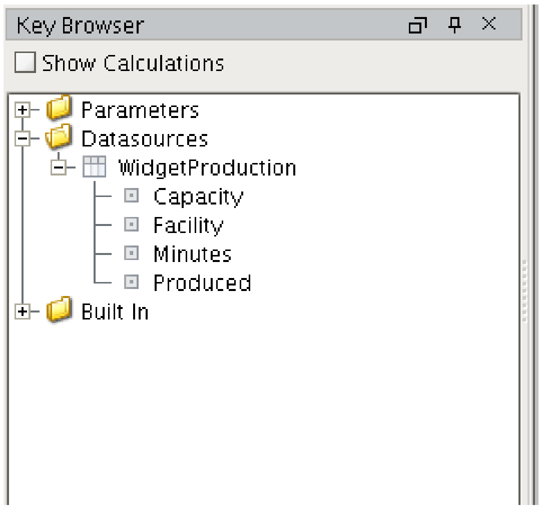
For any Report, you will first need to specify one or more Data Sources. Click the + button to add a data source. We'll start by adding a simple Static CSV source, but if you have a database connected with a table you'd like to use for the tutorial, feel free to add a SQL Query (a simple `SELECT * FROM <YOUR_TABLE_NAME>` is all that is needed to pull in all the data a table contains).



You'll now have a Static CSV Data Source editor open. Lets delete the example values and add our own. We are going to model data collected from international Widget Factories, complete with production capacity, the number of Widgets produced, and the number of minutes it took to produce them. Copy and paste this CSV data into your data source editor and give your data a meaningful name in the *Data Key* field.

```
Facility, Capacity, Produced, Minutes
"California", 2000, 665,  345
"Texas", 3424, 1674, 924
"South Africa", 734, 232, 154
"Brazil", 1131, 882, 325
"China", 5324, 2764, 297
"Norway", 436, 143, 383
"Kenya", 1431, 423, 164
"Italy", 543, 524, 234
"Romania", 154, 78, 45
"Peru", 624, 523, 732
```

Now switch over to the Design tab, and take a look at your Key Browser panel on the left. Expand the Data sources and you'll see that each column of data is represented by its own Data Key. These Data Keys are automatically generated based on the table of data fed to the report system and are *child-keys.* Accessing the data from a child key is accomplished using the path of a key in relation to its parent, sometimes referred to as *keychains*. A keychain is simply the path to your data key using 'dot notation'.   For instance, the keychain for our *Facility* key would be *WidgetProduction.Facility*.



Now that you have some data, lets create simple design and see how our data keys can be used to create dynamic reports.

Design the Report

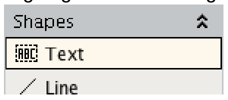
Designing a report is the act of deciding how you want the end result to look, and laying out the pages to accomplish your vision. Each type of Design component has strengths and weaknesses in how it presents data. You may want a line-item table to look at sales of each product, or a Pareto Chart to quickly identify high-impact targets for next year's mining investment. To create comprehensive reports, you will likely find yourself combining different components. Fortunately, this is pretty easy to do using the visual designer, so lets continue our first report and see how easy it is.

For our report, we'd like to have:

1. A Header
2. A Table that gives us a summary of how many Widgets each factory produced, and the totals for each data column.
3. Factory efficiency on a units/minute basis, which we will need to calculate.
4. Page numbers will be helpful if our report gets too long, so we'll add those too.

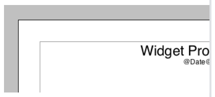
The table will offer a lot of detail, but we know our Widget Management doesn't do well with numbers, so we'd also like to give them a chart that shows the production of each facility.   To really impress, we'll use the new Pareto chart so we can point out which factories are accounting for 80% of our production.

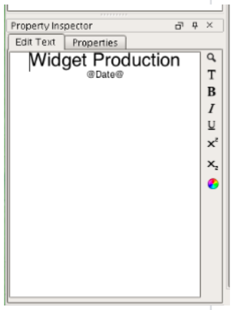
Add A Text Header for the Page

With an idea of what we want to accomplish, as well as the data to do it, we can start designing. To start we'll give our report a header with a title. We use Text shapes anywhere we need to add text to the page. Activate the Text Shape 

and click-drag near the top of the page to create a field for text. Go ahead and give your report a title by typing into the field of the shape. You'll notice the text field in the lower left corner of the Designer, along with some buttons that let you customize the look and layout of the text. This configuration area will change depending on the selected component on the report. In the same area, you'll notice there is a tab titled "Properties" behind the Text Area's custom editor.   This table provides access to a component's various properties. Feel free to experiment with the settings in the editor or property table to customize your title.

Text Shape Configuration





In addition to a title, providing some meta-data about the report can provide context or make it more readable. For instance, adding the date the report was generated is a very common task. However, we don't want to type the date into the text field, because all reports we run (could be today, tomorrow or next year!) will show what we type. Instead, we want the date to reflect the day the report was generated. If you look at your key browser, you'll see a folder called "Built In". This folder provides a number keys that are common in reports. Go ahead and grab the "Date" key and drop it in the same text field as your title.

You'll notice that the key you dropped into the text field is surrounded by '@' symbols. The @ symbols tell the report engine that text inside (in this case Date) is a key, and it should try and find value when the report is generated.   If you click over to the Preview tab, you should see today's date where `@Date@` is. In addition, you'll see that the text is in the same place it was in the designer. As our report gets more complex, we'll want to think about the layout of the items on the page to accomplish the look we want.

Adding text on a page shared with a repeating component (such as a table) will add the text to all pages created by the component. In some cases, like page numbers, this is desirable. In others, you may only want a single header for your table. Tables themselves have headers, which we'll briefly talk about, but you can read about them in more depth in the Reporting Table documentation.

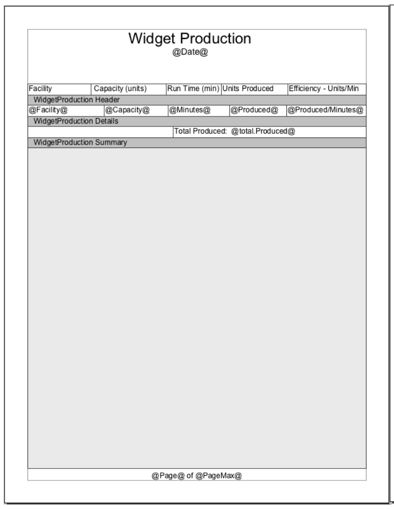
The Power of Data Keys

You can have multiple keys in a text field. In addition, you can use multiple keys and common numeric operators to do calculations *within* the @ symbols! These are called *Keychain Expressions*. Expressions are not something this introductory tutorial will cover, but you can head over to the data key documentation page for more information.

Adding a Table

Grab a Table component off the palette and add it to the page. With the Table selected, look in the bottom left corner under the Key Browser, and you will see the configuration panel for the Table. We want the Table to show us data about our Widgets Factories, so go ahead and grab the parent Data Key from the browser, and drag it to the Data Key field of the Table Config Panel. You'll notice that the Data Key field updates, as does the Table itself. Our WidgetProduction data key is now providing data for the table, so lets set up our table.

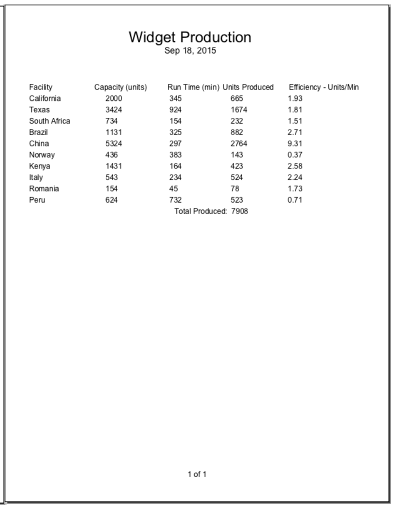
We want a header and summary, so check those boxes and lay out your table however you like. To calculate the efficiency in terms of Units per Minute of runtime, we use the keychain expression `@Produced/Minutes@`.   Here is what we came up with, and the generated report.



Widget Production
@Date@

Facility	Capacity (units)	Run Time (min)	Units Produced	Efficiency - Units/Min
WidgetProduction Header				
@Facility@	@Capacity@	@Minutes@	@Produced@	@Produced/Minutes@
WidgetProduction Details				
		Total Produced: @total.Produced@		
WidgetProduction Summary				

@Page@ of @PageMax@

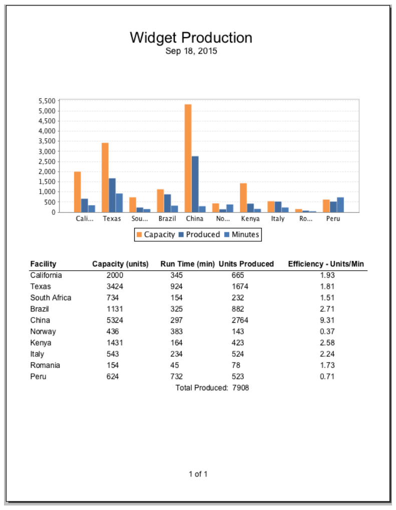


Widget Production
Sep 18, 2015

Facility	Capacity (units)	Run Time (min)	Units Produced	Efficiency - Units/Min
California	2000	345	665	1.93
Texas	3424	924	1674	1.81
South Africa	734	154	232	1.51
Brazil	1131	325	882	2.71
China	5324	297	2764	9.31
Norway	436	383	143	0.37
Kenya	1431	164	423	2.58
Italy	543	234	524	2.24
Romania	154	45	78	1.73
Peru	624	732	523	0.71
		Total Produced:	7908	

1 of 1

To finish the report, we just need to add a chart, we can just resize the table a little bit and add our chart right under the page header. Drag the bar chart into the space above our table, size it and the table to suite. Add the Data Key, for our data and take a look at the preview. Instantly, you'll notice the bar chart adds a bar for each column in the data, where our goal is to only show the number of widgets produced.



Widget Production
Sep 18, 2015

Facility	Capacity (units)	Run Time (min)	Units Produced	Efficiency - Units/Min
California	2000	345	665	1.93
Texas	3424	924	1674	1.81
South Africa	734	154	232	1.51
Brazil	1131	325	882	2.71
China	5324	297	2764	9.31
Norway	436	383	143	0.37
Kenya	1431	164	423	2.58
Italy	543	234	524	2.24
Romania	154	45	78	1.73
Peru	624	732	523	0.71
		Total Produced:	7908	

1 of 1

Edit Data Sources to Accommodate Requirements

If we were using Query data sources, we'd create an additional source use a nested source to get the two rows we want. For this example, we'll go ahead and create an additional data source using another static CSV source that includes only the columns we are interested in.

FacilityProduced Data Source

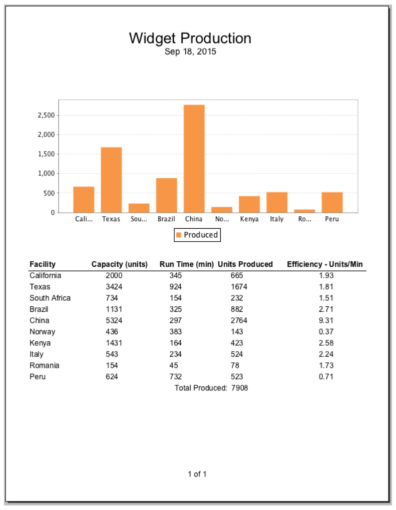
```
Facility, Produced
"California",665
"Texas", 1674
"South Africa", 232
"Brazil", 882
"China", 2764
"Norway", 143
"Kenya", 423
```

```
"Italy", 524
"Romania", 78
"Peru", 523
```

Now when we generate our report with the new data, we get a chart with the items produced, and a table summary of the different locations.

Final Product

We configured our bar chart to use the new data source, and got a result that seems to meet our goals.


Widget Production
Sep 18, 2015

Facility	Capacity (units)	Run Time (min)	Units Produced	Efficiency - Units/Min
California	2000	345	665	1.93
Texas	3424	924	1674	1.81
South Africa	734	154	232	1.51
Brazil	1131	325	882	2.71
China	5324	297	2764	9.31
Norway	436	383	143	0.37
Kenya	1431	164	423	2.58
Italy	543	234	524	2.24
Romania	154	45	78	1.73
Peru	624	732	523	0.71
		Total Produced:	7908	

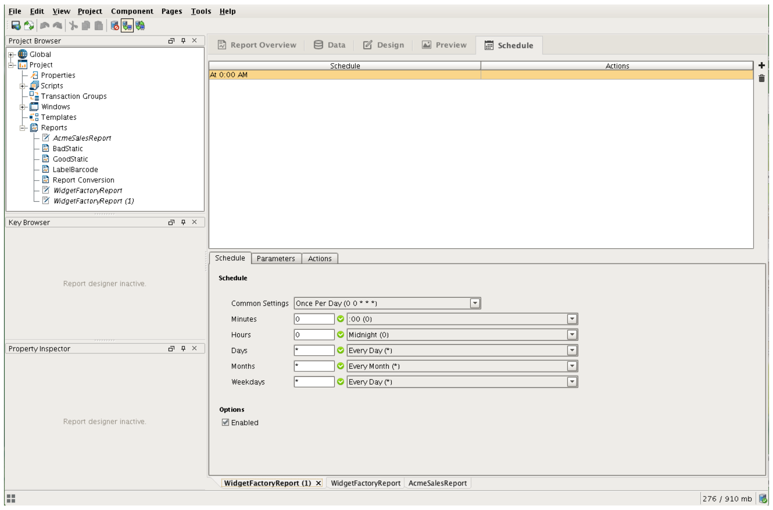
1 of 1

Scheduling Reports

Last by not least, we finish by setting up our report to run on a schedule. Scheduled Reports are easy to setup and allow you to define a number of different Actions to occur on a schedule. For our schedule, we simply want to create a PDF of the report and save it to a local folder at the end of the work week.

Adding a Schedule

Adding a schedule is as easy as clicking on the ✚ button of the Report Schedule panel in our workspace.



Schedule	Actions
At 0:00 AM	

Schedule

Common Settings	Once Per Day (0 0 * * *)	
Minutes	0	:00 (0)
Hours	0	Midnight (0)
Days	*	Every Day (*)
Months	*	Every Month (*)
Weekdays	*	Every Day (*)

Options
☑ Enabled

With the schedule added, use the dropdown options to determine when it should run. We set ours to run on the 5th weekday (Friday) at 23:59, or one minute before midnight. Keep in mind, this report will execute on our Gateway, so the schedule will be according to Gateway Local Time. We didn't use parameters in this report, so we can click over to the Actions tab to set up the actions for this Schedule. Here we will click the ✚ next to the empty Action list, and select Save File. That presents us with Folder Path and FileName configuration space. We change the path from default to something that works for our system. Once happy with the settings, you can test run clicking the ⏩.

Congratulations! You've completed your first report!

Sequential Function Charts

A Sequential Function Chart (SFC) uses a graphical programming language based on the IEC 61131-1 standard. This language may be familiar to PLC programmers, as it is one of the languages commonly available for programming PLCs.

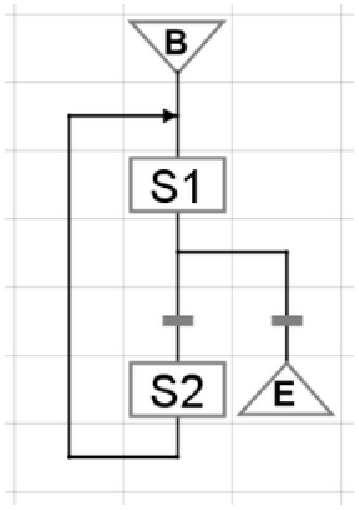
SFCs are used to execute logic in ways that are more convenient to structure than with Python scripts or PLC programming alone. Because of their inherently visual depiction, they help to illuminate logic to users, and facilitate intuitive development and refinement. Charts can be monitored as they run visually, making troubleshooting easier than with scripting alone.

SFCs simplify complex task like running tasks in parallel, writing long-running or slow tasks, and structuring tasks that must wait for conditions before proceeding.

A single SFC in Ignition can be called multiple times resulting in the same logical engine, the SFC, influencing different tags. This allows the developer to recycle the logic to different parts of the project or to run similar systems with a single SFC without having to duplicate the SFC.

The Ignition Designer allows for managing the SFC as it runs. Simple HMI interfaces can be developed to manage the SFC. An SFC can be started with a simple button or it can be fully managed with the SFC Monitor component.

An Example SFC



In this section ...

SFC Basics

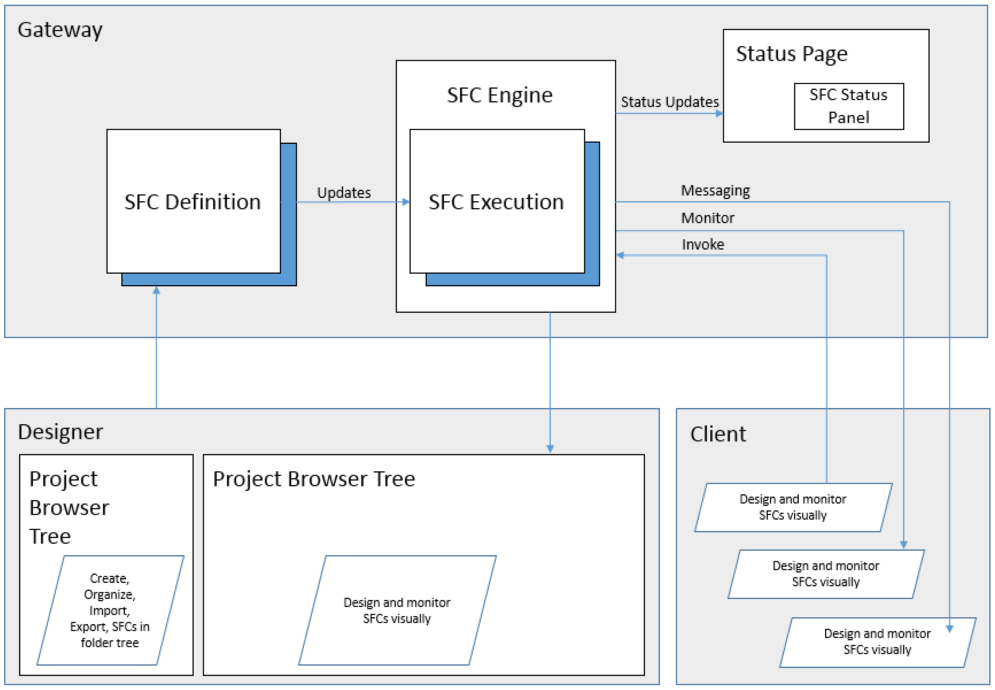
Architecture

SFCs are *designed* through drag-and-drop manipulation in the Designer. The charts can be found under the **Global** folder in the Designer's **Project Browser**. Charts are not part of any specific project; they are shared by all projects.

SFCs are *executed* in the Gateway. While any scope (Client, Designer, or Gateway) can start a new chart instance, the chart instance always executes in the Gateway.

SFC instances can be *monitored* in either the Client or the Designer. To monitor a running chart instance in the Designer, you can open that chart in the SFC workspace and double-click on the running instance. To monitor a running chart in the Client, a Vision project must be designed that uses the SFC module's monitoring panel component.

SFC Execution in Gateway


Gateway
Status Page
SFC Engine
Status Updates
SFC Status Panel
SFC Definition
Updates
SFC Execution
Messaging
Monitor
Invoke
Designer
Project Browser Tree
Project Browser Tree
Create, Organize, Import, Export, SFCs in folder tree
Design and monitor SFCs visually
Client
Design and monitor SFCs visually
Design and monitor SFCs visually
Design and monitor SFCs visually

Scripting Reference

For information about the scripting API methods available for sequential function charts, see Scripting Functions.

In this section ...

Introduction to Sequential Function Charts

Each sequential function chart can be configured to use one of the following *Execution Modes*:

- **Callable**
  This chart can be started via scripting or another chart's enclosing step on-demand. Any number of instances of this chart can be simultaneously running.
- **RunAlways**
  This chart can be started by the Gateway upon startup. It can not be executed in any other way. It is probable that this chart be designed to never end, the idea being that there will always be exactly one instance of this chart running.
- **Disabled**
  This chart is not available for execution.

INDUCTIVE UNIVERSITY

Introduction to Sequential Function Charts

Watch the Video

Next ...

- Chart Elements

Chart Elements

Chart Elements

Watch the Video

Next...

- Designing Charts

Designing Charts

Proper Structure

Before a chart is executed, it must have a correct structure. When designing a chart, the Designer will constantly let you know whether or not your chart is valid or not. If your chart is not valid, you may choose to show the errors. The error shows up as red triangles in the corner of any element which has a problem. Hover your mouse over these elements to discover what is wrong with them.

Here are some rules about structure to keep in mind:

- Everything must be fully connected (except for Notes).
- Flow typically moves from top to bottom. All elements must be entered from the top and exit from the bottom (not the sides).
- There can be any number of end steps. If flow reaches an end step, the chart is stopped. If there are no end steps, it means that your chart must loop back upon itself to satisfy the connected rule.
- End steps are not allowed inside parallel sections.

INDUCTIVE UNIVERSITY

Designing Charts

Watch the Video

Next ...

- Chart Flow Steps and Transitions

Chart Flow Steps and Transitions

INDUCTIVE UNIVERSITY

Chart Flow Steps and Transitions

Watch the Video

Next ...

- Chart Scope

Chart Scope

Chart Scope Variables

Certain chart scoped variables may interfere with the internal functions of the chart. For example, creating a variable like chart.values will conflict with a pyDictionary's values() method and therefore the chart will show an error. Since SFC charts use Python Dictionaries to manage chart scoped variables the methods associated with Python Dictionary's act like reserved words.

There are a number of built-in variables maintained by the SFC engine that can be read through the `chart` scope.

SFC built-in Variables	Description
chart.instanceId	The string UUID of the running chart instance
chart.startTime	A `java.util.Date` object that indicates when the chart instance started running.
chart.runningTime	An integer representing the number of seconds the chart has been running for.
chart.parent	The chart scope of the enclosing chart (if any). null if this chart was not executed as part of an enclosing step.
chart.running	Returns true if the chart is in the running state
chart.state	An integer representing the state of the chart as the following: 0 Aborted 1 Aborting 2 Cancelled 3 Canceling 4 Initial 5 Paused 6 Pausing 7 Resuming 8 Running 9 Starting 10 Stopped 11 Stopping

Chart Scope

Watch the Video

Scripting system.sfc.* Functions

See the following links for a description of each function:

Next ...

- Chart Rules

Chart Rules

Chart Rules

Watch the Video

Next...

- Chart Properties

Chart Properties

Chart Properties

Watch the Video

Next ...

- Chart Lifecycle

Chart Lifecycle

The chart has three terminal states: stopped, aborted, and canceled, and each has a different meaning:

- **Stopped**
  This means the chart stopped normally by reaching an End Step.
- **Aborted**
  This means the chart stopped abnormally because of an error.
- **Canceled**
  This means that the chart was canceled by a user or a call to `system.sfc.cancelChart()`.

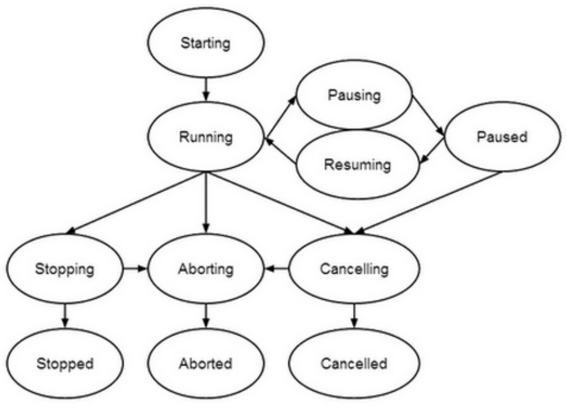
The chart can have scripts configured to run during some transitions between states.

- **onStart**
  This script runs before the chart moves to the Running state.
- **onStop**
  This script runs in the Stopping state before the chart moves to Stopped state.
- **onAbort**
  This script runs in the Aborting state before the chart moves to Aborted state. If this script creates an error, it is logged, and then the chart moves to Aborted state.
- **onCancel**
  This script runs in the Canceling state before the chart moves to Canceled state.

Chart Lifecycle

Watch the Video

The figure shows the states a chart can be in, and how it flows from one state to another



Next ...

- SFC Elements

SFC Elements

The Chart

The chart itself is a grid of cells upon which elements are placed. The chart has some configuration that can determine how and when the chart is started up, as well as opportunities to respond to chart lifecycle events with scripting, such as onStart, onStop, onCancel, and onAbort, see Chart Lifecycle for details.

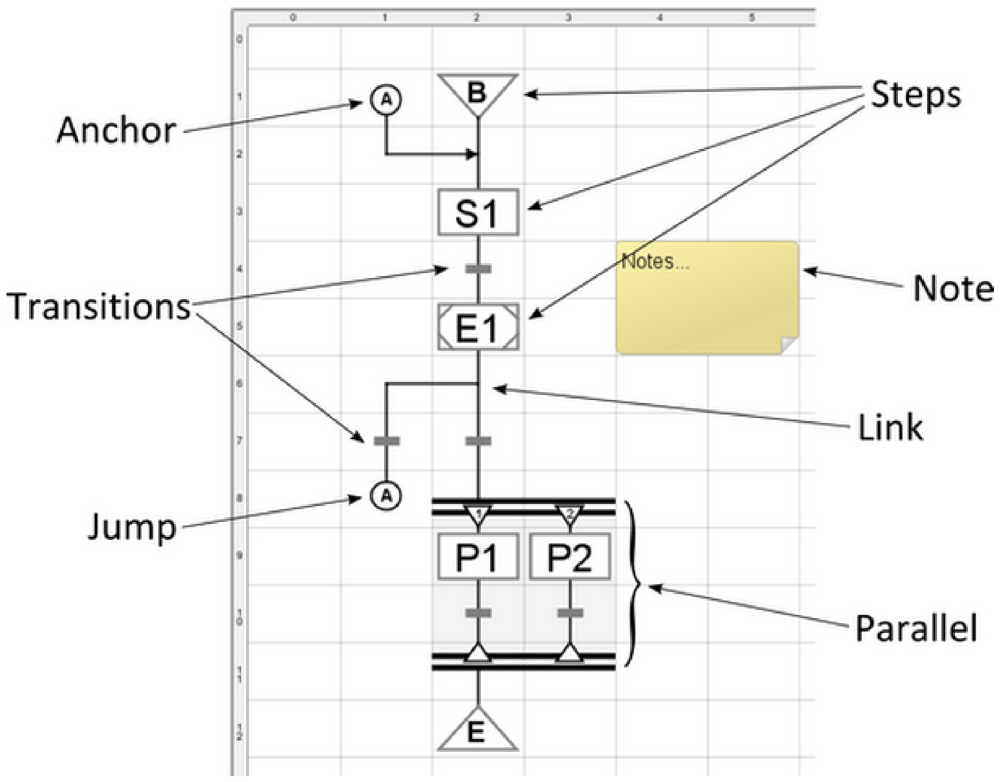
Steps

Steps are the parts of the chart that do useful work. Steps are represented as a rectangle that occupies a one-cell region of the chart, except for the begin and end steps, which are triangles. Steps might run scripts, or execute other charts.

There are four types of steps:

- Begin Step
- End Step
- Action Step
- Enclosing Step

SFC Chart Elements



In this section ...

Begin Step

The **Begin Step** is where all charts start, and cannot be removed, cut, or copied. The begin step is where you can define initial values for your chart's scope. These initial values are also hints as to what parameters your chart expects. If the chart receives any of these parameters as starting parameters, the initial values are ignored.

One of the chart parameters defined on the begin step can be marked as the **Key Param**. This means that this parameter is used as an identifier for the chart. For example, suppose your chart defined the automation process for a car through an assembly line. You can define the car's VIN as the key param, which means that instances of this chart are identified by the VIN they were started with.

Begin Step

Watch the Video

Next ...

- End Step

End Step

The **End Step** of a chart has no configuration. This is used to mark the termination of the chart. There can be many end steps in a chart, although only one is ever reached for a given chart instance. End steps are not allowed inside parallel sections.

End Step

Watch the Video

Next ...

- Action Step

Action Step

The action step can have any number of scripts associated with it. Each of these scripts is called an action. There are various kinds of actions. The different kinds of actions execute at different times in the step's lifecycle. During the lifecycle of the step, many actions can run, but only one action is ever allowed to run at a time.

The scripts configured in the action step have access to the chart's scope, as well as a special scope that is private to the step. This scope, called "step scope" is initialized when the step starts and destroyed when the step stops, but retained while the step is running. This is a good place for timer actions to store intermediate results.

INDUCTIVE UNIVERSITY

Action Step

Watch the Video

On Start

This action runs when the step is started. These actions always run to completion before flow can move beyond the step, and before any other scripts on the action step will run.

On Stop

When flow is ready to move beyond the step, it is told to stop. The step will then wait for any currently executing action, for example, **On Start** or **Timer** actions, to finish. Then it will execute its **On Stop** action , if configured. Only after these actions complete will the chart be allowed to move on.

Timer

Timer actions run every so often while the action step is running. The timer actions only start running after the **On Start** action finishes (if there is one). In addition to the timer action's script, it must have a rate, specified in milliseconds. This is the amount of time to wait between running the timer action's script. The clock starts after the **On Start** action finishes.

It is important to realize that, unlike **On Start** and **On Stop** scripts, a timer action can not run at all for an action step. If the step is ready to stop before the timer is ready, it will never run.

Error Handler

This action will run only if any of the other actions throw an unexpected error. This provides a chance for chart designers to do their own error handling, and gracefully recover from an error. If this script throws an error, the chart will abort, see Chart Concepts for more information.

Design Tips

When designing SFCs, the Action step will be the primary workhorse of your charts. There are a few tips to keep in mind while writing your scripts for the action step:

Tip 1: Don't Block/Pause/Wait/Sleep
This is the primary rule for SFCs. You want your scripts to run as quickly as possible. This means that you don't want to use any sort of sleep() or wait() call. Pausing or waiting is the job of transactions in an SFC.

Some sorts of blocking is, of course, unavoidable. For example, running a SQL query that takes some time to execute, or writing to a device over a slow radio connection. These sorts of activities are fine, however, this brings us to tip 2.

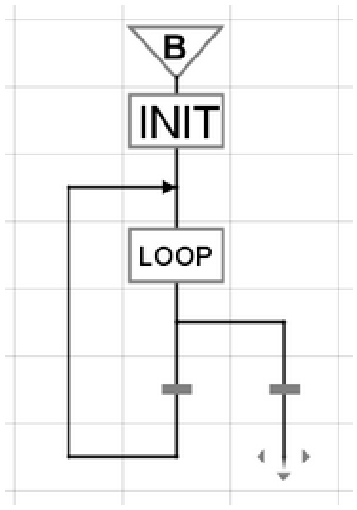
Tip 2: Refractor Loops
This is really just an extension of the don't block rule. Imagine that you have 100 widgets that need processing. Each widget takes some non-trivial amount of time, let's say, 20seconds, to process. The most obvious way to handle this would be with an **On Start** script that had a `while` loop from 1 to 100, processing each widget. This script would take about 33 minutes to run.

Instead of processing the 100 items in a `while` loop, you can solve this problem in two different ways with SFCs:

- Option 1: Timer Action
  In this option, you have a single action step, but instead of having your loop from 1 to 100 in a `while` loop, you initialize a counter to 1 in the On Start action. Then you write a timer action with a rate of 0ms. The timer action processes one widget, and then increments the counter. You place a transition beneath the step whose expression is: `{counter} >= 100`

- Option 2: Chart Loop
  Similar to option 1, you can design the loop in the chart itself. Have one action step do your initialization work, in our example: `chart.counter = 0`

Have another step do the processing work for one item in its On Start script. Use two transitions, the one on the left set to `{counter} < 100` and the one on the `right` set to `true` . The loop action will run 100 times, and then the flow will continue down the other path.



Option 2 Illustrated

Rationale

By now you should understand that you want to keep your individual script duration to a minimum. You may be wondering why this is so important. After all, in the example above, it still takes 33 minutes to complete the given work after refactoring the loop as shown.

The reason this is important is to support pausing, persistence, and redundancy. Charts can only be paused after any running script finishes. Similarly, a chart's state only gets synchronized with the redundancy system between script executions. If you had a script that took half an hour to run, you couldn't pause that chart until the script ended, and the redundancy system would be very out of date and not able to recover as well if the Gateway went offline.

As a bonus, breaking your work up into smaller units helps make the chart easier to debug and more visible for monitoring.

Next ...

- Transition

Transition

Transitions control the flow of the chart. A transition serves to either block or allow flow, depending on its value. All transitions have a value: true or false, which is determined by an expression. Transitions occupy a one-cell region of the chart, and are represented by a short horizontal bar in the middle of the cell. Read more about how transitions control chart flow in the Chart Flow Steps and Transitions section.

In addition to their expression, a transition can also specify a timeout. If enabled, the transition will set a flag after a certain amount of time has passed. This flag is a boolean variable set in chart scope, and can be used to make the expression or another expression close.

INDUCTIVE UNIVERSITY

Transition

Watch the Video

Links

A link is simply a line that connects other elements. Links are created in the Designer by dragging the arrows that appear next to unconnected elements. Links cannot cross above or below other elements or links. Links only travel in a single direction. This direction is determined by what the link is connecting to. Most elements such as steps and transitions only accept incoming links from above and outgoing links from below.

Next ...

- Enclosing Step

Enclosing Step

The **Enclosing Step** references another SFC defined on the same Gateway. This is an important tool for SFC design, because it lets the chart designer create reusable blocks of logic encapsulated into charts, which can make chart design more modular.

When talking about enclosing steps, the chart that the enclosing step references is called its enclosed chart, or **subchart**. The chart that the enclosing step is in is called the **parent chart**.

When flow reaches an enclosing step, it starts its enclosed chart. Using the enclosing step's Execution Mode property, the step can be configured to work in one of two very different ways:

Execution Mode = Block
Let the enclosed chart run to completion. This means that the enclosed chart should have an **End Step** in it, and that flow will not be able to move beyond the enclosing step until the enclosed chart stops by reaching its end step.

Execution Mode = Cancel
Cancel the subchart when the enclosing step is ready to stop. This means that the subchart is canceled when flow is ready to move beyond the enclosing step. Any running steps in the enclosed chart are told to stop, and flow ceases in the enclosed chart.

INDUCTIVE UNIVERSIT

Enclosing Step

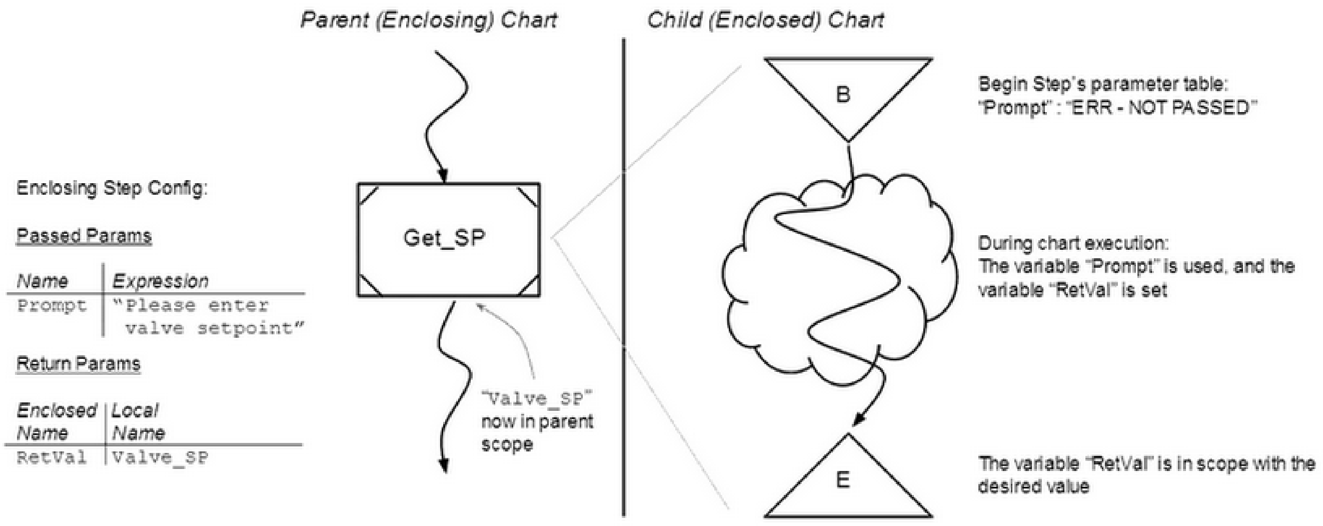
Watch the Video

Parameter Passing

When invoking a subchart via an enclosing step, you have the opportunity to define how variables are passed and returned between the parent and child chart's scopes.

The enclosing step can define a list of parameters to be passed into the enclosed chart's scope. The values for the parameters will be expressions, thus they can be literal values or they can be references to variables in the enclosing chart's scope.

The enclosing step can also define a list of **return values** to receive from the enclosed chart. This is a mapping of variable names from the enclosed chart's scope to variable names in the parent chart's scope.



Parent (Enclosing) Chart
Child (Enclosed) Chart
Enclosing Step Config:
Passed Params
Name
Expression
Prompt
"Please enter valve setpoint"
Return Params
Enclosed Name
Local Name
RetVal
Valve_SP
Get_SP
"Valve_SP" now in parent scope
B
E
Begin Step's parameter table: "Prompt" : "ERR - NOT PASSED"
During chart execution: The variable "Prompt" is used, and the variable "RetVal" is set
The variable "RetVal" is in scope with the desired value

Enclosed Chart Parameter Passing

Next ...

- Parallel Section

Parallel Section

A parallel section is a rectangular section of the chart that can contain other chart elements inside it. The top and bottom of the parallel section is demarcated by two thick, parallel lines. The top lines are called the "parallel branch" and the bottom are the "parallel sync". These sections are used to execute multiple branches of the chart at the same time.

Parallel Section

Watch the Video

Next ...

- Jump and Anchor

Jump and Anchor

A jump is an element that moves the flow to its matching anchor. This is a convenience for when a link would be unsightly or impossible due to crossing other links. Each jump and anchor element is identified by a single character; jumps identified by **X** will jump to the anchor also identified by **X**. There can be many jumps on a chart that all jump to the same anchor.

Jump and Anchor

Watch the Video

Next ...

- Note

Note

Notes are elements which have no import or function, but serve as documentation. Note elements can be placed anywhere except that they can not overlap other elements.

INDUCTIVE UNIVERSITY

Note

Watch the Video

Next ...

- SFCs in Action

SFCs in Action

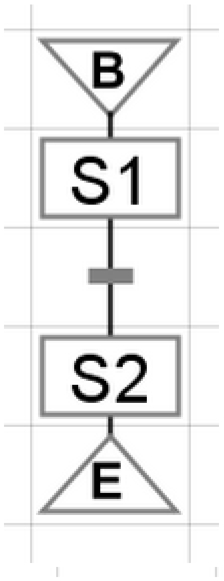
Chart Flow

All charts start at their begin step. The begin step can define initial values for variables in the chart's scope. These initial values are defined as expressions.

Flow always moves downward out of chart elements, except for links, which can move flow in any direction.

When flow hits a step, that step is started. The step continues to execute until the transition beneath it becomes true. If there is no transition beneath a step, the step starts and is told to stop as soon as possible. In practice, this means that an action step's onStart and onStop scripts will be run, but no timer scripts.

Examples

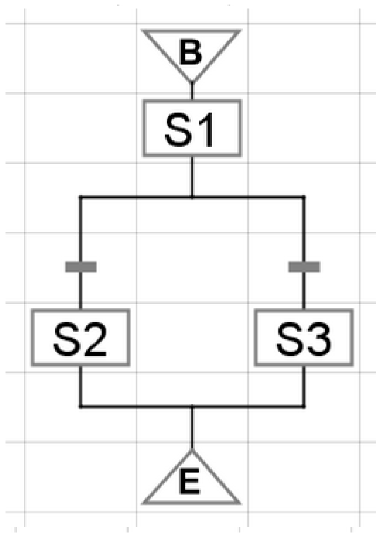


Example 1

In this example, step S1 executes as soon as the chart starts, and continues executing until the transition beneath it becomes true. Once that transition becomes true, Step S1 is told to stop, which means it finishs executing any scripts that are currently running, and then it executes its onStop action (if any).

After S1 has stopped, step S2 starts. It is immediately told to stop, which means that if it has any timer actions, they will not run, but the start and stop actions will run.

After S2 is finished, the chart stops.



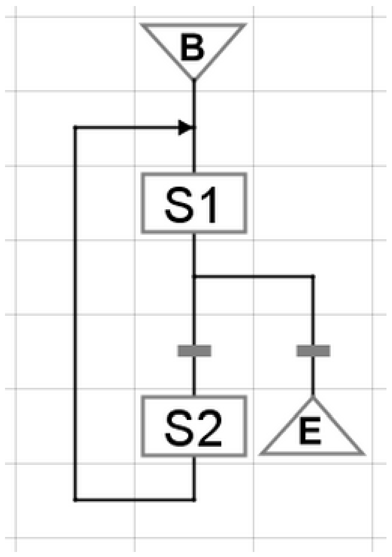
Example 2

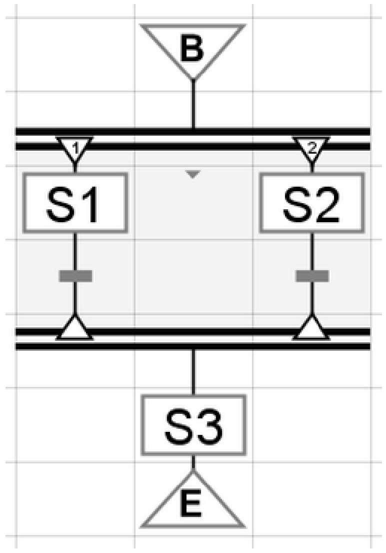
In this example, step S1 executes as above, except that it has two transitions beneath it. This is how you do conditional logic in a chart. S1 runs until either of these transitions becomes true. When one transition becomes true, flow will follow that branch of of the chart. If both transitions are true, the transition on the left is chosen. Position is meaningful for charts - transition precedence goes from left to right.

Only one of S2 or S3 will run, but never both.

Example 3

In this example, S1 executes as above, but can execute multiple times. This is how you configure repeating logic in a chart. The two transitions determine whether this chart continues running (possibly indefinitely) or stops.





Example 4

In this example, steps S1 and S2 execute simultaneously. They both continue to run until the transitions beneath them become true.

Flow only moves past the parallel sync (the bottom of the parallel section) once both transitions become true. Step S3 then runs, and then the chart stops.

In this section ...

Starting a Chart

A chart can be started in one of four ways:

1. **From Scripting**
   Using the `system.sfc.startChart` method of the scripting API, a chart can be started from anywhere. The chart must be in **Callable** execution mode.

2. **From an Enclosing Step**
   A chart can spawn an instance of another chart using an Enclosing Step.

3. **Automatically**
   A chart whose execution mode is **RunAlways** is automatically started when the Gateway starts up. If the chart stops, the Gateway does not re-execute it. If you want a chart that runs all the time, it should be designed to never stop, for example, by looping back upon itself continuously.

4. **From the Designer**
   While designing a chart, you can start instances of it from the **Chart Control** panel in the Designer using the **Start** link.

Starting a Chart

Watch the Video

Next ...

- Monitoring Charts

Monitoring Charts

While a chart is running, it can be monitored visually in the Designer or via a Vision Client.

In the Designer

Open the chart you wish to monitor, and any running instances of that chart will appear in the list to the right of the design space with the heading **Chart Control**.

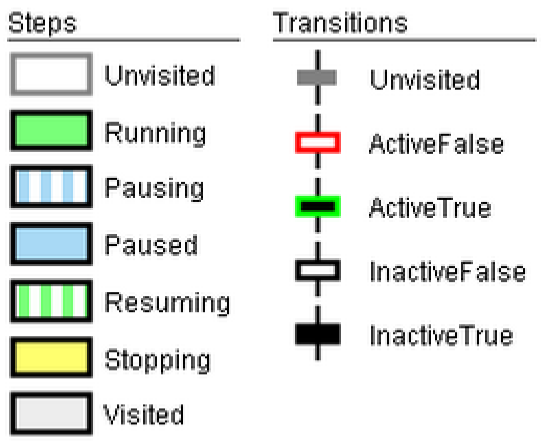
Double-click on an instance to enter monitoring mode. While in monitoring mode, you'll view the current state of the chart elements. There is a banner at the top of the Designer which will bring you back to design mode.

In a Vision Client

The SFC module adds a component to the Vision module under the **Admin** category called the **SFC Monitor**. Add this component to a window to be able to monitor SFC instances from your project.

This component can either display a pick-list on its left side to pick which instance to monitor, or you can give it the ID of a specific chart to monitor and hide the pick list.

Element Legend



Monitoring Charts

Watch the Video

Next ...

- Pause, Resume, and Cancel

Pause, Resume, and Cancel

Pause, Resume, and Cancel

Watch the Video

Notes ...

- Interacting with a Client

Interacting with a Client

INDUCTIVE UNIVERSITY

Interacting with a Client

Watch the Video

Notes ...

- Action Step Best Practices

Action Step Best Practices

Action Step Best Practices

Watch the Video

Notes ...

Enterprise Administration

The Enterprise Administration Module allows you to create a single Controller that creates and manages Agents throughout your network.

In this section ...

Enterprise Administration Basics

Learn how to set up Controllers and Agents

In this section ...

Introduction to Enterprise Administration Module

Overview

The Enterprise Administration Module (hereafter abbreviated to EAM) allows a user to control and monitor any number of Ignition Gateways from a central Gateway. In EAM terminology, the central Gateway is called a controller, and the monitored Gateways are called agents. The EAM takes advantage of the Gateway Network system to seamlessly deliver data between Gateways.

Introduction to Enterprise Administration Module

Watch the Video

Task Management

EAM allows you to schedule common Gateway tasks to execute against remote agent Gateways. Tasks include agent Gateway restart, backup collection, system restoration, module installation, license key activation/unactivation, and project distribution. Tasks can be scheduled to run once at a later time, or as part of a recurring schedule.

Agent Events

Every EAM agent sends information to the controller about its general health status. Whenever problems occur on an agent, the controller is notified of the agent’s status in the form of agent events. Problems can include high CPU usage on the agent, large numbers of system errors in a short timeframe, and unusually high usage of database connections. Loss of connectivity to the agent also triggers an agent event on the controller. Agent events are recorded in an external database for analysis and reporting later. Agent events can also be directly configured in the controller to automatically issue alarms to a specified alarm pipeline.

Archival and System Recovery

The Collect Backup gateway task does more than simply collect a gateway backup from agents. Whenever the task runs, the gateway backup, all installed modules and the current license are collected from the agent and stored in the archive system on the controller. This system allows for quick recovery later if an agent machine catastrophically fails and must be reinstalled.

Next...

- Gateway Area Network Settings

Gateway Area Network Settings

Overview

The Gateway Network allows you to connect multiple Gateways together over a wide area network. The Gateway Network provides the following features:

- Websockets provide fast, firewall-friendly 2-way communication over a single configured connection.
- 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.
- SSL mode available. When enabled, connections must send SSL certificates to prove their identity. A connection will not be accepted until its SSL certificate is approved.

INDUCTIVE UNIVERSITY

Gateway Area Network Settings

Watch the Video

Currently, the Enterprise Administration Module (EAM) takes advantage of the Gateway Network. The 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. Many other features are planned to use the Gateway Network in the near future.

General Settings

Main

The Gateway Area 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.

Enabled: uncheck this checkbox to disable using the Gateway Area Network on this Ignition.

Web Sockets Enabled: If true, the system will attempt to use websockets in order to increase performance.

Require SSL: If true, only connections that use SSL to encrypt traffic will be allowed.

Require Two Way Auth: Controls whether both sides must explicitly trust each other before successful handshake. 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

Overload Control

If there is too much traffic on the GAN, then it can tell Agents to stop sending messages for a specified amount of time.

Receive Queue Max: Number of received messages that can be held until they are processed by the local system. When this capacity is exceeded, the local system will start sending WAIT messages to remote machines.

Overload Wait Seconds: The number of seconds that a remote machine is instructed to wait before resuming message delivery. This setting is used when an overload is detected on the local machine.

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, navigate to Configure -> Configuration -> Gateway Network and select the General Settings tab.

Allow Incoming Connections: 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.

Connection Policy:

- **Unrestricted**: default mode, allows all incoming connection 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 an ID in this list are automatically allowed if the security mode is set to SpecifiedList. Separate Gateway names with a comma.

Outgoing vs. Incoming Connections

When using the Gateway Network, you will be working with two type of connections. To establish communications, you 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. 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 Server A. The remote machine is known as Server B. Server B also knows about another remote machine named Server C. As soon as your local Server A establishes a connection with Server B, Server B also sends information about the existence of Server C. Modules such as the EAM are aware of this relationship and allow communication between Server A and Server C, even though there is no direct connection from the local machine to Server C.

Basic Setup

To establish a basic comm link between two Gateways, log into the Gateway where you want to establish the outgoing connection. Navigate to Configure -> Configuration -> Gateway Network and click on the Outgoing Connections tab. Click the “Create new” link.

On the New Outgoing Connection page, you will need to enter the network address and port of the remote machine. Click the Create New Connection button at the bottom of the page.

On the Outgoing Connections tab, your new connection will be shown. After a few moments, the Gateway Name field will be updated with the name of the remote server and the status will be set to “Running”.

To view the incoming connection, log into the other Gateway and navigate to Configure -> Configuration -> Gateway Network and click on the Incoming Connections tab. Your new connection will be shown with “Connected” status.

Deleting Connections

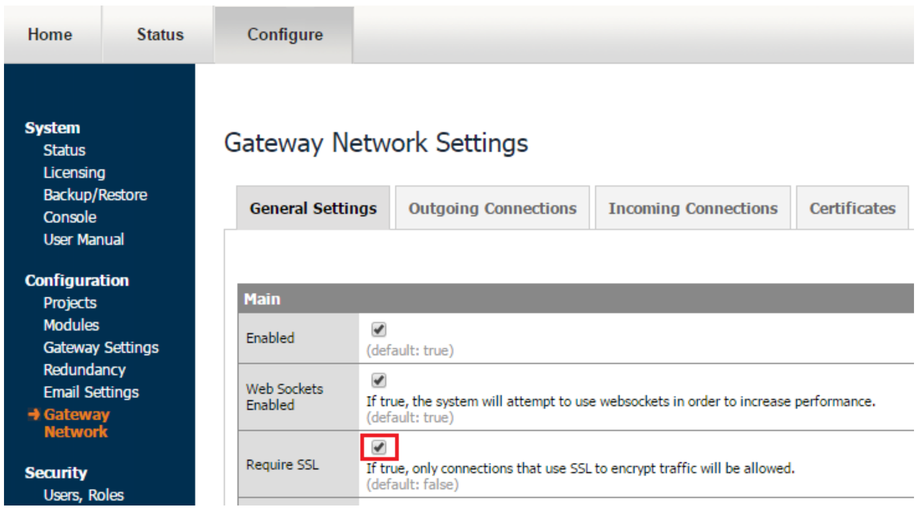
Outgoing and incoming connections can be deleted for cases when the connection no longer exists on the other side. To delete a connection, navigate to Configure -> Configuration -> Gateway Network and click on either the Outgoing Connections tab or the Incoming Connection tab. Then click the delete link 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.

Certificates

SSL and Certificates

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, there 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.

- To enable SSL on the Gateway Network, you must enable SSL in two places. On the machine that will receive incoming connections (which we will call Server A), you must navigate to Configure -> Configuration -> Gateway Network, click on the General Settings tab, and check the Require SSL checkbox. This mode requires all incoming connections to use SSL mode. Any existing incoming connections will become faulted as they are not using SSL. Click the Save Changes button.

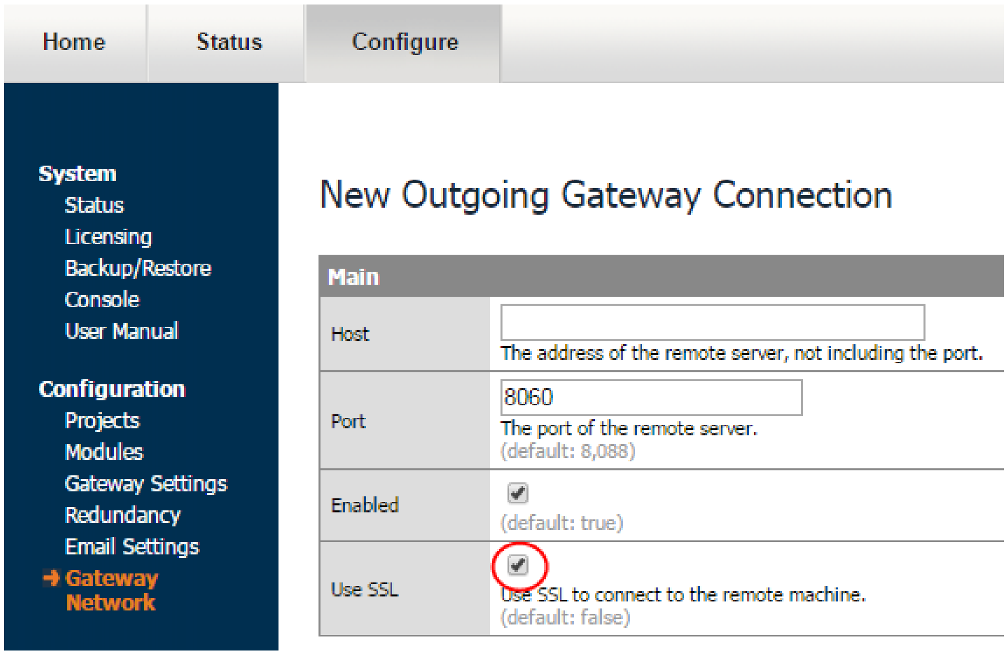


Gateway Network Settings

Main	
Enabled	✓ (default: true)
Web Sockets Enabled	✓ If true, the system will attempt to use websockets in order to increase performance. (default: true)
Require SSL	✓ If true, only connections that use SSL to encrypt traffic will be allowed. (default: false)

- Next, log into a Gateway where you wish to establish an outgoing connection. We will call this Server B. Navigate to Configure -> Configuration -> Gateway Network and click on the Outgoing Connections tab and click the “Create new” link. As you are entering the settings for the new connection, check the Use SSL checkbox. Also, change the Port to the SSL port used by the other machine. By default this is set to 8060 (which is defined /data/gateway.xml). Save the new connection.

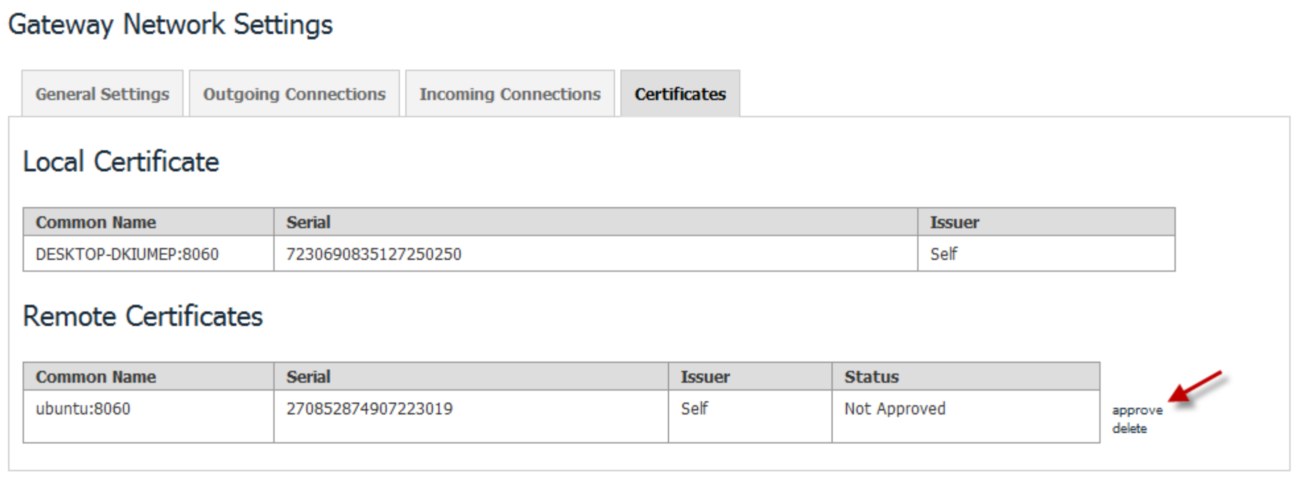
> Note that this port is different from the default SSL port an Ignition Gateway would use when communicating to a client (default port 8043).



New Outgoing Gateway Connection

Main	
Host	The address of the remote server, not including the port.
Port	8060 The port of the remote server. (default: 8,088)
Enabled	✓ (default: true)
Use SSL	✓ Use SSL to connect to the remote machine. (default: false)

- At this point, Server B has transmitted its certificate to Server A. But Server A has never seen this certificate, and so the incoming connection is not yet allowed. Note that the Server B’s connection will NOT show up under the Incoming Connections tab until AFTER the certificate has been approved. But if you click on the Certificates tab, Server B’s certificate should be present.


Gateway Network Settings

Local Certificate

Common Name	Serial	Issuer
DESKTOP-DKIUMEP:8060	7230690835127250250	Self

Remote Certificates

Common Name	Serial	Issuer	Status	
ubuntu:8060	270852874907223019	Self	Not Approved	approve delete

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. Click the approve link to approve the certificate. After certificate approval, the incoming connection will be accepted, and will eventually appear on the Incoming Connections tab. It make take up to 30 seconds for the connection to appear, as reconnect attempts are made every 30 seconds by default (controllable by the Missed Pings outgoing connection setting).

After a certificate has been approved, and the incoming connection is established, you can view the certificate information for a specific incoming connection under the Certificate field in the Incoming Connections tab. The certificate common name and serial number are displayed here, and reflect the actual certificate that was sent by the remote machine.

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 not 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 Area Network Technical Information

Web Sockets

The web socket protocol rides on top of the TCP protocol, and allows 2-way communications between machines. Either side can send a message to the other side at any time. If web sockets cannot be used on your network, or your network setup does not require a high-speed low-latency connection, the Gateway Network connections can use a pure HTTP long polling system for communications. This mode uses standard HTTP GET and POST messages for communication, and is appropriate for situations when an always-on connection is not practical. An example would be a Gateway that communicates once every 15 minutes over a cell connection. Message retrieval is not as efficient as web sockets, as regular polling of the remote machine is required to learn of new messages. To enable HTTP long polling, you must either disable web sockets under either Gateway Network General Settings (to disable for all connections) or disable web sockets on an outgoing connection.

Next...

- Agent Event Threshold Settings
- Creating a Controller
- Adding an Agent

Agent Event Threshold Settings

Overview

One of the main functions of the controller is to keep track of connected agents and report when an agent starts to experience performance problems, logs many errors in a short time, or unexpectedly goes offline. Agent events are held in the controller's configured database in the **agent_events** table. Within this table, you can find three types of event categories: **agent**, **metric**, and **task**. The agent event category is used for major connectivity events, such as loss of connectivity from an agent. The metric event category is used for events that report abnormal agent health statistics, such as abnormally high CPU usage. The task event category does not affect metrics, and is detailed in the Gateway Task Results section.

Agent Event Threshold Settings

Watch the Video

General Settings

Alarm Evaluation

This section is for activity and metric events, you can configure alarms to trigger when an event is reported at the warning or error level. You can also set the alarm pipeline that will process the generated alarms.

Enable Activity Alarms: If true, alarms will be generated for agent activity events, such as when an agent stops responding.

Enable Metrics Alarms: If true, alarms will be generated for agent metric events.

Warning Priority: The priority assigned to all of the **warning** thresholds. Options are: Diagnostic, Low, Medium, High, Critical.

Error Priority: The priority assigned to all of the **error** thresholds. Options are: Diagnostic, Low, Medium, High, Critical.

Active Pipeline: The Pipeline assigned to all of the **warning** thresholds. Note that this Pipeline must be created before the alarm event happens, and that the name is case-sensitive.

Ack Pipeline: The Pipeline assigned to all of the **error** thresholds. Note that this Pipeline must be created before the alarm event happens, and that the name is case-sensitive.

Activity Monitor

The Activity Monitor configures how agent inactivity is reported. When contact is lost with an agent, an inactivity warning or error event is fired if the configured time in minutes has elapsed since last contact.

Inactivity Warning (Minutes): The number of minutes before a **warning** threshold alarm is activated. (default 5)

Inactivity Error (Minutes): The number of minutes before an **error** threshold alarm is activated. (default 15)

System Metric Tresholds

In addition to inactivity alarms, alarms can be set on all agents when certain metrics like CPU usage, number of clients, error rates, and more are reached. Each one has both a **warning** and an **error** level.

CPU Usage Warning (%): The **warning** level of an Agent's CPU usage. (default 70)

CPU Usage Error (%): The **error** level of an Agent's CPU usage. (default 90)

Memory Usage Warning (%): The **warning** level of an Agent's memory (RAM) usage. (default 70)

Memory Usage Error (%): The **error** level of an Agent's memory (RAM) usage. (default 90)

Errors Per Minute Warning: The **warning** level of an Agent's error rate (minute). The contents of the Agent's errors can be checked in the Agent's console. (default 2)

Errors Per Minute Error: The **error** level of an Agent's error rate (minute). The contents of the Agent's errors can be checked in the Agent's console. (default 5)

Errors Per Hour Warning: The **warning** level of an Agent's hourly error rate. The contents of the Agent's errors can be checked in the Agent's console. (default 20)

Errors Per Hour Error: The **error** level of an Agent's hourly error rate. The contents of the Agent's errors can be checked in the Agent's console. (default 60)

Connected Clients Warning: The number of clients connected to that Agent required to raise a **warning** alarm. (default 50)

Connected Clients Error: The number of clients connected to that Agent required to raise an **error** alarm. (default 100)

Queries Per Sec Warning: The number of SQL queries per second from that Agent required to raise a **warning** alarm. (default 5)

Queries Per Sec Error: The number of SQL queries per second from that Agent required to raise an **error** alarm. (default 10)

Query Duration Warning (MS): The average duration of a SQL query on an Agent (in milliseconds) to raise a **warning** alarm. (default 10000ms or 10 seconds)

Query Duration Error (MS): The average duration of a SQL query on an Agent (in milliseconds) to raise an **error** alarm. (default 30000ms or 30 seconds)

DB Connections Warning: The number of currently active database connections on an Agent to raise a **warning** alarm. (default 8)

DB Connections Error: The number of currently active database connections on an Agent to raise an **error** alarm. (default 12)

Active Queries Warning: The number of currently active SQL queries on an Agent to raise a **warning** alarm. (default 60)

Active Queries Error: The number of currently active SQL queries on an Agent to raise an **error** alarm. (default 100)

Next...

- EAM in the Designer
- Alarm Status
- Alarm Journal

EAM in the Designer

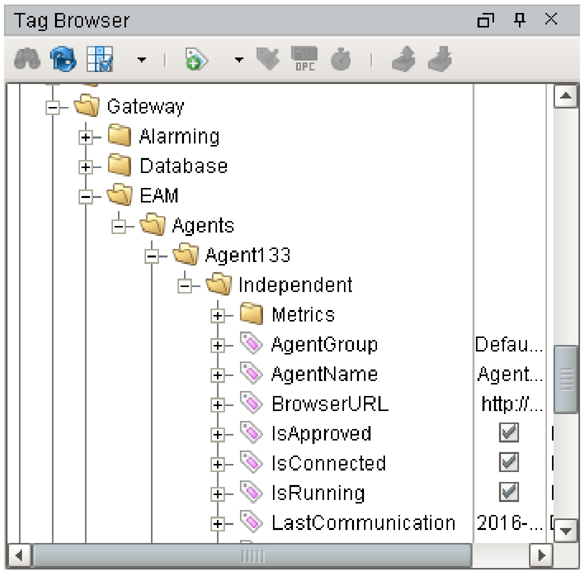
INDUCTIVE UNIVERSITY

EAM in the Designer

Watch the Video

Agent System Tags

System tags are created on the controller for each agent, allowing you to build EAM monitoring clients, set alarms on individual tags, and many other tag-related operations. To view EAM tags, launch a Designer on the controller Gateway. In the tag browser, navigate to System -> Gateway -> EAM -> Agents. A tag folder is created for each agent. As with other system tag values, all EAM system tag values are read-only.



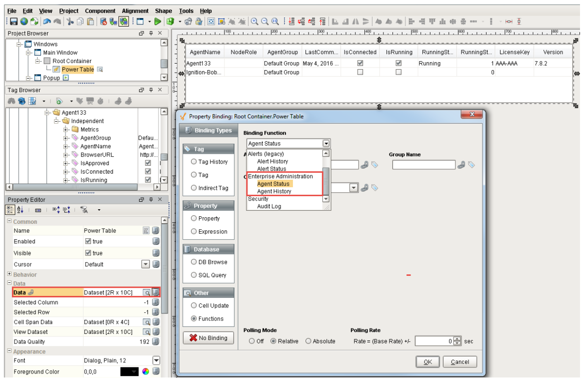
```
Gateway
  Alarming
  Database
  EAM
    Agents
      Agent133
        Independent
          Metrics
          AgentGroup           Defau...
          AgentName            Agent...
          BrowserURL           http://...
          IsApproved           ✓
          IsConnected          ✓
          IsRunning            ✓
          LastCommunication    2016-...
```

Property Binding Functions

The EAM adds its own binding functions to the standard property binding functions, allowing you to quickly add agent information to a component such as a table. The following binding functions are available:

Agent Status: reports some basic information about each agent, such as platform version, connection status, and the last time a message was received from the agent.

Agent History: reports agent events from the the **agent_events** table in the controller's configured database. Agent events include gateway task results, abnormal health statistics, and agent communication status events.



AgentName	NodeRole	AgentGroup	LastComm...	IsConnected	IsRunning	RunningSt...	RunningSt...	LicenseKey	Version
Agent133		Default Group	May 4, 2016 ...	✓	✓	Running	1	AAA-AAA	7.8.2
Ignition-Bob...		Default Group		☐	☐		0		

Name	Power Table
Enabled	✓ true
Visible	✓ true
Cursor	Default
Data	Dataset [2R x 10C]
Selected Column	-1
Selected Row	-1
Cell Span Data	Dataset [0R x 4C]
View Dataset	Dataset [2R x 10C]
Data Quality	192
Font	Dialog, Plain, 12
Foreground Color	0,0,0

Binding Function: Agent Status

Polling Mode: ○ Off ● Relative ○ Absolute

Rate = (Base Rate) +/- 0 sec

Scripting Functions

There are several scripting functions that are included in the system.eam package:

- **getGroups()**: returns a String list of all agent groups configured in the controller
- **queryAgentHistory()**: returns a Dataset of agent events from the controller's configured database
- **queryAgentStatus()**: returns a Dataset with some basic information about each agent, such as platform version, connection status, and the last time a message was received from the agent

More details and examples can be found in the system.eam section of the Appendix.

Next...

- Configuring Controllers and Agents
- Tags
- Binding Properties
- Scripting Functions

Configuring Controllers and Agents

Learn how to set up Controllers and Agents

Tips for Configuring Controllers and Agents

- If you have any existing installations of Ignition running, it's a good idea to perform a backup before you begin this process.
- If you have 2 or more Gateway Webpages open in your browser, you may want to have the Controller and Agents on separate monitors so you don't get confused which one you are working on.
- It's always a good idea to verify that all your modules are loaded correctly. Go to Configure > Modules to check which modules you have installed.

In this section ...

Creating a Controller

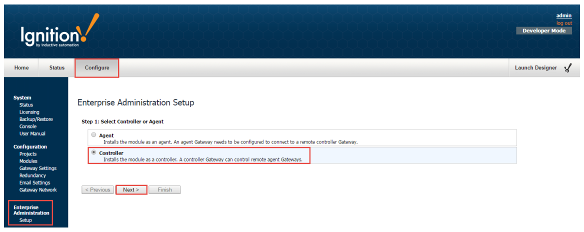
Controller Setup

To set up the EAM, you must first designate one machine as the controller. Keep track of the IP address of this machine, as you must enter this address on each agent during agent installation.

Install the EAM module in the Gateway if it has not already been installed. After module installation, you must designate a database connection that will be used by the controller to store agent event history. Navigate to Configure -> Databases -> Connections, and set up a connection to the database that you intend to use. After the database connection has been set up, navigate to Configure -> Enterprise Administration -> Setup. You will now enter the EAM Setup Wizard.

Step 1: Select Controller or Agent

Since you are installing the controller, select “Controller” on this screen. Click the Next button.



Enterprise Administration Setup

Step 1: Select Controller or Agent

Agent
Installs the module as an agent. An agent Gateway needs to be configured to connect to a remote controller Gateway.

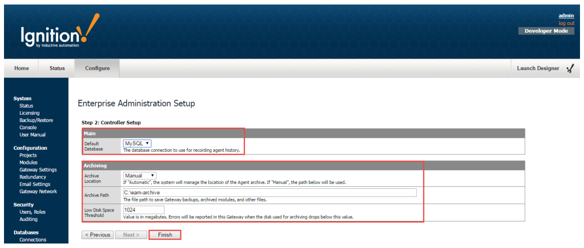
[✓ Controller]
Installs the module as a controller. A controller Gateway can control remote agent Gateways.

Step 2: Controller Setup

Default Database- for the default database, you must select the database connection that you created before entering the EAM Setup Wizard. If you forgot to create the database connection, you can do so now and return to the EAM Setup Wizard later.

Archive Path- this is a file path on the Gateway’s local machine where all archived files will be stored. You can use a locally mounted network share for archiving. Use a value such as “E:\eam-archive”. It is strongly recommended that you put a system into place to externally monitor the free size of the disk that contains the archive. Gateway backups can potentially be many megabytes in size, and can eat up large volumes of space when using many agents.

After you have determined your settings, click the Finish button to save the controller settings and exit the wizard. This machine is now ready to accept agents.



Enterprise Administration Setup

Step 2: Controller Setup

Main	
Default Database	MySQL The database connection to use for recording agent history.

Archiving	
Archive Location	Manual If "Automatic", the system will manage the location of the Agent archive. If "Manual", the path below will be used.
Archive Path	C:\eam-archive The file path to save Gateway backups, archived modules, and other files.
Low Disk Space Threshold	1024 Value is in megabytes. Errors will be reported in this Gateway when the disk used for archiving drops below this value.

Next...

- Adding an Agent
- Agent Event Threshold Settings

INDUCTIVE UNIVERSITY

Creating a Controller

Watch the Video

Adding an Agent

Agent Setup

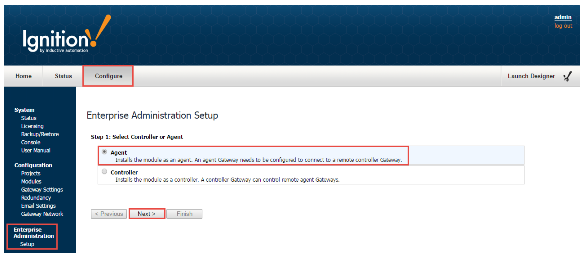
To communicate with the controller, agents send data across a Gateway Network connection. If your Gateway Network setup requires that connections be manually approved in advance on the controller Gateway, you will need to set up the Gateway Network connection before starting the EAM Setup Wizard. You will be able to select the existing Gateway Network connection within the wizard. Navigate to Configure -> Enterprise Administration -> Setup to begin the EAM Setup Wizard.

Adding an Agent

Watch the Video

Step 1: Select Controller or Agent

Since you are installing an agent, select “Agent” on this screen. Click the Next button.



Enterprise Administration Setup

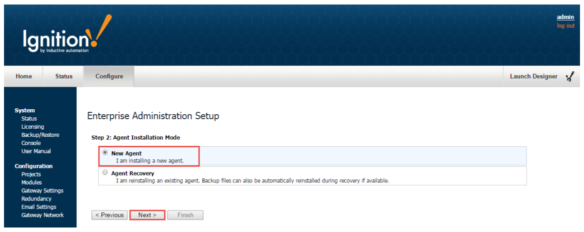
Step 1: Select Controller or Agent

[✓ Agent]
Installs the module as an agent. An agent Gateway needs to be configured to connect to a remote controller Gateway.

Controller
Installs the module as a controller. A controller Gateway can control remote agent Gateways.

Step 2: Agent Installation Mode

Select New Agent or Agent Recovery. Select “New Agent” if you are setting up a machine that is unknown to the controller. Select “Agent Recovery” if you are reinstalling an existing agent (possibly on a different machine), and the machine has been known to the controller in the past. This option also gives an opportunity to perform a system recovery (restore Gateway backup, reinstall missing modules and license) if possible. After you have made your selection, click the Next button.



Enterprise Administration Setup

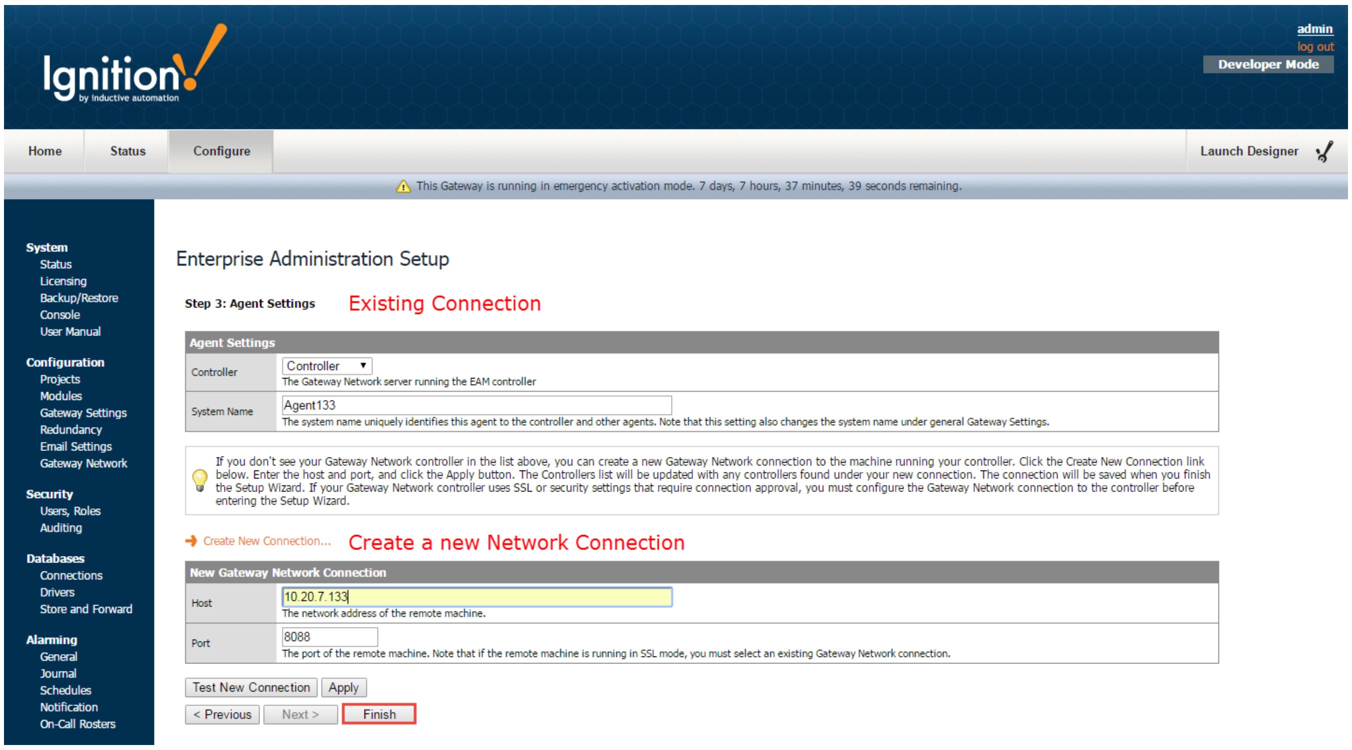
Step 2: Agent Installation Mode

[✓ New Agent]
I am installing a new agent.

Agent Recovery
I am reinstalling an existing agent. Backup files can also be automatically reinstalled during recovery if available.

Step 3: Gateway Network Setup

As mentioned previously, a Gateway Network connection is required to communicate with a controller. On this page, you can either enter the settings to create a new Gateway Network connection, or select an existing connection from the dropdown. Use the Test Connection button to test the settings you have selected. Note that you cannot create a new SSL connection on this page. When using SSL, you must create the Gateway Network connection before entering the EAM Setup Wizard.



This Gateway is running in emergency activation mode. 7 days, 7 hours, 37 minutes, 39 seconds remaining.

Enterprise Administration Setup

Step 3: Agent Settings — Existing Connection

Agent Settings	
Controller	Controller ▾ The Gateway Network server running the EAM controller
System Name	Agent133 The system name uniquely identifies this agent to the controller and other agents. Note that this setting also changes the system name under general Gateway Settings.

If you don't see your Gateway Network controller in the list above, you can create a new Gateway Network connection to the machine running your controller. Click the Create New Connection link below. Enter the host and port, and click the Apply button. The Controllers list will be updated with any controllers found under your new connection. The connection will be saved when you finish the Setup Wizard. If your Gateway Network controller uses SSL or security settings that require connection approval, you must configure the Gateway Network connection to the controller before entering the Setup Wizard.

Create New Connection... — Create a new Network Connection

New Gateway Network Connection	
Host	10.20.7.133 The network address of the remote machine.
Port	8088 The port of the remote machine. Note that if the remote machine is running in SSL mode, you must select an existing Gateway Network connection.

Step 4: Agent Settings

Gateway Network Server- select your controller machine from this dropdown list.

System Name- when running in New Agent mode, you have the opportunity here to set the Gateway System Name. This name is used to reference the agent on the controller Gateway pages. Note that changing the name here will also change the system name under general Gateway Settings. When running in Agent Recovery mode, you must choose the Gateway System Name from the System Name dropdown. This dropdown holds all the agents for which the controller has lost contact. If you are running an Agent Recovery, then the agent you are trying to recover should be on this list.

Perform System Recovery- this checkbox is only shown when running in Agent Recovery mode. After you click the Finish button, the Gateway will attempt to download the most recent gateway backup, previously installed modules, and the license file.

After you have made your selections, click the Finish button.

Agent Post Setup

The EAM module will now attempt to register the local Gateway as an EAM agent on the controller, and download files if performing a system recovery.

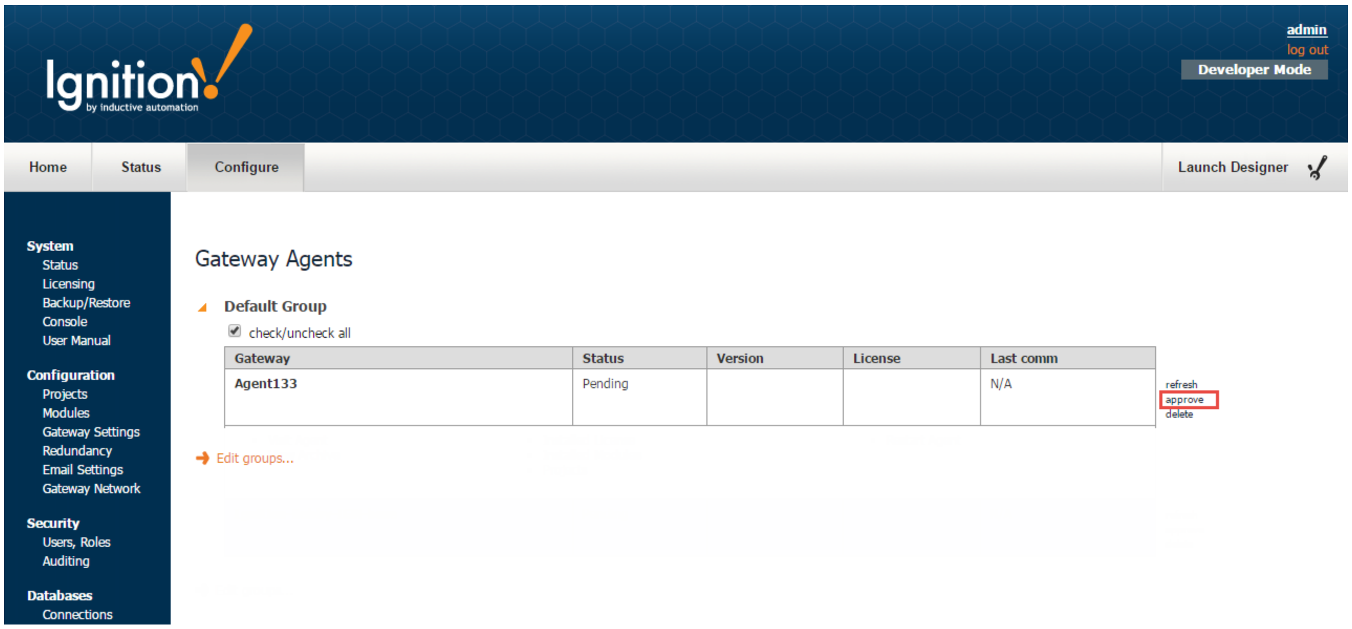
If you are running in New Agent mode, you will see the agent status check on the screen. When the agent status check is complete, you will need to navigate to the controller Gateway and approve the agent on the Gateway Agents page. Click the Retry button to recheck agent status on the controller if the first status check failed.

If you are running in Agent Recovery mode, and you have selected the System Recovery option, you will see the download status of the license file, installed modules, and the most recently collected Gateway backup file. Depending on the contents of your archive, some files may not be available. Click the Apply Files button to apply downloaded files. Note that this operation will require a Gateway restart if a Gateway backup file needs to be applied. Click the Ignore Files button to close this page and not apply downloaded files. Note that the agent settings have already been saved, so it is safe to not apply downloaded files if you choose. Click the Retry button to attempt to download the recovery files again.

[IMPORTANT] After you have set up your agents, it is very strongly recommended that you set up a regularly scheduled Collect Backup task for your agents. System recovery thru the EAM Setup Wizard will not work if there are no archived backup files!

Agent Approval

When an unknown agent first contacts the controller, it is saved in pending status. No interaction with the controller is allowed until you approve the agent. Agent version and license information fields will remain empty until the agent is approved. To approve the agent, click on the “approve” link on the right side. After a few moments, the agent’s status should change to “Connected”. To remove an agent, click the “delete” link on the right side. If the agent machine is still active, the agent will reappear on the list in pending status, but will no longer be able to interact with the controller. To remove the agent permanently, you must navigate to the agent Gateway and uninstall the agent via the Agent Settings page.



Gateway Agents

Default Group

check/uncheck all

Gateway	Status	Version	License	Last comm
Agent133	Pending			N/A

Agent Groups

Agents can be grouped into agent groups that you create. This allows you to organize agent lists by location or agent function. Agent groups can also be selected in gateway tasks. For example, you can create an agent group for a location that collects backups every night at 3AM. Any new agent that is added to the group will automatically be included when the gateway tasks executes. To create an agent group, click the "Edit Groups" link at the bottom of the page. You will now be on the Groups page, where you can add groups, edit a group name, and delete a group. If you delete a group that contains agents, the agent will be automatically reassigned to the default group.

To assign an agent to a group, check the checkbox on the left side next to the agent name. Then locate the agent groups dropdown on the right side next to the "Move selected Agents to" link. Select the new group from the dropdown. Then click the "Move selected Agents to" link to move the agents.

Agent Actions

Each agent row is expandable with a ▶ arrow link on the left side. When the row is expanded, you will see some actions that can be taken on the agent.

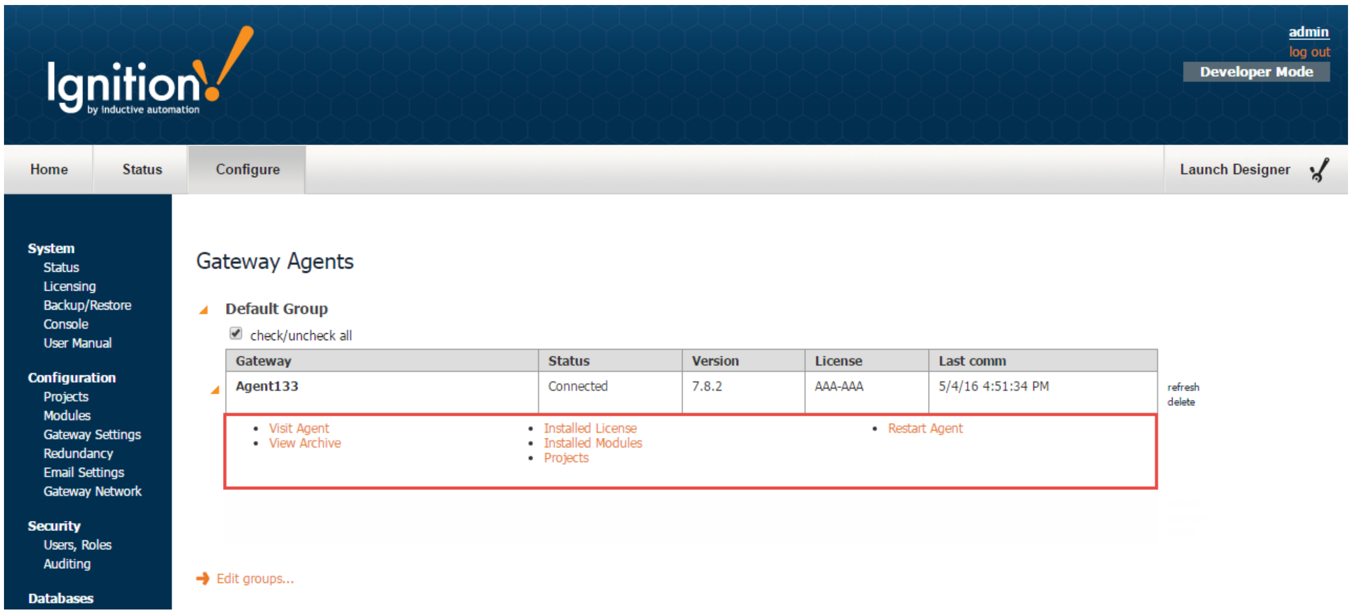
Restart: creates a gateway task that immediately restarts the agent Gateway. After clicking the link, you will be asked to confirm that you really want to restart the agent.

License Detail: opens a page which compares the agent's current license to the license key on file at Inductive Automation. This page is handy when you have purchased new software for a license, and you need to verify that your agent's license key is up to date. You can update the license for this agent by clicking the "Update Agent License" link at the bottom of the screen. This will schedule a gateway task to immediately update the license on the agent.

View Archive: opens a page that allows you to view all gateway backups and modules that have been archived for this agent. You can schedule a gateway task to restore a specified gateway backup from this page by clicking the "Restore Backup" link on the right side.

Installed Modules: opens a page that displays all modules that are currently installed in the agent Gateway. The module version and the current state of each module is also displayed

Projects Detail: opens a page that displays all projects that currently exist on the agent. The Project Source field is populated by the controller name if the project was deployed to the agent by the controller via a gateway task.



Gateway Agents

Default Group

check/uncheck all

Gateway	Status	Version	License	Last comm
Agent133	Connected	7.8.2	AAA-AAA	5/4/16 4:51:34 PM

refresh
delete

- Visit Agent
- View Archive
- Installed License
- Installed Modules
- Projects
- Restart Agent

Edit groups...

Next...

- Agent Recovery
- License Management
- Agent Task Scheduling

Agent Recovery

Once the Controller is up and running, you can use it to perform recoveries on Agents. For example, if one of your computers crashed that is running Ignition, you can do a fresh install from a previous backup collection that was performed on your Agent. When the recovery is complete, the Agent will immediately pick up where it left off like nothing ever happened.

Backup Required to Perform Agent Recovery

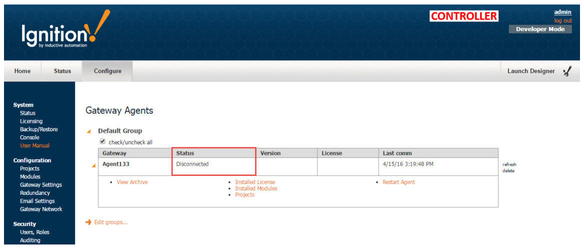
Agent recovery requires that a backup collection was performed, otherwise, you will not be able to use this procedure to recover your Agent's instance of Ignition.

INDUCTIVE UNIVERSITY

Agent Recovery

Watch the Video

In this example, let's assume that the Agent computer crashed. To check the status, go to your **Controller,** navigate to **Enterprise Administration > Agent Management** and check the status of the Gateway Agent. You can see that Agent133 is disconnected.



Gateway Agents

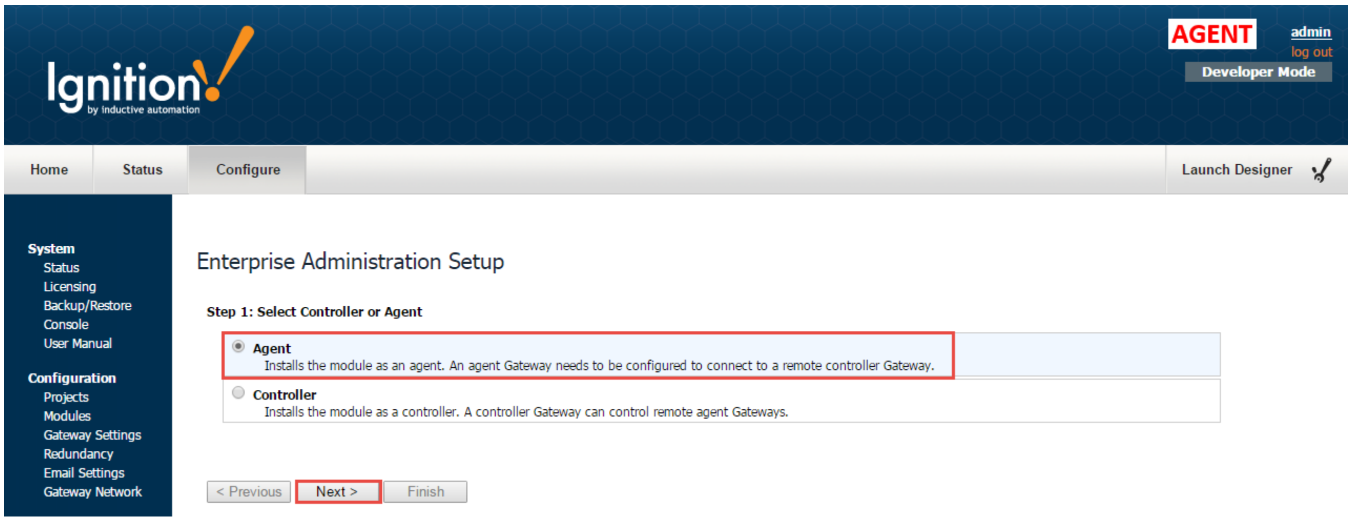
Default Group

check/uncheck all

Gateway	Status	Version	License	Last comm
Agent133	Disconnected			4/15/16 3:19:48 PM

First: on your **Agent** computer, install Ignition and make sure the EAM Module is also installed (it is not added by default if you use the 'Typical' selection during installation).

On your **Agent**, go to the **Configure** tab and select the **Enterprise Administration Setup**. The EAM wizard will walk you through each step of the Agent recovery process.
Select **Agent** to configure. Press **Next**.



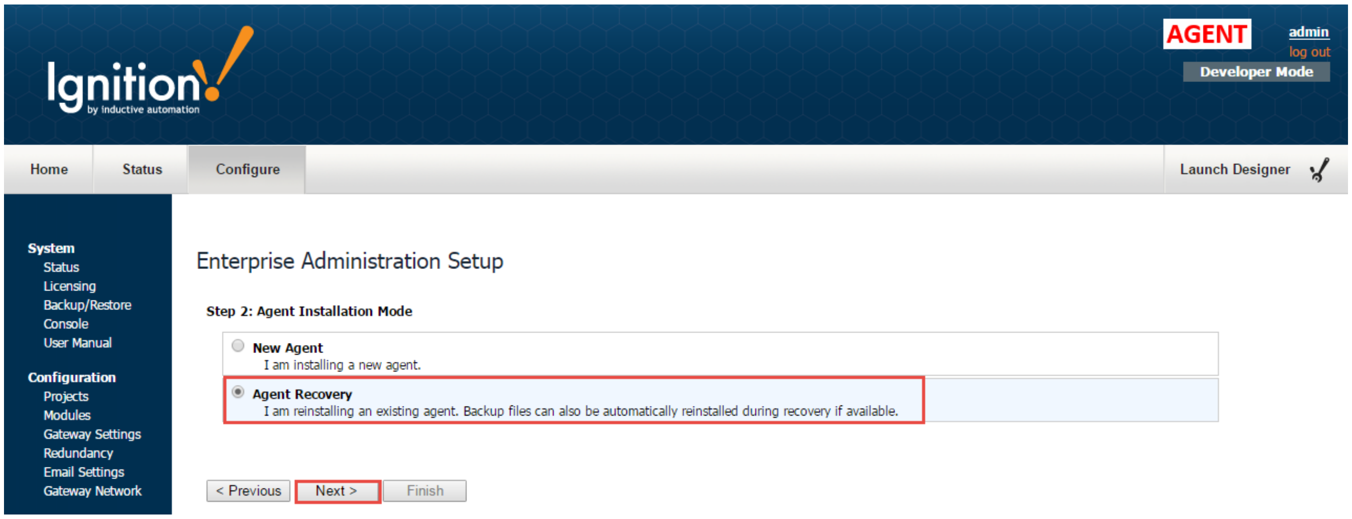
Enterprise Administration Setup

Step 1: Select Controller or Agent

[✓ Agent] Installs the module as an agent. An agent Gateway needs to be configured to connect to a remote controller Gateway.

Controller Installs the module as a controller. A controller Gateway can control remote agent Gateways.

In order to continue with the Agent Recovery procedure, you must have a backup collection of the Agent. When you initially setup your Agent on the Controller, you created an Archive Path where the backup files are stored. EAM will automatically know where to retrieve the backup files.
Click on **Agent Recovery** and press **Next**.



Enterprise Administration Setup

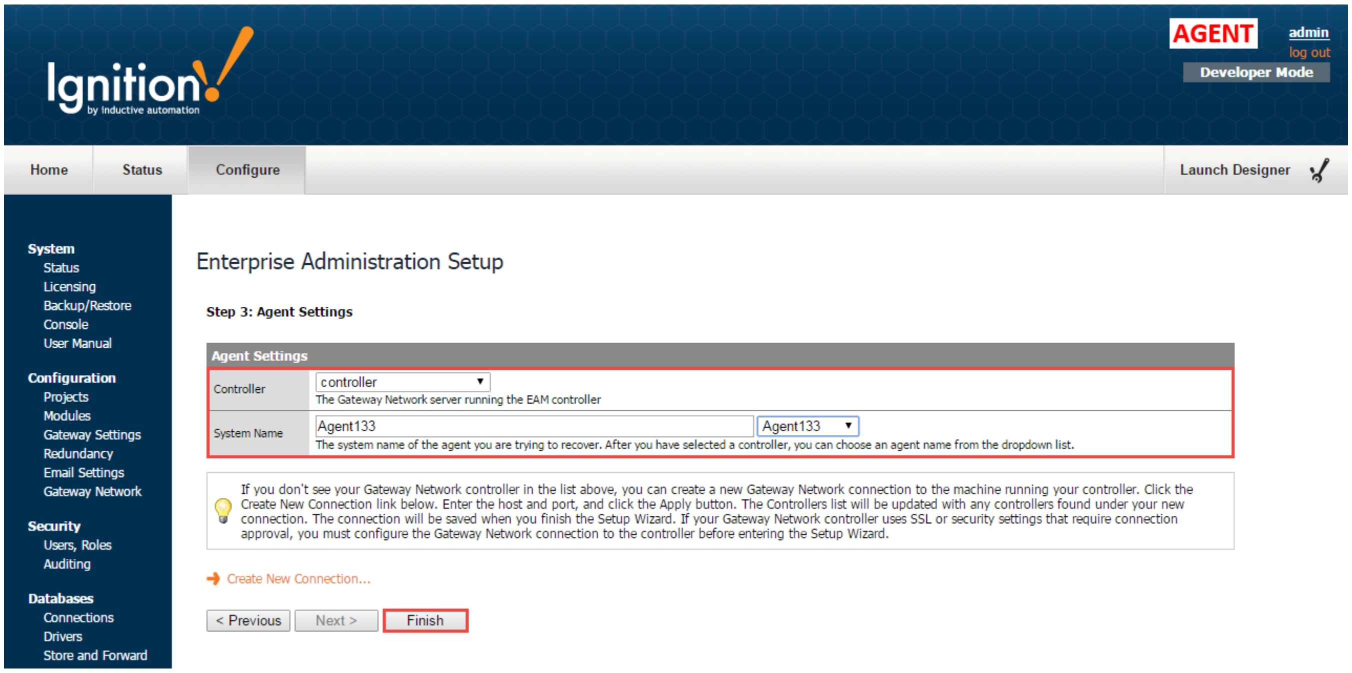
Step 2: Agent Installation Mode

New Agent
I am installing a new agent.

[✓ Agent Recovery]
I am reinstalling an existing agent. Backup files can also be automatically reinstalled during recovery if available.

< Previous | Next > | Finish

Since Agent133 existed previously, the EAM remembered the Agent's prior settings. Choose from the list of names in the dropdown lists for the Controller and Agent.
In this example, select your controller from the dropdown list, and select this Agent from the System Name dropdown list (or you can type it in). Press **Finish**.



Enterprise Administration Setup

Step 3: Agent Settings

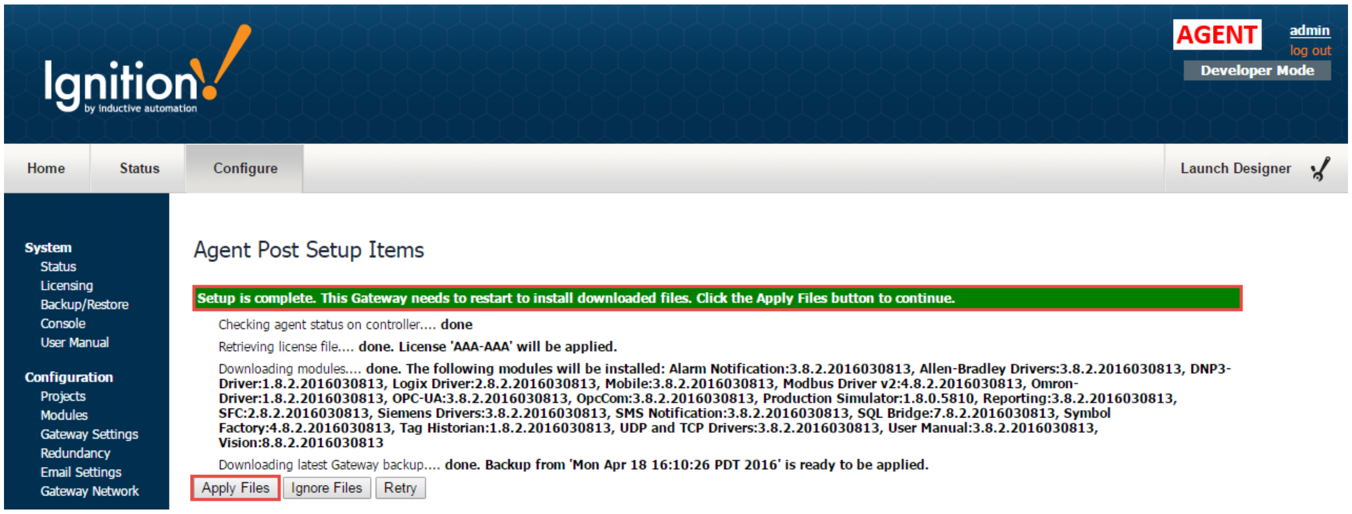
Agent Settings	
Controller	controller The Gateway Network server running the EAM controller
System Name	Agent133 / Agent133 The system name of the agent you are trying to recover. After you have selected a controller, you can choose an agent name from the dropdown list.

If you don't see your Gateway Network controller in the list above, you can create a new Gateway Network connection to the machine running your controller. Click the Create New Connection link below. Enter the host and port, and click the Apply button. The Controllers list will be updated with any controllers found under your new connection. The connection will be saved when you finish the Setup Wizard. If your Gateway Network controller uses SSL or security settings that require connection approval, you must configure the Gateway Network connection to the controller before entering the Setup Wizard.

Create New Connection...

< Previous | Next > | Finish

The Agent will go through a series of steps: check the Agent status on the Controller, retrieve the Ignition license file, and download Ignition modules and the latest Gateway backup to the Agent.
When you get a message that this step is complete, press **Apply Files**.



Agent Post Setup Items

Setup is complete. This Gateway needs to restart to install downloaded files. Click the Apply Files button to continue.

Checking agent status on controller.... **done**

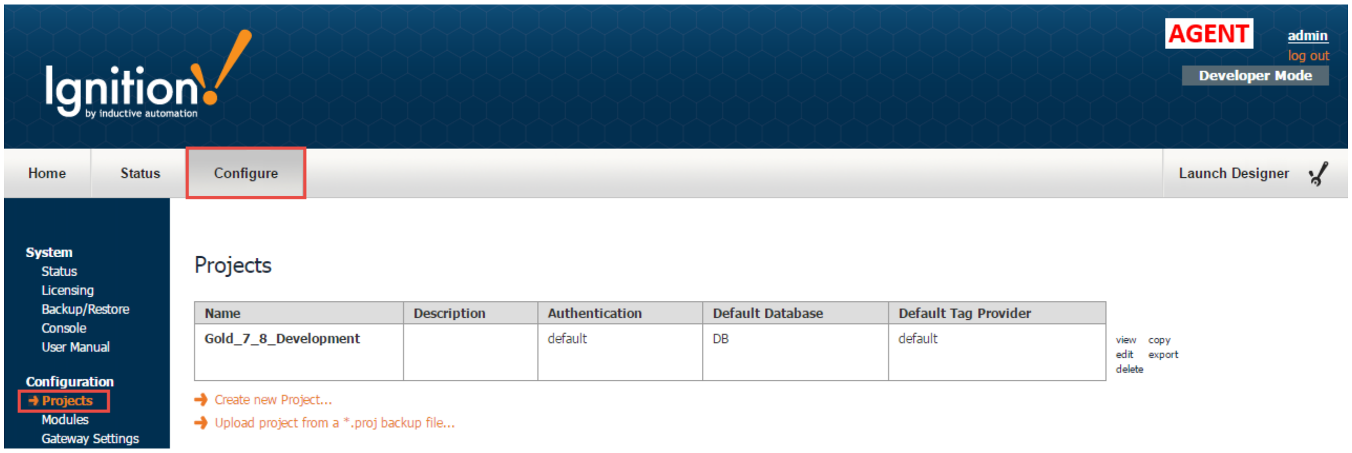
Retrieving license file.... **done. License 'AAA-AAA' will be applied.**

Downloading modules.... **done. The following modules will be installed: Alarm Notification:3.8.2.2016030813, Allen-Bradley Drivers:3.8.2.2016030813, DNP3-Driver:1.8.2.2016030813, Logix Driver:2.8.2.2016030813, Mobile:3.8.2.2016030813, Modbus Driver v2:4.8.2.2016030813, Omron-Driver:1.8.2.2016030813, OPC-UA:3.8.2.2016030813, OpcCom:3.8.2.2016030813, Production Simulator:1.8.0.5810, Reporting:3.8.2.2016030813, SFC:2.8.2.2016030813, Siemens Drivers:3.8.2.2016030813, SMS Notification:3.8.2.2016030813, SQL Bridge:7.8.2.2016030813, Symbol Factory:4.8.2.2016030813, Tag Historian:1.8.2.2016030813, UDP and TCP Drivers:3.8.2.2016030813, User Manual:3.8.2.2016030813, Vision:8.8.2.2016030813**

Downloading latest Gateway backup.... **done. Backup from 'Mon Apr 18 16:10:26 PDT 2016' is ready to be applied.**

Once you press the Apply Files button, you will see two consecutive progress bars; one for stopping the Gateway and one for starting. It will take about a minute to bring the Gateway up. The screen will refresh once the Gateway is up and running.
Click on the **Configure** tab. You will be prompted to log into the Designer.

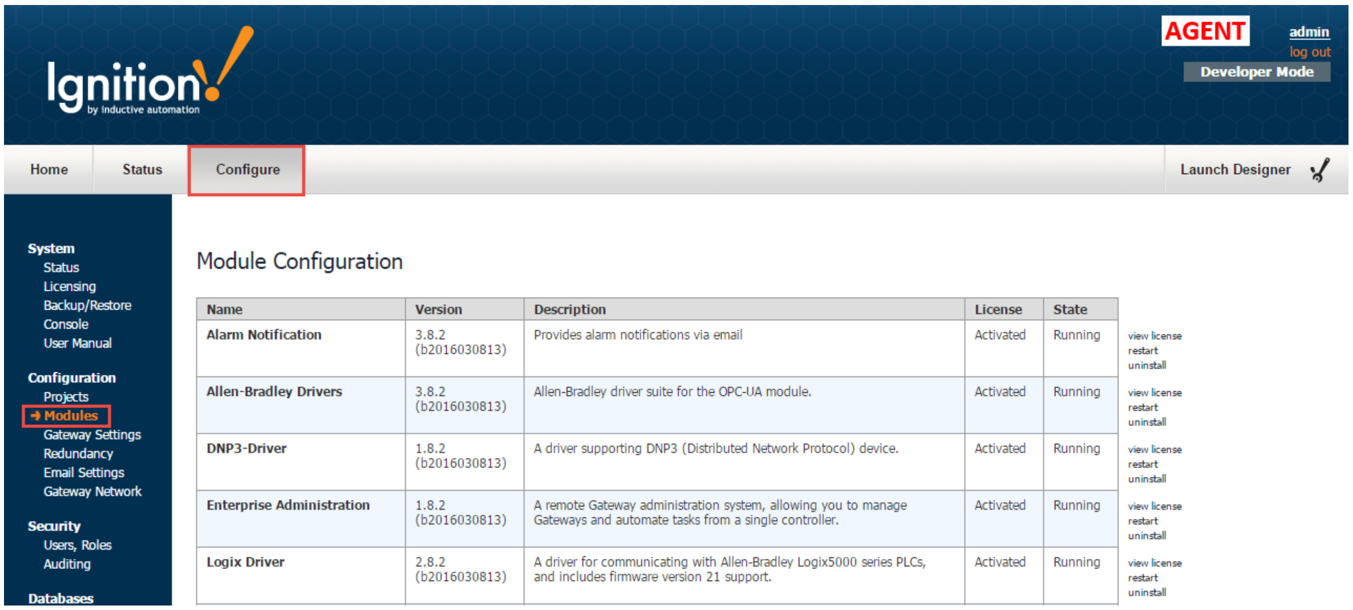
After logging in, you will see the System Status window. Click on **Configure > Projects** and you will notice that all your projects are restored.



Projects

Name	Description	Authentication	Default Database	Default Tag Provider
Gold_7_8_Development		default	DB	default

It's also a good idea to verify that all your modules are activated by going to the **Configure** tab and selecting **Modules**.



Module Configuration

Name	Version	Description	License	State	
Alarm Notification	3.8.2 (b2016030813)	Provides alarm notifications via email	Activated	Running	view license restart uninstall
Allen-Bradley Drivers	3.8.2 (b2016030813)	Allen-Bradley driver suite for the OPC-UA module.	Activated	Running	view license restart uninstall
DNP3-Driver	1.8.2 (b2016030813)	A driver supporting DNP3 (Distributed Network Protocol) device.	Activated	Running	view license restart uninstall
Enterprise Administration	1.8.2 (b2016030813)	A remote Gateway administration system, allowing you to manage Gateways and automate tasks from a single controller.	Activated	Running	view license restart uninstall
Logix Driver	2.8.2 (b2016030813)	A driver for communicating with Allen-Bradley Logix5000 series PLCs, and includes firmware version 21 support.	Activated	Running	view license restart uninstall

Next...

- License Management

License Management

License Management

The License Management page acts as a central repository for managing Ignition licenses. You can preload a list of licenses, view details of a license as it appears on file at Inductive Automation , assign licenses to individual agents, and bulk assign free licenses in the Activate License gateway task. To visit the License Management page, navigate to Configure -> Enterprise Administration -> License Management.

License Management

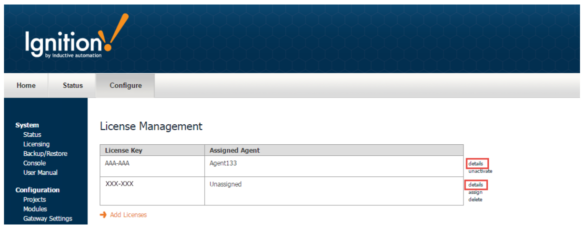
Watch the Video

Agent License Reporting

Whenever an approved agent contacts the controller, it sends its license information. The License Management page is automatically updated whenever new or updated license information is received from an agent. This means that even if you install a license directly in an agent Gateway, the license installation will still be known to the controller.

License Details

You can click on the “details” link on the right side of each license to view modules and versions currently assigned to the license. Note that the information displayed on the License Detail page reflects the license as it appears on file at Inductive Automation. This functionality allows you to confirm exactly which modules and versions are currently assigned to a license.



License Management

License Key	Assigned Agent	
AAA-AAA	Agent133	details unactivate
XXX-XXX	Unassigned	details assign delete

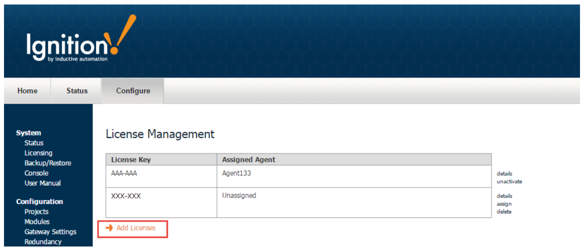
Add Licenses

Activate and unactivate a license

The controller acts as a license proxy on behalf of agents. This means that only the controller requires Internet access to the Inductive Automation licensing servers. Whenever you use the License Management page or a gateway task to assign, update or unactivate a license, the controller will request system information from the agent and forward the information to the licensing servers. If performing an activation, the activation data is sent back to the agent after the data is received from the licensing servers.

Preloading Licenses

You can preload a list of license keys in the controller. After the license list is loaded, you can individually assign licenses to agents, or assign all free licenses to agents in one step via the Activate License task. To preload a license list, navigate to the License Management page and click the “Add Licenses” link.



License Management

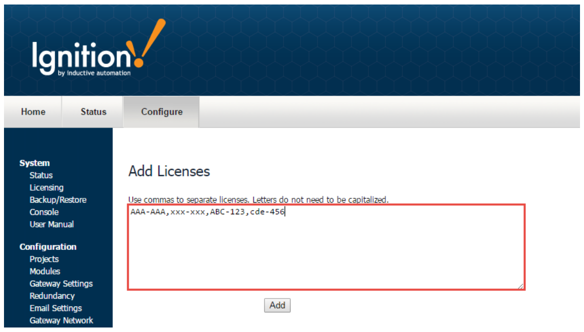
License Key	Assigned Agent	
AAA-AAA	Agent133	details unactivate
XXX-XXX	Unassigned	details assign delete

Add Licenses

The license list must be formatted like so:

```
AAA-AAA, xxx-xxx, ABC-123, cde-456
```

Click the Add button to save preloaded licenses.



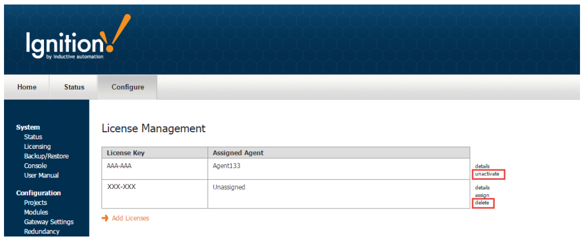
Add Licenses

Use commas to separate licenses. Letters do not need to be capitalized.

```
AAA-AAA,xxx-xxx,ABC-123,cde-456
```

Deleting Licenses

You can remove unused licenses from the License Management page, but only if the license is not currently assigned to any agent. If you want to delete an assigned license, click the “unactivate” link on the right side of the license. After the Assigned Agent field changed to “Unassigned,” you can delete the license.



License Management

License Key	Assigned Agent	
AAA-AAA	Agent133	details unactivate
XXX-XXX	Unassigned	details assign delete

Next...

- Agent Task Scheduling

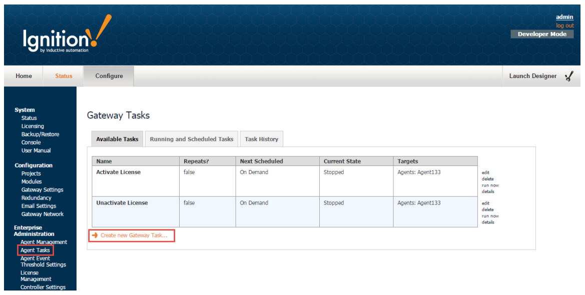
Agent Task Scheduling

Agent Task Scheduling

Watch the Video

Agent Gateway Tasks

Agent gateway tasks are enormously useful for applying an action to many agent gateways at once. To create a new task, or to view running tasks, navigate to Config -> Enterprise Administration -> Agent Tasks. The Gateway Tasks page will display scheduled tasks, currently executing tasks, and unscheduled tasks that can be run on demand. To create a new task, Click the “Create new Gateway Task” link. You will enter the Gateway Task wizard.

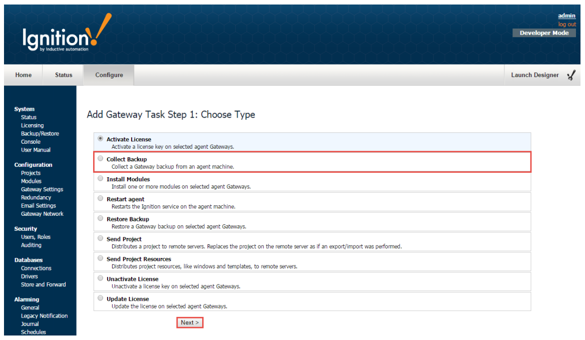


Gateway Tasks

Name	Repeats?	Next Scheduled	Current State	Targets	
Activate License	false	On Demand	Stopped	Agents: Agent133	edit delete run now details
Unactivate License	false	On Demand	Stopped	Agents: Agent133	edit delete run now details

Step 1: Choose Type

You must select the type of task that you wish to execute. After you have selected a type of task, click the Next button.



Add Gateway Task Step 1: Choose Type

- ● **Activate License** — Activate a license key on selected agent Gateways.
- ○ **Collect Backup** — Collect a Gateway backup from an agent machine.
- ○ **Install Modules** — Install one or more modules on selected agent Gateways.
- ○ **Restart agent** — Restarts the Ignition service on the agent machine.
- ○ **Restore Backup** — Restore a Gateway backup on selected agent Gateways.
- ○ **Send Project** — Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- ○ **Send Project Resources** — Distributes project resources, like windows and templates, to remote servers.
- ○ **Unactivate License** — Unactivate a license key on selected agent Gateways.
- ○ **Update License** — Update the license on selected agent Gateways.

Step 2: Task Scheduling

Set the task name to a descriptive name of your choice. The following scheduling options are available:

Execute immediately: the task will begin execution as soon as you complete the wizard.

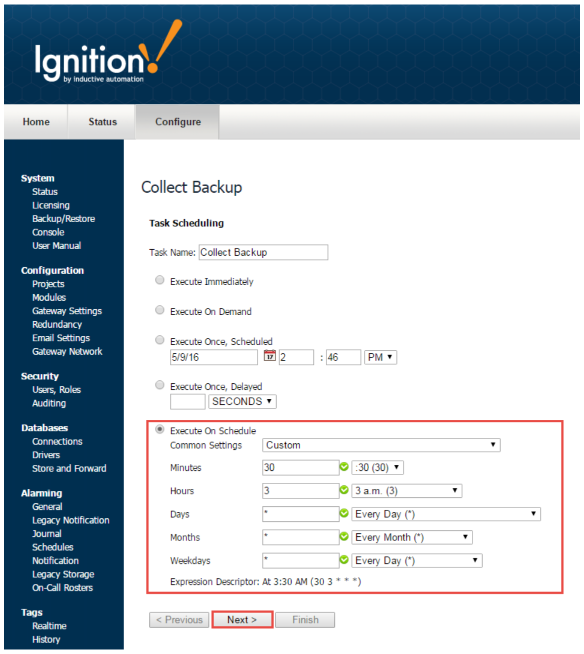
Execute on demand: the task will not be scheduled, but is available for immediate execution at any time after you complete the wizard.

Execute once, scheduled: the task will run once at the date and time that you specify.

Execute once, delayed: the task will run once after the specified number of seconds, minutes, hours, or days.

Execute on schedule: this type of schedule uses UNIX cron type scheduling, where a pattern of 5 numbers determines the recurring schedule. For example, “30 3 * * *” means run a task every day at exactly 3:30 AM. Use the dropdowns next to the pattern textboxes to build out the pattern for the schedule.

After you have determined your task schedule, click the Next button.



Collect Backup

Task Scheduling

Task Name: Collect Backup

( ) Execute Immediately

( ) Execute On Demand

( ) Execute Once, Scheduled
5/9/16 2 : 46 PM

( ) Execute Once, Delayed
SECONDS

(●) Execute On Schedule

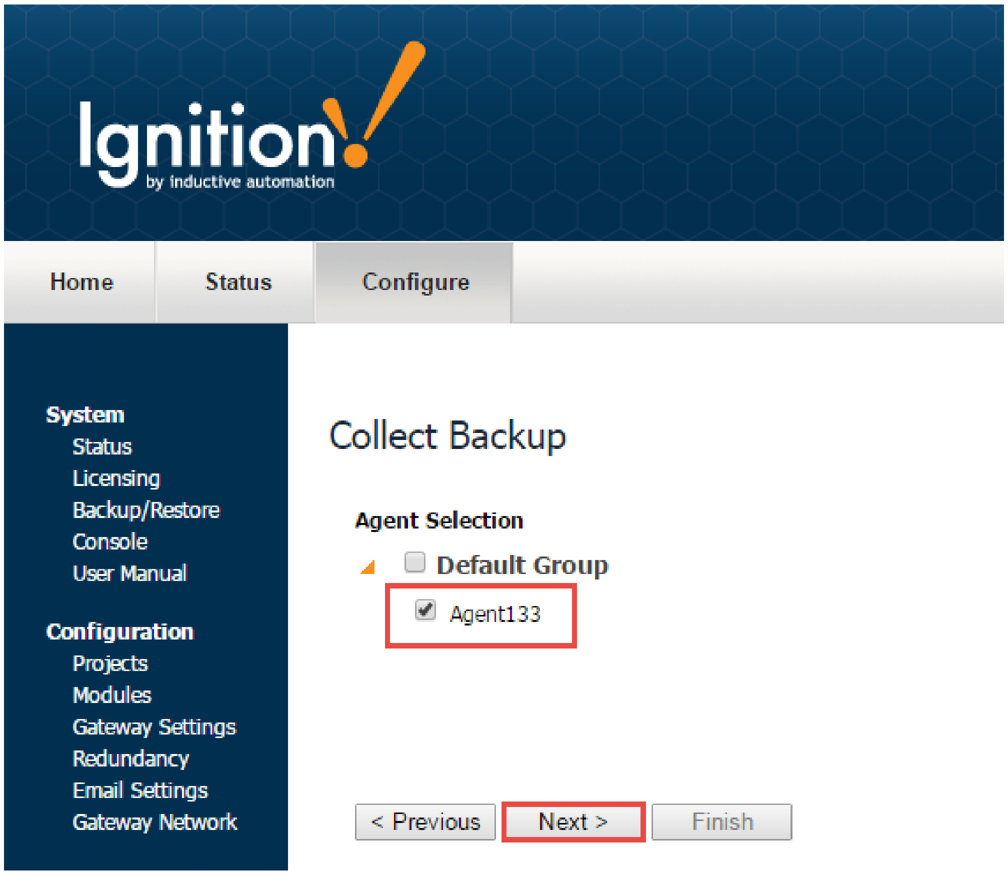
Common Settings	Custom	
Minutes	30	:30 (30)
Hours	3	3 a.m. (3)
Days	*	Every Day (*)
Months	*	Every Month (*)
Weekdays	*	Every Day (*)

Expression Descriptor: At 3:30 AM (30 3 * * *)

Step 3: Agent Selection

In this step, you will select the agent groups or individual agents that will be included when the task executes. You can select agents that are not currently connected, but you cannot select pending agents.

Depending on the type of task, you may need to complete another page to add some information required for that task. If not, you can click the Finish button to schedule the task.



Collect Backup

Agent Selection

☐ Default Group

☑ Agent133

Gateway Task Results

Gateway task execution is reported in the agent_events table in the controller's configured database. The individual result of the task execution for each agent is report as a separate entry. For example, the following query will return the results of all task executions:

```
select * from agent_events where event_category = 'task' order by event_time desc;
```

A task that completed normally for an agent will contain the text NORMAL in the event_level field. If an error occurred for an agent, the event_level field is set to ERROR, and the error message is recorded in the message field.

Running a task outside its schedule

You can run any task outside the task's normal schedule by clicking the "run now" link on the right side of the task. Keep in mind:

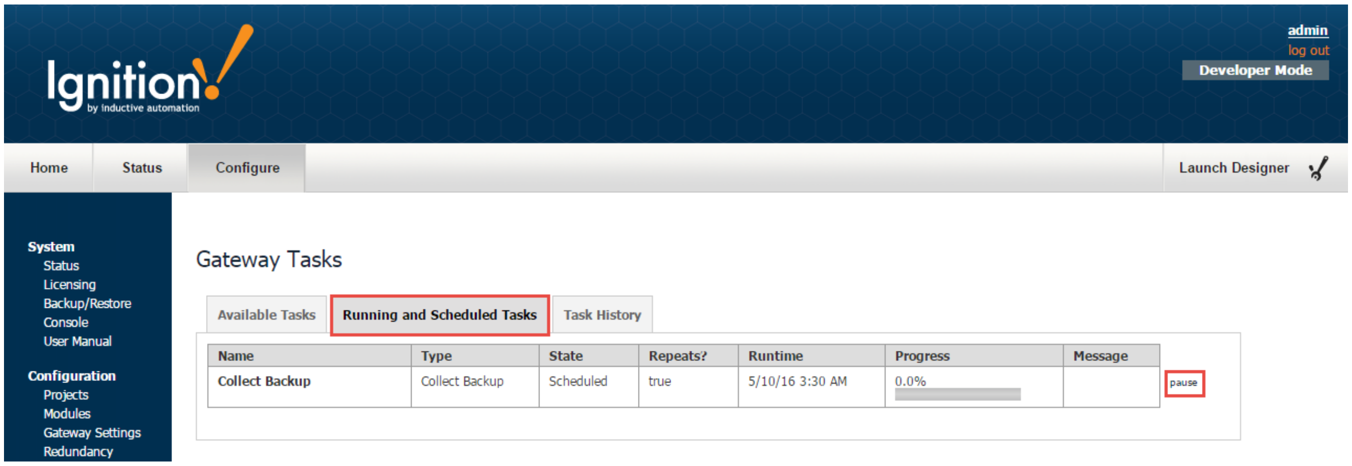
- Tasks that have been scheduled to run immediately are already running, and the "run now" link will not be visible.
- A task that has been scheduled to execute only once at a scheduled time or after a delay will be rescheduled to run immediately. The task will not run again after it has been rescheduled.
- A scheduled task will continue to run at its normally scheduled time, even if you run the task now.

Pausing and Cancelling Tasks (7.8.1)

Pause and Resume

After most tasks are created, they are put onto the task schedule. The exception to this is on demand tasks, which are not put on the schedule until you click the "run now" link. After a task has been added to the schedule, you can pause and resume the task. The state of the task (scheduled, running, waiting) and the scheduling configuration affect how the task is paused and resumed.

Task configuration	Pause action	Resume Action
Task is currently running or waiting	System will attempt to cancel outstanding calls to remote servers.	System will run task again against any servers that were outstanding when task was paused.
Task is scheduled once at a specific date and time	The task will be temporarily taken off the schedule.	If the specified time has not yet arrived, the task will be put back onto the schedule at the original time. If the specified time has already passed, the task will execute immediately upon resume.
Task is scheduled once after a specified delay	The task will be temporarily taken off the schedule.	The task will be put back onto the schedule at the original calculated time. For example, if the delay is set to 2 hours, and the task is paused for 1 hour, the task will fire in 1 hour after resume. If the delay period has already passed, it will fire immediately after resume.
Task is on a recurring schedule	All future iterations of the task will be temporarily taken off the schedule.	The task will **not** fire if task iterations have been missed when the task was paused. The task will return to firing on its normal schedule.



Name	Type	State	Repeats?	Runtime	Progress	Message
Collect Backup	Collect Backup	Scheduled	true	5/10/16 3:30 AM	0.0%	

Cancel

Scheduled one-shot tasks can be cancelled at any time before the task starts. Scheduled one-shot tasks include tasks that are scheduled to fire once after a delay or fire at a specific time. Running tasks can also be cancelled, but there are no guarantees about how much of the task is processed at the time of cancel. Keep in mind that cancelling a task will cause also delete the task at the same time, and it cannot be retrieved after cancel.

Next...

- Agent Task - Licensing

Agent Task - Licensing

One of the many tasks that Enterprise Administration Module (EAM) allows you to perform is license key (CD Key) activation and unactivation. This section demonstrates how to Activate, Unactivate and Update licenses from the Controller.

INDUCTIVE UNIVERSITY

Agent Task - Licensing

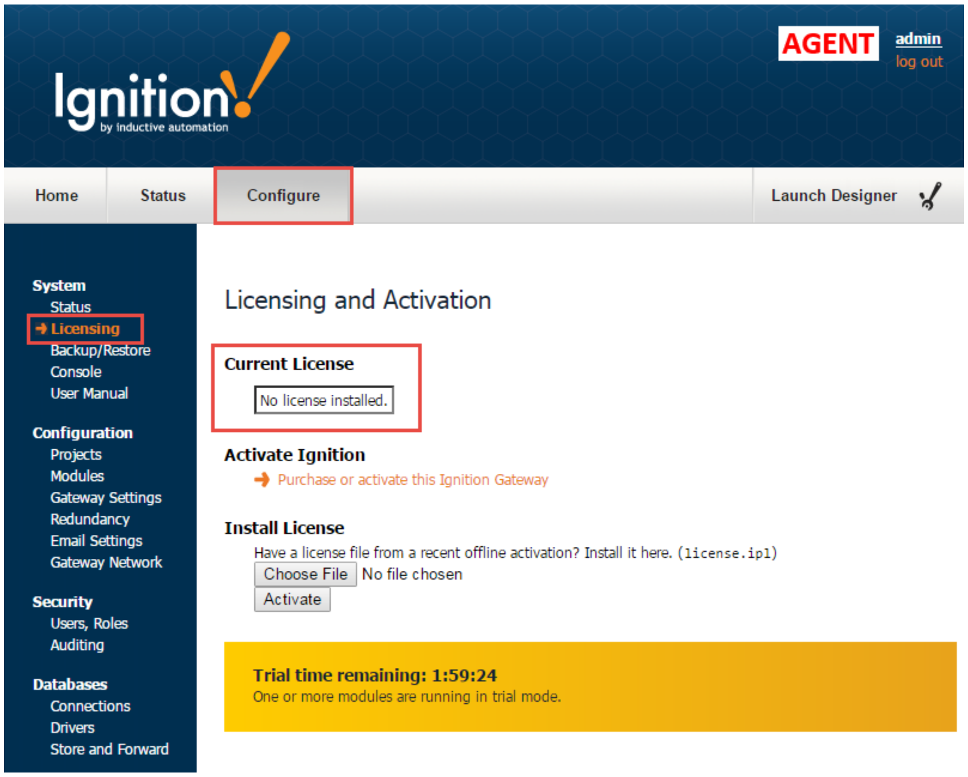
Watch the Video

Before Activating a License

- This section assumes you already have your Controller and Agent created.
- Make sure you know your License Key serial number.
- For clarity, each screenshot is identified with either a Controller or Agent label in the header so it's clear which machine you are working on.
- Refer to Configuring Controllers and Agents for some more useful tips with this section.

Activate License

1. On your **Agent**, go to the **Configure** tab and select **Licensing**. You can see that no license is currently installed.



Licensing and Activation

Current License

No license installed.

Activate Ignition

Purchase or activate this Ignition Gateway

Install License

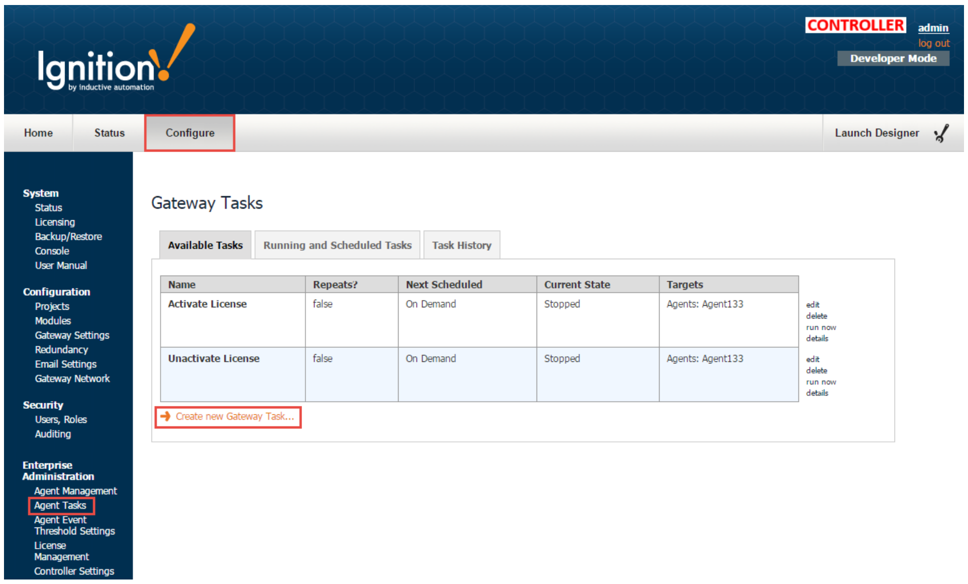
Have a license file from a recent offline activation? Install it here. (license.ipl)

No file chosen

Trial time remaining: 1:59:24

One or more modules are running in trial mode.

2. Activating a license on an Agent must be done from the Controller. Go to your **Controller** and select **Configure > Enterprise Administration > Agent Tasks**. Click on the **'Create new Gateway Task'** link.



System
- Status
- Licensing
- Backup/Restore
- Console
- User Manual

Configuration
- Projects
- Modules
- Gateway Settings
- Redundancy
- Email Settings
- Gateway Network

Security
- Users, Roles
- Auditing

Enterprise Administration
- Agent Management
- Agent Tasks
- Agent Event Threshold Settings
- License Management
- Controller Settings

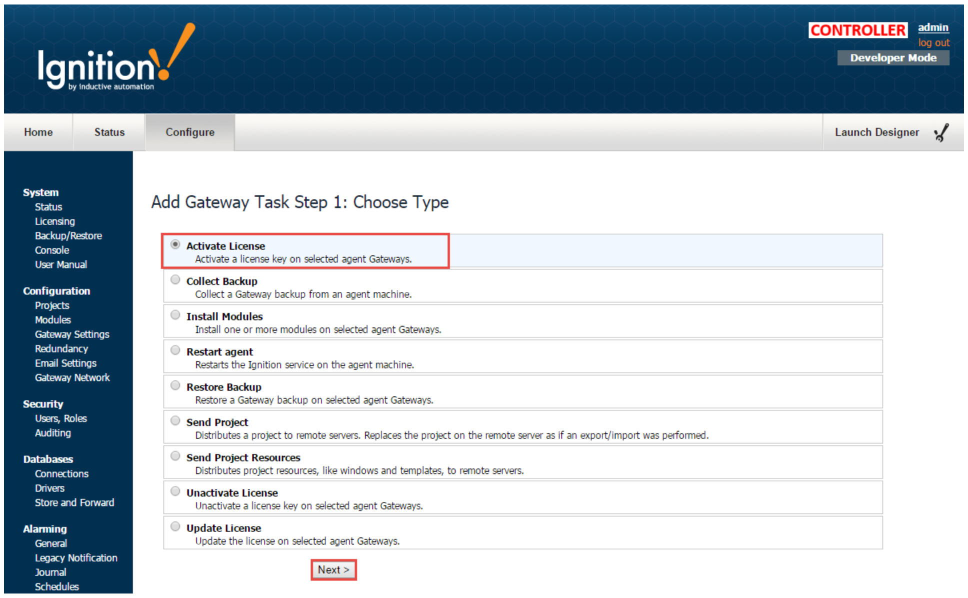
Gateway Tasks

Available Tasks | Running and Scheduled Tasks | Task History

Name	Repeats?	Next Scheduled	Current State	Targets	
Activate License	false	On Demand	Stopped	Agents: Agent133	edit delete run now details
Unactivate License	false	On Demand	Stopped	Agents: Agent133	edit delete run now details

Create new Gateway Task...

3. Activating a License key is one of several tasks you can perform under **Gateway Tasks**. Click the **Activate License** button and press **Next**. The EAM wizard will walk you through each step of the activation process.



System
- Status
- Licensing
- Backup/Restore
- Console
- User Manual

Configuration
- Projects
- Modules
- Gateway Settings
- Redundancy
- Email Settings
- Gateway Network

Security
- Users, Roles
- Auditing

Databases
- Connections
- Drivers
- Store and Forward

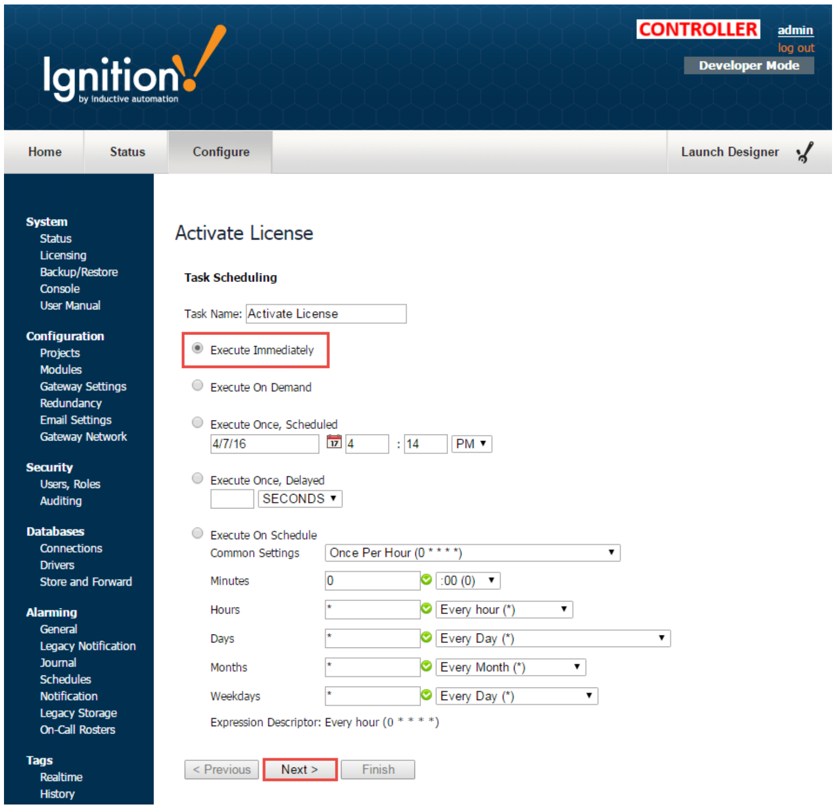
Alarming
- General
- Legacy Notification
- Journal
- Schedules

Add Gateway Task Step 1: Choose Type

- ● **Activate License**
  Activate a license key on selected agent Gateways.
- ○ **Collect Backup**
  Collect a Gateway backup from an agent machine.
- ○ **Install Modules**
  Install one or more modules on selected agent Gateways.
- ○ **Restart agent**
  Restarts the Ignition service on the agent machine.
- ○ **Restore Backup**
  Restore a Gateway backup on selected agent Gateways.
- ○ **Send Project**
  Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- ○ **Send Project Resources**
  Distributes project resources, like windows and templates, to remote servers.
- ○ **Unactivate License**
  Unactivate a license key on selected agent Gateways.
- ○ **Update License**
  Update the license on selected agent Gateways.

Next >

4. The Activate License task has several scheduling options. For this example, select **Execute Immediately** and press **Next**.



Activate License

Task Scheduling

Task Name: Activate License

● Execute Immediately

○ Execute On Demand

○ Execute Once, Scheduled
4/7/16 4 : 14 PM

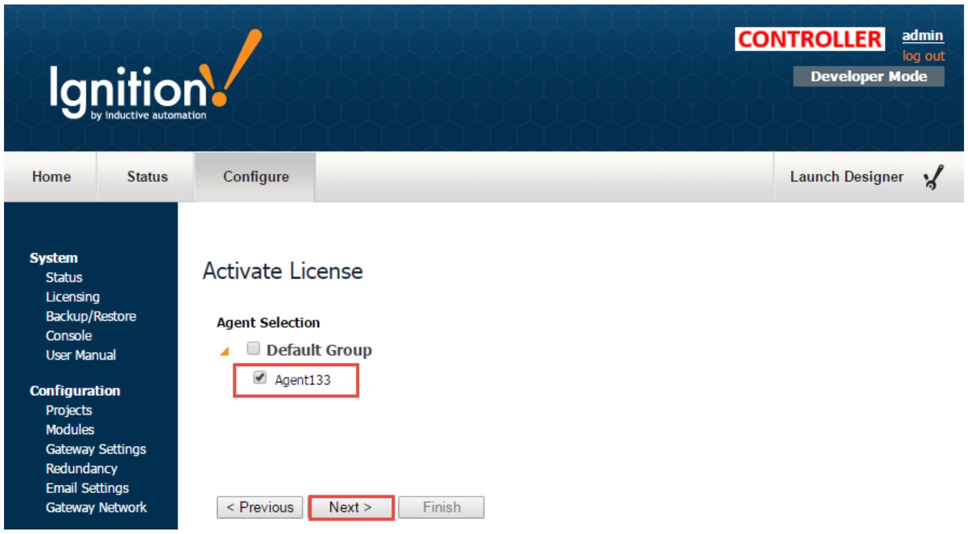
○ Execute Once, Delayed
SECONDS

○ Execute On Schedule

Common Settings	Once Per Hour (0 * * * *)	
Minutes	0	:00 (0)
Hours	*	Every hour (*)
Days	*	Every Day (*)
Months	*	Every Month (*)
Weekdays	*	Every Day (*)

Expression Descriptor: Every hour (0 * * * *)

5. Now, select the **Agent** you want to activate the license on and press **Next**. In this example, Agent133 was selected.



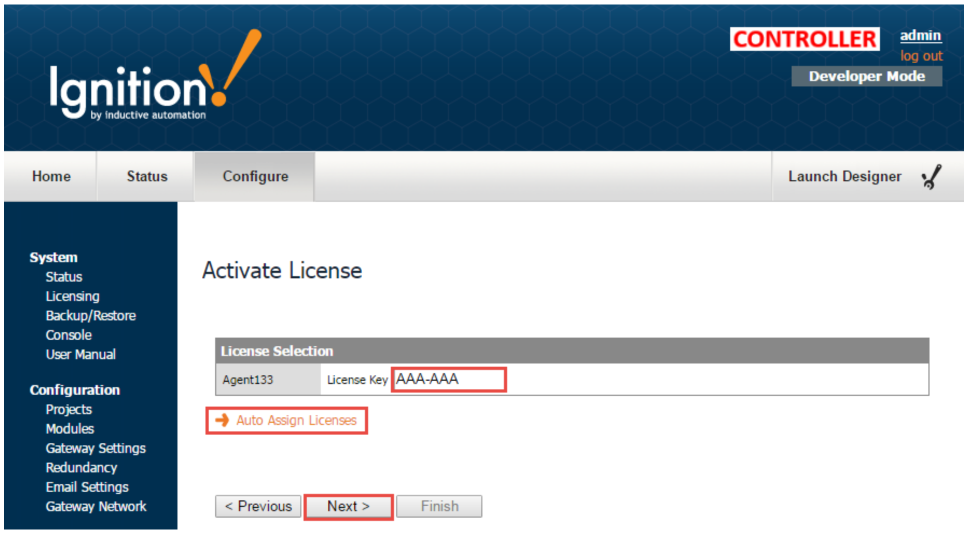
Activate License

Agent Selection

☐ Default Group

☑ Agent133

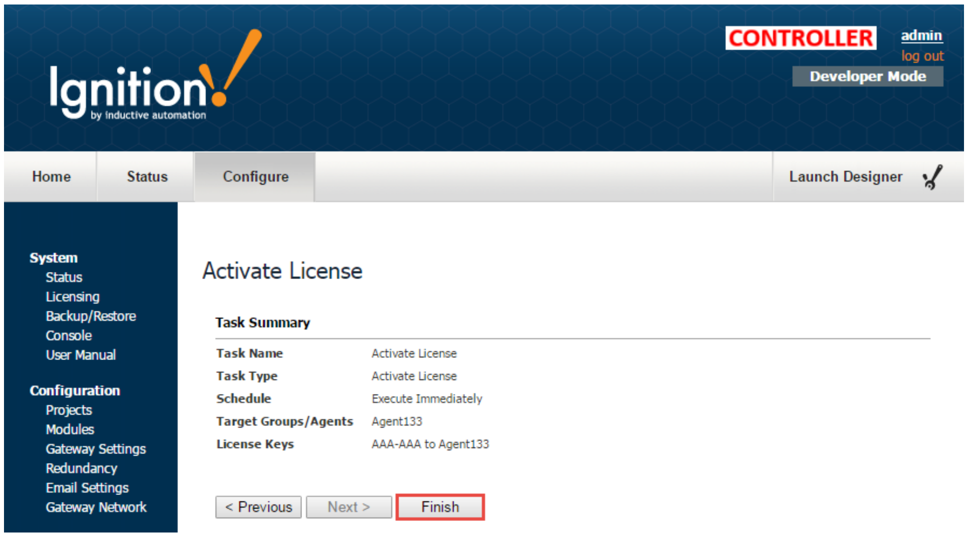
6. Enter your license key. You can enter it one of two ways: type it in the **License Key field** if you know it, or press **Auto Assign the License** and the license key will auto populate. **Press Next**.



Activate License

License Selection	
Agent133	License Key AAA-AAA

7. The Activate License page will refresh so you can validate your licensing information. Press **Finish**.

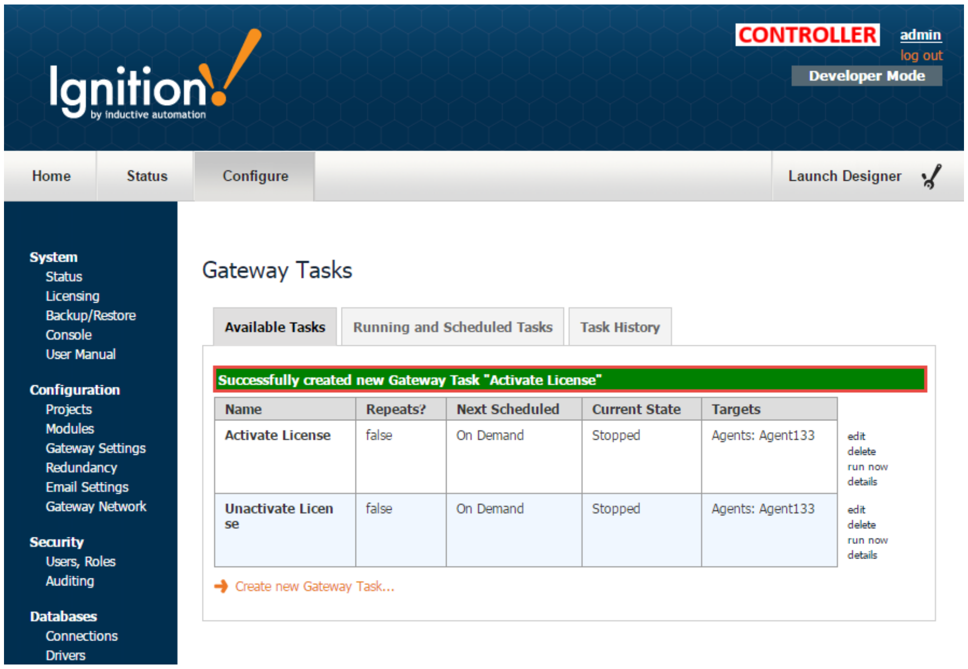


Activate License

Task Summary

Task Name	Activate License
Task Type	Activate License
Schedule	Execute Immediately
Target Groups/Agents	Agent133
License Keys	AAA-AAA to Agent133

The screen will refresh to the Gateway Tasks window. A message will be displayed stating that the license was activated successfully.



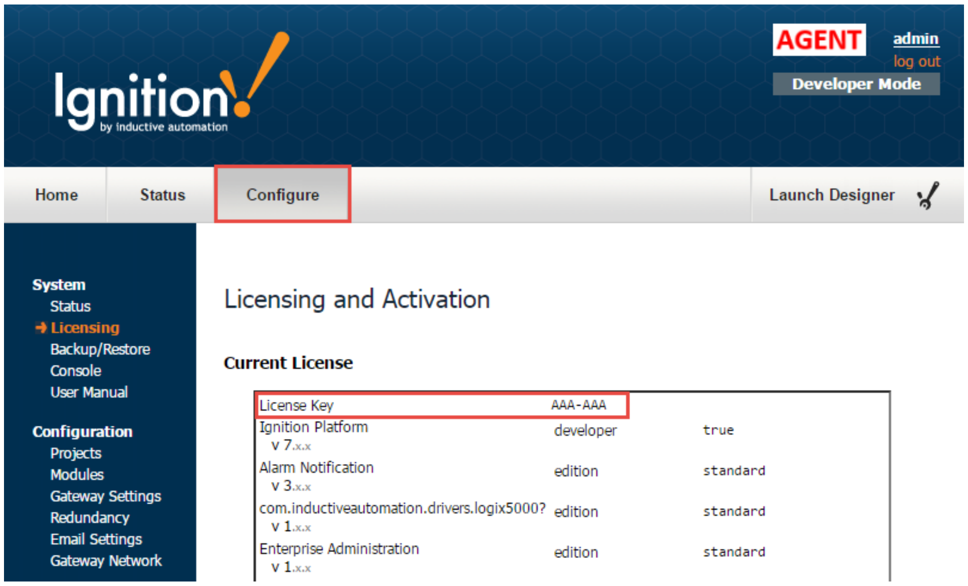
Gateway Tasks

Successfully created new Gateway Task "Activate License"

Name	Repeats?	Next Scheduled	Current State	Targets	
Activate License	false	On Demand	Stopped	Agents: Agent133	edit delete run now details
Unactivate License	false	On Demand	Stopped	Agents: Agent133	edit delete run now details

8. Now go to the **Agent** to validate that the Agent license was activated. You may see an Access Denied message. This is because the page was automatically updated. Don't panic! Click on the **'Return to home page'** link to refresh the Ignition Home page on your Agent.

9. Click on **Configure > Licensing** and verify your License Key on the Agent. This page shows all the modules on your license as well as the versions they are licensed for.



Licensing and Activation

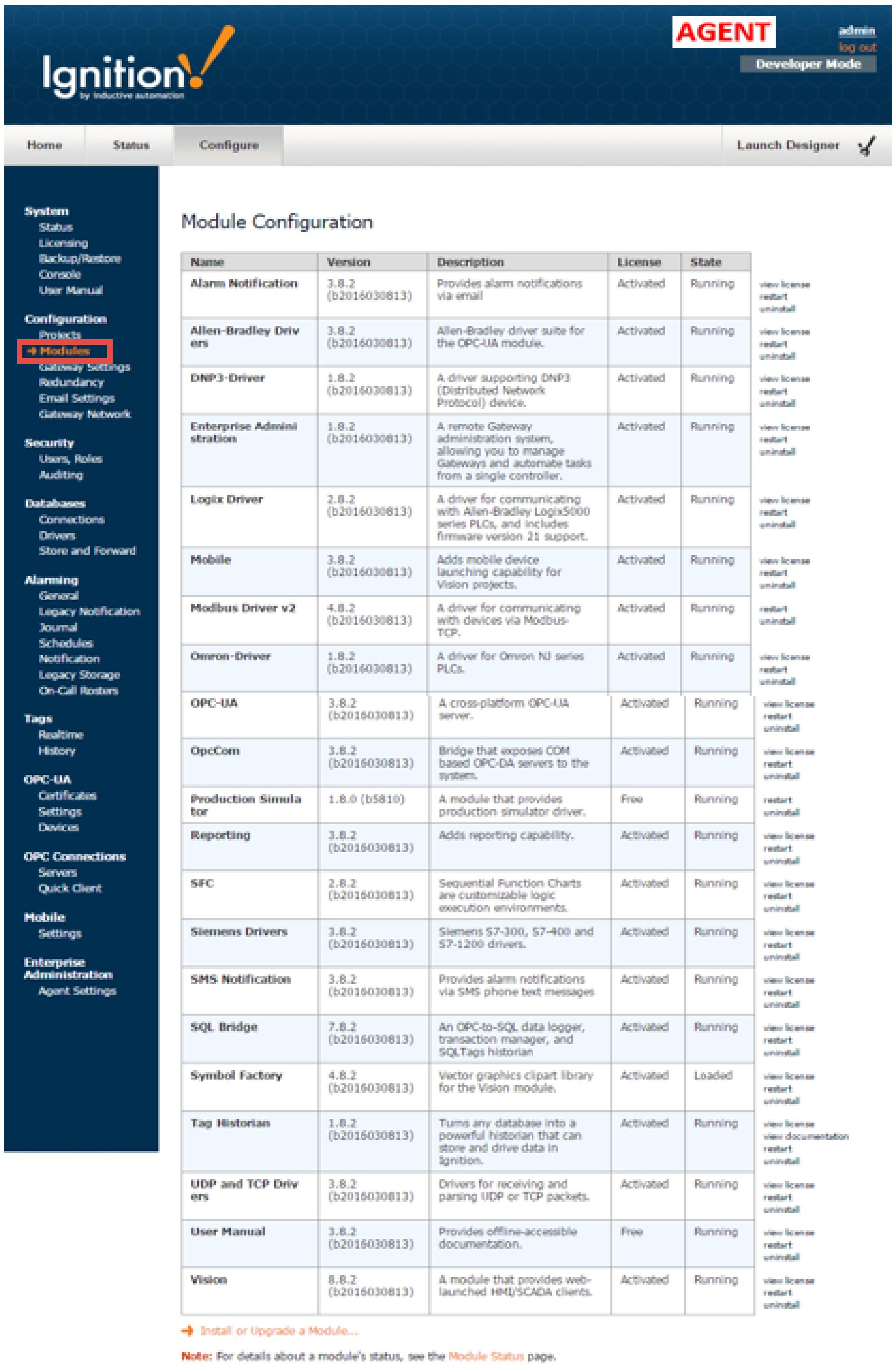
Current License

License Key	AAA-AAA	
Ignition Platform v 7.x.x	developer	true
Alarm Notification v 3.x.x	edition	standard
com.inductiveautomation.drivers.logix5000? v 1.x.x	edition	standard
Enterprise Administration v 1.x.x	edition	standard

If you are running Ignition in Trial Mode

If you are running Ignition in Trial Mode on your Agent, there is a chance that the trial timer expired, and the license activation will fail. If this happens, reset the trial timer and resume the activation for the Agent. You can check the status of the installed modules by going to **Configure > Modules**. Modules you are licensed for will not expire, only the modules that are being used in Trial Mode.

10. It's a good idea to check to see if all your modules are Activated. Go to your **Configure** tab and select **Modules**. If any are still in trial mode, that means your license key either didn't include those modules or they are a newer version than your license allows.



Module Configuration

Name	Version	Description	License	State	
Alarm Notification	3.8.2 (b2016030813)	Provides alarm notifications via email	Activated	Running	view license restart uninstall
Allen-Bradley Drivers	3.8.2 (b2016030813)	Allen-Bradley driver suite for the OPC-UA module.	Activated	Running	view license restart uninstall
DNP3-Driver	1.8.2 (b2016030813)	A driver supporting DNP3 (Distributed Network Protocol) device.	Activated	Running	view license restart uninstall
Enterprise Administration	1.8.2 (b2016030813)	A remote Gateway administration system, allowing you to manage Gateways and automate tasks from a single controller.	Activated	Running	view license restart uninstall
Logix Driver	2.8.2 (b2016030813)	A driver for communicating with Allen-Bradley Logix5000 series PLCs, and includes firmware version 21 support.	Activated	Running	view license restart uninstall
Mobile	3.8.2 (b2016030813)	Adds mobile device launching capability for Vision projects.	Activated	Running	view license restart uninstall
Modbus Driver v2	4.8.2 (b2016030813)	A driver for communicating with devices via Modbus-TCP.	Activated	Running	restart uninstall
Omron-Driver	1.8.2 (b2016030813)	A driver for Omron NJ series PLCs.	Activated	Running	view license restart uninstall
OPC-UA	3.8.2 (b2016030813)	A cross-platform OPC-UA server.	Activated	Running	view license restart uninstall
OpcCom	3.8.2 (b2016030813)	Bridge that exposes COM based OPC-DA servers to the system.	Activated	Running	view license restart uninstall
Production Simulator	1.8.0 (b5810)	A module that provides production simulator driver.	Free	Running	restart uninstall
Reporting	3.8.2 (b2016030813)	Adds reporting capability.	Activated	Running	view license restart uninstall
SFC	2.8.2 (b2016030813)	Sequential Function Charts are customizable logic execution environments.	Activated	Running	view license restart uninstall
Siemens Drivers	3.8.2 (b2016030813)	Siemens S7-300, S7-400 and S7-1200 drivers.	Activated	Running	view license restart uninstall
SMS Notification	3.8.2 (b2016030813)	Provides alarm notifications via SMS phone text messages	Activated	Running	view license restart uninstall
SQL Bridge	7.8.2 (b2016030813)	An OPC-to-SQL data logger, transaction manager, and SQLTags historian	Activated	Running	view license restart uninstall
Symbol Factory	4.8.2 (b2016030813)	Vector graphics clipart library for the Vision module.	Activated	Loaded	view license restart uninstall
Tag Historian	1.8.2 (b2016030813)	Turns any database into a powerful historian that can store and drive data in Ignition.	Activated	Running	view license view documentation restart uninstall
UDP and TCP Drivers	3.8.2 (b2016030813)	Drivers for receiving and parsing UDP or TCP packets.	Activated	Running	view license restart uninstall
User Manual	3.8.2 (b2016030813)	Provides offline-accessible documentation.	Free	Running	view license restart uninstall
Vision	8.8.2 (b2016030813)	A module that provides web-launched HMI/SCADA clients.	Activated	Running	view license restart uninstall

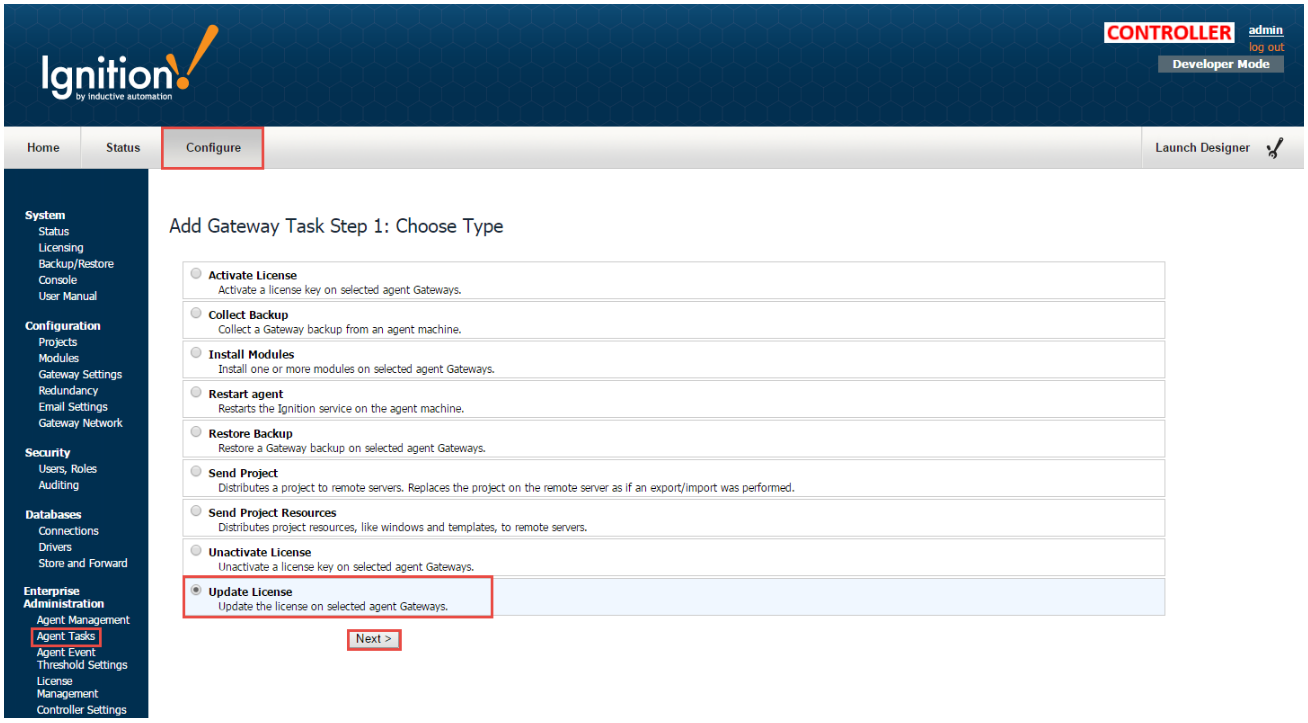
Install or Upgrade a Module...

Note: For details about a module's status, see the Module Status page.

Update a License

In the event you need to reload or update a license on your Agent, go to the **Controller**, navigate to **Configure > Enterprise Administration > Agent Tasks**. Click the **'Create new Gateway'** link.

Select **Update License** and press **Next**. The wizard will guide you through the license update steps. Refer to steps 4 through 10.



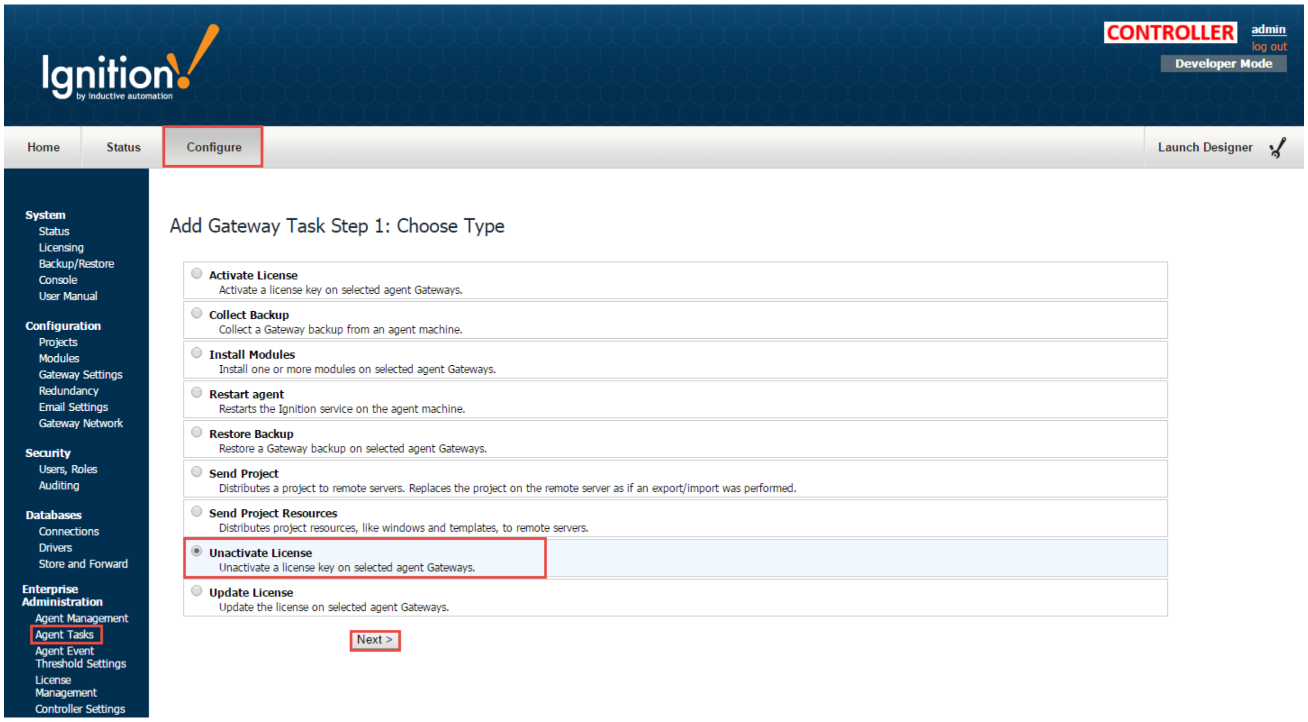
Add Gateway Task Step 1: Choose Type

- **Activate License** – Activate a license key on selected agent Gateways.
- **Collect Backup** – Collect a Gateway backup from an agent machine.
- **Install Modules** – Install one or more modules on selected agent Gateways.
- **Restart agent** – Restarts the Ignition service on the agent machine.
- **Restore Backup** – Restore a Gateway backup on selected agent Gateways.
- **Send Project** – Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- **Send Project Resources** – Distributes project resources, like windows and templates, to remote servers.
- **Unactivate License** – Unactivate a license key on selected agent Gateways.
- [✓ **Update License**] – Update the license on selected agent Gateways.

Unactivate a License

If you need to unactivate a license, go to your **Controller**, navigate to **Configure > Enterprise Administration > Agent Tasks**. Click the **'Create new Gateway Task'** link.

Select **Unactivate License** and press **Next**. Once again, the wizard will guide you through the unactivate license steps. Refer to steps 4 through 9.



Add Gateway Task Step 1: Choose Type

- **Activate License** – Activate a license key on selected agent Gateways.
- **Collect Backup** – Collect a Gateway backup from an agent machine.
- **Install Modules** – Install one or more modules on selected agent Gateways.
- **Restart agent** – Restarts the Ignition service on the agent machine.
- **Restore Backup** – Restore a Gateway backup on selected agent Gateways.
- **Send Project** – Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- **Send Project Resources** – Distributes project resources, like windows and templates, to remote servers.
- [✓ **Unactivate License**] – Unactivate a license key on selected agent Gateways.
- **Update License** – Update the license on selected agent Gateways.

Next...

- Agent Task - Backup and Restore

Agent Task - Backup and Restore

INDUCTIVE UNIVERSITY

Agent Task - Backup and Restore

Watch the Video

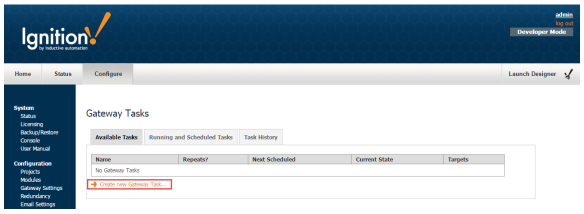
The Backup and Restore tasks are an extremely important part of the Enterprise Administration Module. Performing a Collect Backup is required if you want to do an Agent recovery.

The Gateway Tasks we are going to discuss in this section are the Collect Backup, Restore Backup, Install Modules and Restart Agent.

Collect Backup

Collect Backup is a Gateway Task that performs a Gateway backup on the selected Agent's machine.

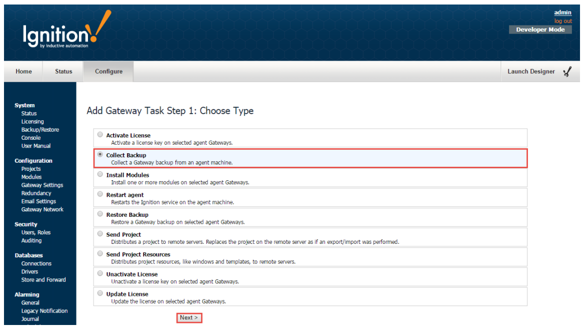
1. On the **Controller**, navigate to **Configure > Enterprise Administration > Agent Tasks**, and click on **Create new Gateway Task**.



Gateway Tasks

Name	Repeats?	Next Scheduled	Current State	Targets
No Gateway Tasks				

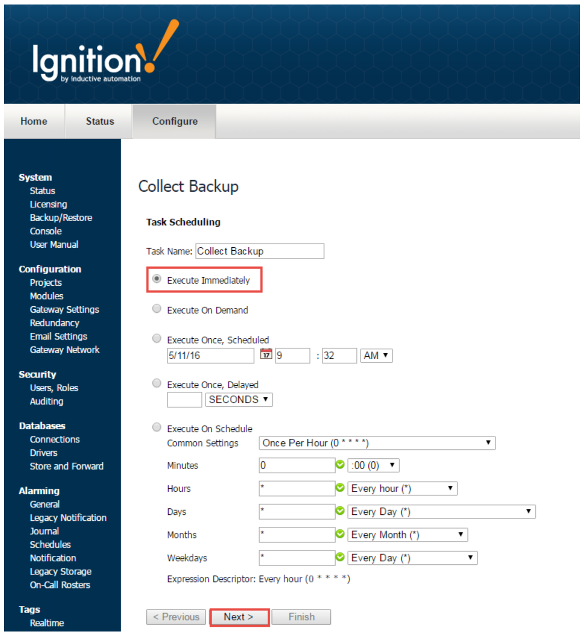
2. Collect Backup is one of the several Gateway tasks you can perform. Select **Collect Backup,** and press **Next**. The EAM task wizard will walk you through each step of the Collect Backup process.



Add Gateway Task Step 1: Choose Type

- Activate License — Activate a license key on selected agent Gateways.
- [✓ Collect Backup] — Collect a Gateway backup from an agent machine.
- Install Modules — Install one or more modules on selected agent Gateways.
- Restart agent — Restarts the Ignition service on the agent machine.
- Restore Backup — Restore a Gateway backup on selected agent Gateways.
- Send Project — Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- Send Project Resources — Distributes project resources, like windows and templates, to remote servers.
- Unactivate License — Unactivate a license key on selected agent Gateways.
- Update License — Update the license on selected agent Gateways.

3. There are several different task scheduling options available. For this Collect Backup example, choose to **Execute Immediately**, and press **Next**. If you need more information on other scheduling options, refer to Task Scheduling.



Collect Backup

Task Scheduling

Task Name: Collect Backup

[✓ Execute Immediately]

Execute On Demand

Execute Once, Scheduled
5/11/16 9 : 32 AM

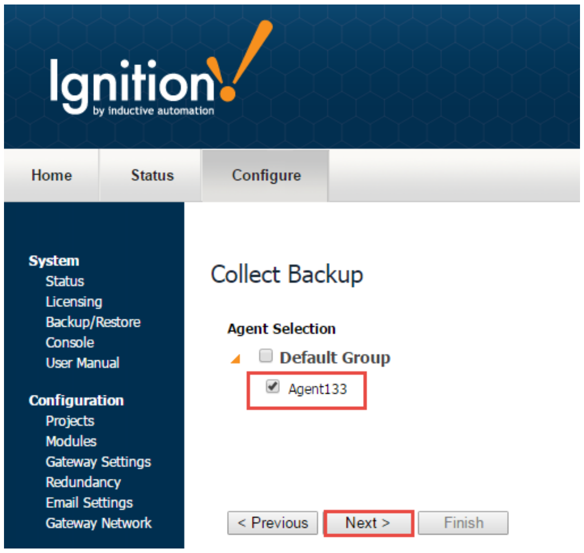
Execute Once, Delayed
SECONDS

Execute On Schedule

Common Settings	Once Per Hour (0 * * * *)	
Minutes	0	:00 (0)
Hours	*	Every hour (*)
Days	*	Every Day (*)
Months	*	Every Month (*)
Weekdays	*	Every Day (*)

Expression Descriptor: Every hour (0 * * * *)

4. Select the Agent you want to perform the Collect Backup on, and press **Next.**



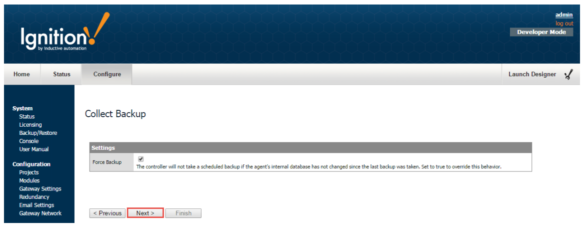
Collect Backup

Agent Selection

Default Group

[✓ Agent133]

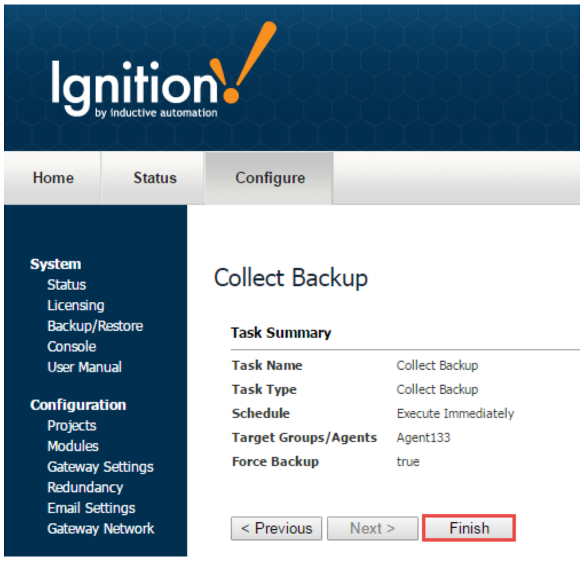
5. The EAM task wizard will prompt you if you want to do a Force Backup even if nothing changed since the last backup. This step is optional, but it is useful when your backups are performed on a regular schedule. It captures the changes between the time the last backup was performed and the unscheduled backup. To be safe, check the **Force Backup** box, and press **Next**.



Collect Backup

Settings	
Force Backup	✓ The controller will not take a scheduled backup if the agent's internal database has not changed since the last backup was taken. Set to true to override this behavior.

6. Review the **Collect Backup Task Summary** information, and press **Finish**.

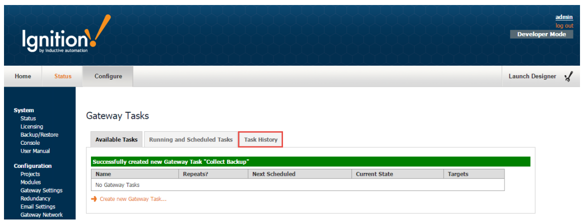


Collect Backup

Task Summary

Task Name	Collect Backup
Task Type	Collect Backup
Schedule	Execute Immediately
Target Groups/Agents	Agent133
Force Backup	true

7. Once the Collect Backup task is complete, a successful message will be displayed. You can even check the history by opening the Task History tab.

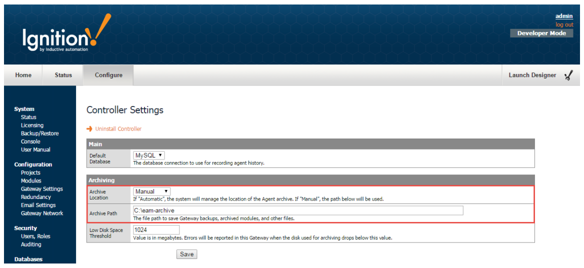


Gateway Tasks

Successfully created new Gateway Task "Collect Backup"

Name	Repeats?	Next Scheduled	Current State	Targets
No Gateway Tasks				

Your new Gateway backup will be stored wherever your Controller settings are pointing to when the Controller was initially created. To check, go to the **Controller,** and navigate to **Configure > Enterprise Administration > Controller Settings**. You can have your Archive Path set to Automatic or Manual. If you have it set to Automatic, you can find your backups inside the directory for Ignition, otherwise, it will be the folder location that is specified in the Controller Settings. To learn more about controller settings, refer to Creating a Controller.



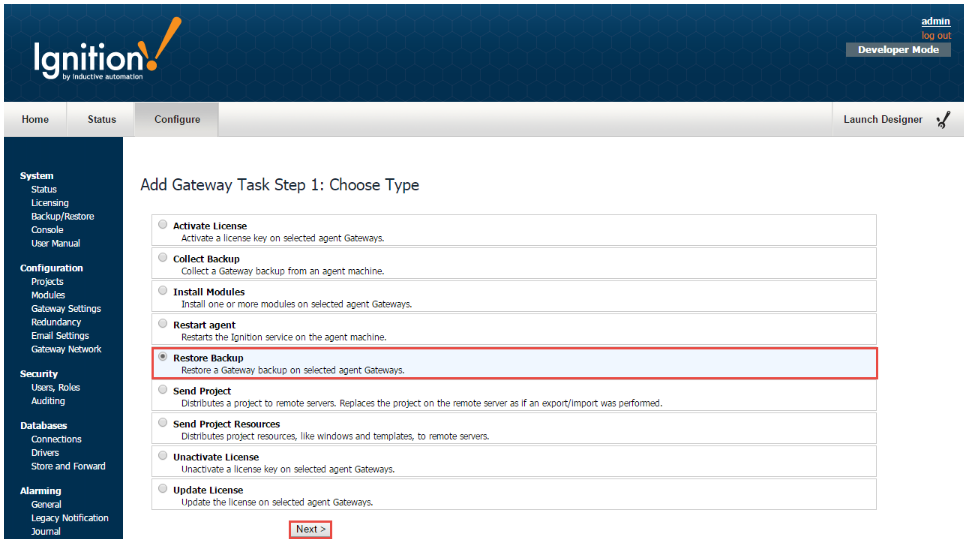
Controller Settings

Main	
Default Database	MySQL The database connection to use for recording agent history.
Archiving	
Archive Location	Manual If "Automatic", the system will manage the location of the Agent archive. If "Manual", the path below will be used.
Archive Path	C:\eam-archive The file path to save Gateway backups, archived modules, and other files.
Low Disk Space Threshold	1024 Value is in megabytes. Errors will be reported in this Gateway when the disk used for archiving drops below this value.

Restore Backup

Now, let's restore a Gateway backup on the selected Agent.

1. Go to the **Controller,** and navigate to **Configure > Enterprise Administration > Agent Tasks**. Click on the **Create new Gateway Task** link. Select the **Restore Backup** task, and press **Next**.



Add Gateway Task Step 1: Choose Type

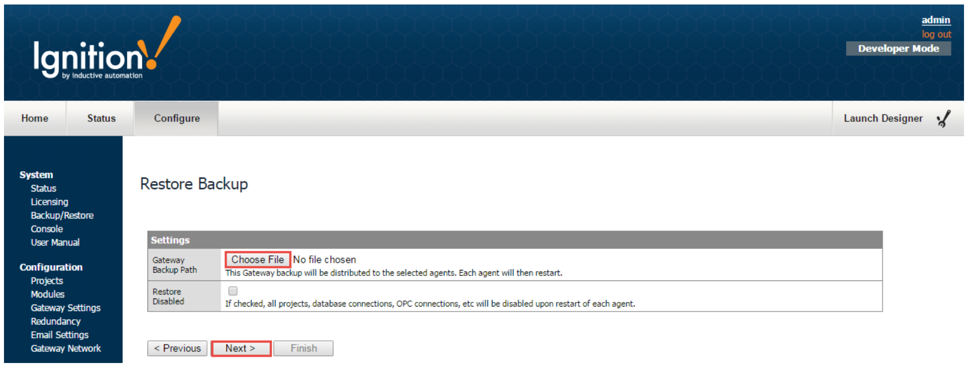
- ○ **Activate License** — Activate a license key on selected agent Gateways.
- ○ **Collect Backup** — Collect a Gateway backup from an agent machine.
- ○ **Install Modules** — Install one or more modules on selected agent Gateways.
- ○ **Restart agent** — Restarts the Ignition service on the agent machine.
- ● **Restore Backup** — Restore a Gateway backup on selected agent Gateways.
- ○ **Send Project** — Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- ○ **Send Project Resources** — Distributes project resources, like windows and templates, to remote servers.
- ○ **Unactivate License** — Unactivate a license key on selected agent Gateways.
- ○ **Update License** — Update the license on selected agent Gateways.

Next >

Screenshots

Some screenshots will not be duplicated. You can refer to the screenshots in Collect Backup as you work your way through each section.

2. Schedule the **Restore Backup** task to **Execute Immediately**, and press **Next**.
3. Select the **Agent**, and press **Next**.
4. If your Archive Path was set to Manual when you created your Controller, you will have to select the **Choose File** button, and navigate to the folder on your computer to locate the Gateway Backup file. If you are unsure about your Archive Path, refer to your Controller Settings.

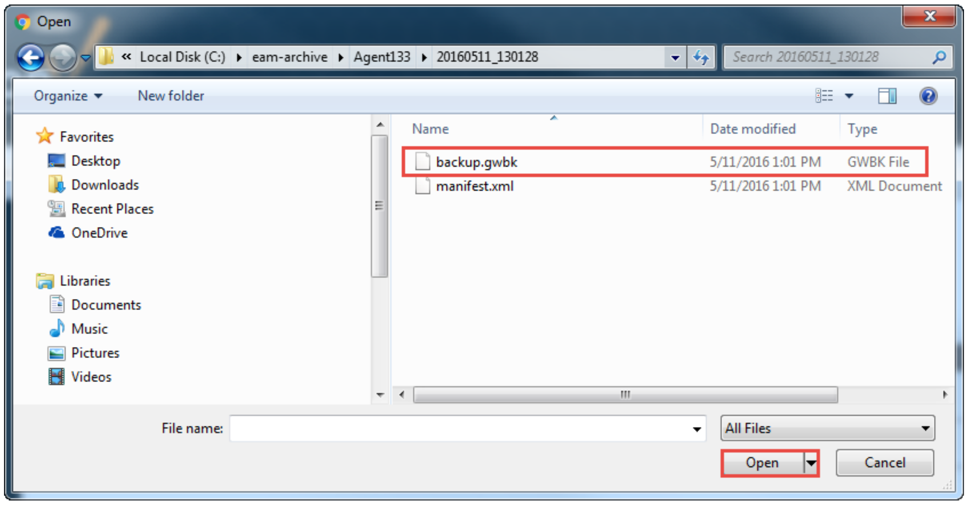


Restore Backup

Settings	
Gateway Backup Path	Choose File No file chosen This Gateway backup will be distributed to the selected agents. Each agent will then restart.
Restore Disabled	☐ If checked, all projects, database connections, OPC connections, etc will be disabled upon restart of each agent.

< Previous | Next > | Finish

Navigate to the folder on your computer to locate the Gateway Backup file. Select the **backup.gwbk,** and click **Open.** Press **Next** on the Restore Backup task.



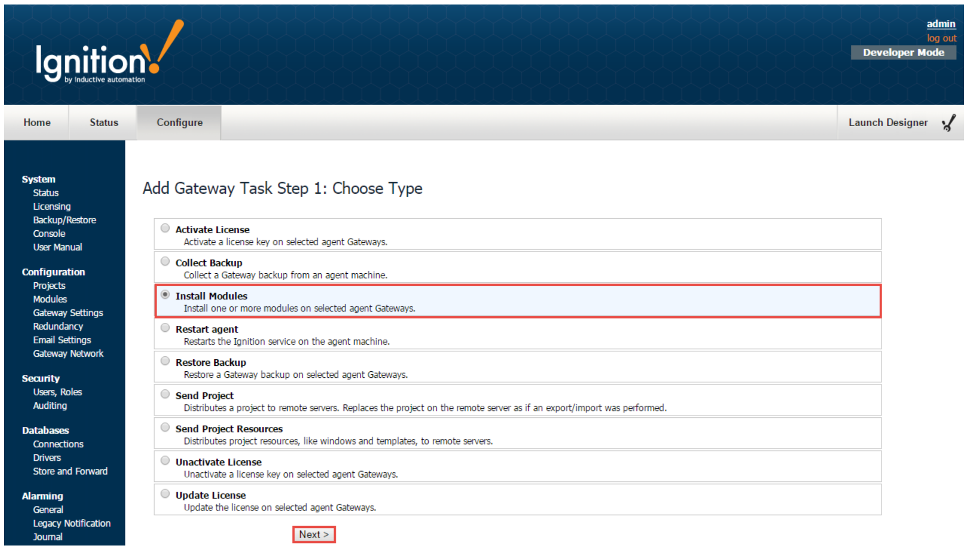
Name	Date modified	Type
backup.gwbk	5/11/2016 1:01 PM	GWBK File
manifest.xml	5/11/2016 1:01 PM	XML Document

5. Review the **Restore Backup Task Summary**, and press **Finish.** Once the Restore Backup task completes, you will receive a successful message from the Gateway Task.
6. Now go to your **Agent,** and click **Home**. It takes a few moments to bring up the Gateway. You will see a progress bar while the Gateway is starting up. Once it's complete, you will see all your projects uploaded and ready to go.

Install Modules

As an example, suppose you accidentally uninstalled the Symbol Factory and Vision Modules from your Agent. Let's go ahead and reinstall them using the Install Modules task.

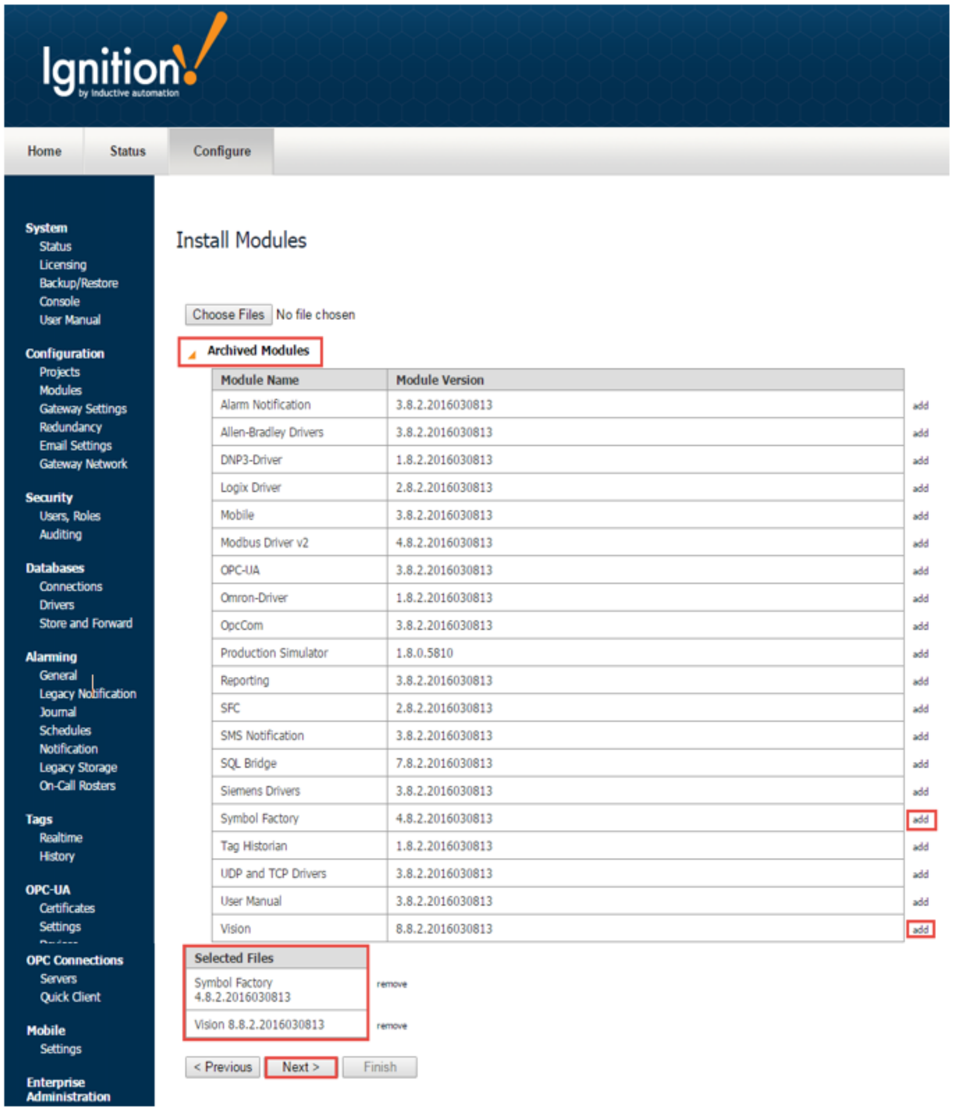
1. Go to the **Controller,** and navigate to **Configure > Enterprise Administration > Agent Tasks**. Click the **Create new Gateway Task** link. Select the **Install Module** task, and press **Next**.



Add Gateway Task Step 1: Choose Type

- **Activate License** – Activate a license key on selected agent Gateways.
- **Collect Backup** – Collect a Gateway backup from an agent machine.
- [✓ **Install Modules**] – Install one or more modules on selected agent Gateways.
- **Restart agent** – Restarts the Ignition service on the agent machine.
- **Restore Backup** – Restore a Gateway backup on selected agent Gateways.
- **Send Project** – Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- **Send Project Resources** – Distributes project resources, like windows and templates, to remote servers.
- **Unactivate License** – Unactivate a license key on selected agent Gateways.
- **Update License** – Update the license on selected agent Gateways.

2. Schedule the **Install Module** task to **Execute Immediately**, and press **Next**.
3. Select the **Agent**, and press **Next**.
4. From the list of **Archived Modules**, click the **Add** buttons for **Symbol Factory** and the **Vision Module**. You will notice, that once you click the **Add** button next to a module, it will show up in the **Selected Modules** box. You can choose as many modules as you need, and when you're finished, press **Next**.

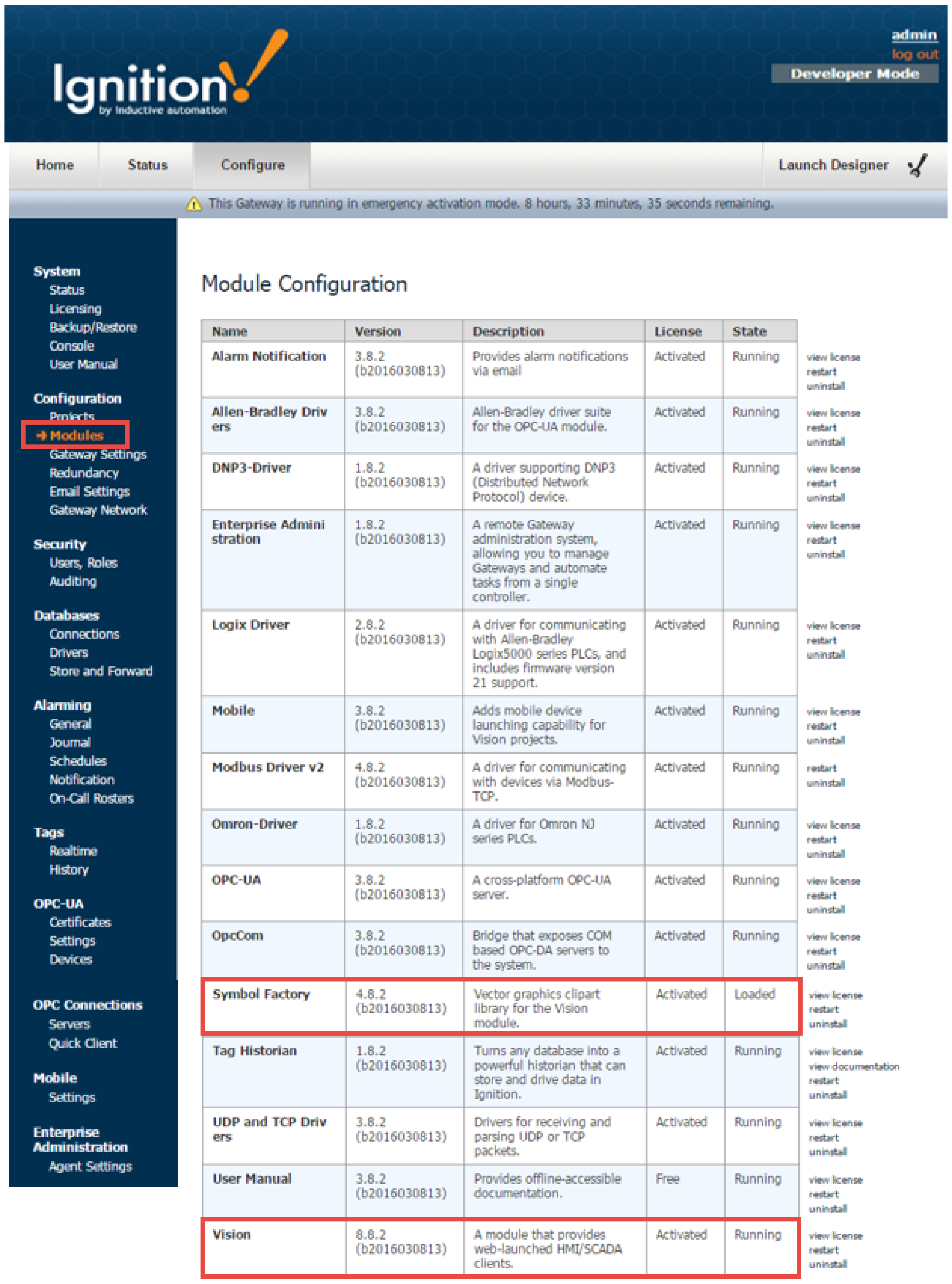


Install Modules

Module Name	Module Version	
Alarm Notification	3.8.2.2016030813	add
Allen-Bradley Drivers	3.8.2.2016030813	add
DNP3-Driver	1.8.2.2016030813	add
Logix Driver	2.8.2.2016030813	add
Mobile	3.8.2.2016030813	add
Modbus Driver v2	4.8.2.2016030813	add
OPC-UA	3.8.2.2016030813	add
Omron-Driver	1.8.2.2016030813	add
OpcCom	3.8.2.2016030813	add
Production Simulator	1.8.0.5810	add
Reporting	3.8.2.2016030813	add
SFC	2.8.2.2016030813	add
SMS Notification	3.8.2.2016030813	add
SQL Bridge	7.8.2.2016030813	add
Siemens Drivers	3.8.2.2016030813	add
Symbol Factory	4.8.2.2016030813	add
Tag Historian	1.8.2.2016030813	add
UDP and TCP Drivers	3.8.2.2016030813	add
User Manual	3.8.2.2016030813	add
Vision	8.8.2.2016030813	add

Selected Files	
Symbol Factory 4.8.2.2016030813	remove
Vision 8.8.2.2016030813	remove

5. Review the **Install Modules Task Summary**, and press **Finish**. Once the Install Module task completes, you will get a successful message from the Gateway Task.
6. Go back to your **Agent**, and navigate to **Configure > Modules** to verify that both the Symbol Factory and Vision Modules were successfully installed.



This Gateway is running in emergency activation mode. 8 hours, 33 minutes, 35 seconds remaining.

Module Configuration

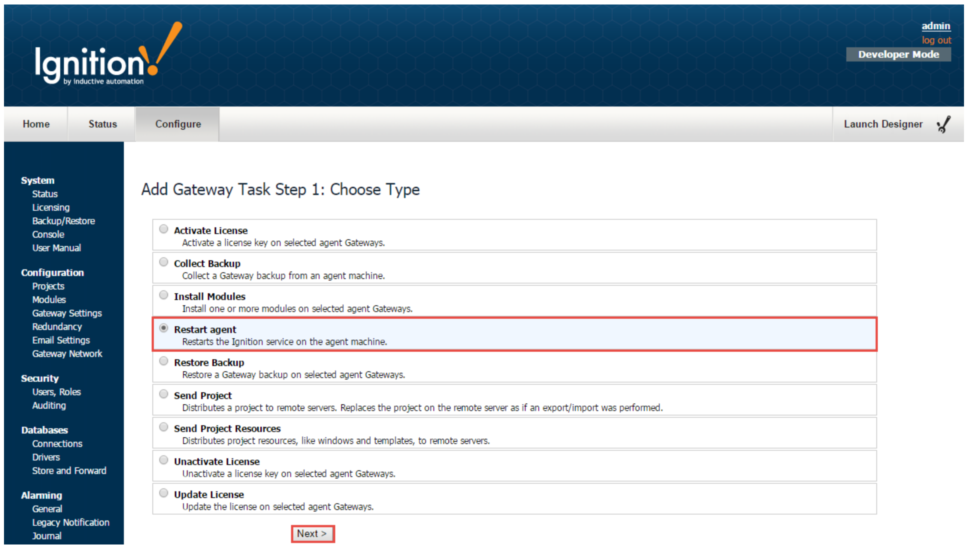
Name	Version	Description	License	State	
Alarm Notification	3.8.2 (b2016030813)	Provides alarm notifications via email	Activated	Running	view license restart uninstall
Allen-Bradley Drivers	3.8.2 (b2016030813)	Allen-Bradley driver suite for the OPC-UA module.	Activated	Running	view license restart uninstall
DNP3-Driver	1.8.2 (b2016030813)	A driver supporting DNP3 (Distributed Network Protocol) device.	Activated	Running	view license restart uninstall
Enterprise Administration	1.8.2 (b2016030813)	A remote Gateway administration system, allowing you to manage Gateways and automate tasks from a single controller.	Activated	Running	view license restart uninstall
Logix Driver	2.8.2 (b2016030813)	A driver for communicating with Allen-Bradley Logix5000 series PLCs, and includes firmware version 21 support.	Activated	Running	view license restart uninstall
Mobile	3.8.2 (b2016030813)	Adds mobile device launching capability for Vision projects.	Activated	Running	view license restart uninstall
Modbus Driver v2	4.8.2 (b2016030813)	A driver for communicating with devices via Modbus-TCP.	Activated	Running	restart uninstall
Omron-Driver	1.8.2 (b2016030813)	A driver for Omron NJ series PLCs.	Activated	Running	view license restart uninstall
OPC-UA	3.8.2 (b2016030813)	A cross-platform OPC-UA server.	Activated	Running	view license restart uninstall
OpcCom	3.8.2 (b2016030813)	Bridge that exposes COM based OPC-DA servers to the system.	Activated	Running	view license restart uninstall
Symbol Factory	4.8.2 (b2016030813)	Vector graphics clipart library for the Vision module.	Activated	Loaded	view license restart uninstall
Tag Historian	1.8.2 (b2016030813)	Turns any database into a powerful historian that can store and drive data in Ignition.	Activated	Running	view license view documentation restart uninstall
UDP and TCP Drivers	3.8.2 (b2016030813)	Drivers for receiving and parsing UDP or TCP packets.	Activated	Running	view license restart uninstall
User Manual	3.8.2 (b2016030813)	Provides offline-accessible documentation.	Free	Running	view license restart uninstall
Vision	8.8.2 (b2016030813)	A module that provides web-launched HMI/SCADA clients.	Activated	Running	view license restart uninstall

Restart Agent

Let's go back to the **Controller** and create one more new Gateway Task called Restart Agent. This task will stop the Ignition service on the Agent and start it back up again.

1. Go to the **Controller,** and navigate to **Configure > Enterprise Administration > Agent Tasks**. Click on the **Create new Gateway Task** link.

   Select the **Restart Agent,** and press **Next.**



Add Gateway Task Step 1: Choose Type

- ○ **Activate License** Activate a license key on selected agent Gateways.
- ○ **Collect Backup** Collect a Gateway backup from an agent machine.
- ○ **Install Modules** Install one or more modules on selected agent Gateways.
- [✓ ● **Restart agent** Restarts the Ignition service on the agent machine.]
- ○ **Restore Backup** Restore a Gateway backup on selected agent Gateways.
- ○ **Send Project** Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- ○ **Send Project Resources** Distributes project resources, like windows and templates, to remote servers.
- ○ **Unactivate License** Unactivate a license key on selected agent Gateways.
- ○ **Update License** Update the license on selected agent Gateways.

2. Schedule the **Restart Agent** task to **Execute Immediately**, and press **Next**.
3. Select the **Agent,** and press **Next**.
4. Review the **Restart Agent Task Summary**, and press **Finish**. Once the Restart Agent task completes, you will receive a successful message from the Gateway Task.
5. Go to your **Agent** and refresh your page. You will see a progress bar while the Gateway is starting up. This will take a few moments, and once this task completes, your Agent will be up and running.

Next...

- Agent Task - Send Project

Agent Task - Send Project

Agent Tasks have the ability to send projects and project resources from a Controller to an Agent.

INDUCTIVE UNIVERSIT

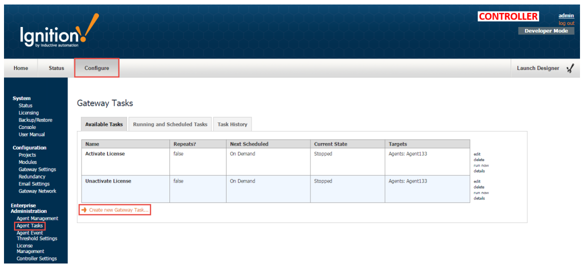
Agent Task - Send Project

Watch the Video

Send Project Task

On the **Controller**, under the **Configure** tab go to **Enterprise Administration > Agent Tasks**. Here you can create a Gateway Task to send a project from the Controller to an Agent.

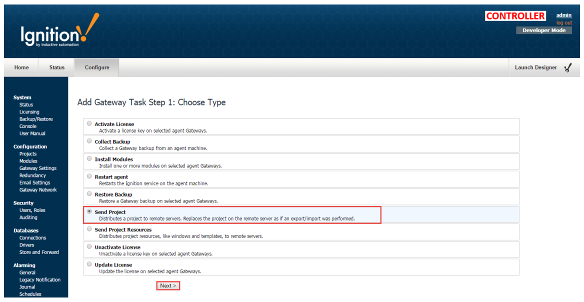
1. The following example demonstrates how to create a Gateway Task and send a project to an agent. Click the **'Create new Gateway Task'** link. The EAM wizard will guide you through each step of the Send Project task.



Gateway Tasks

Name	Repeats?	Next Scheduled	Current State	Targets
Activate License	false	On Demand	Stopped	Agents: Agent133
Unactivate License	false	On Demand	Stopped	Agents: Agent133

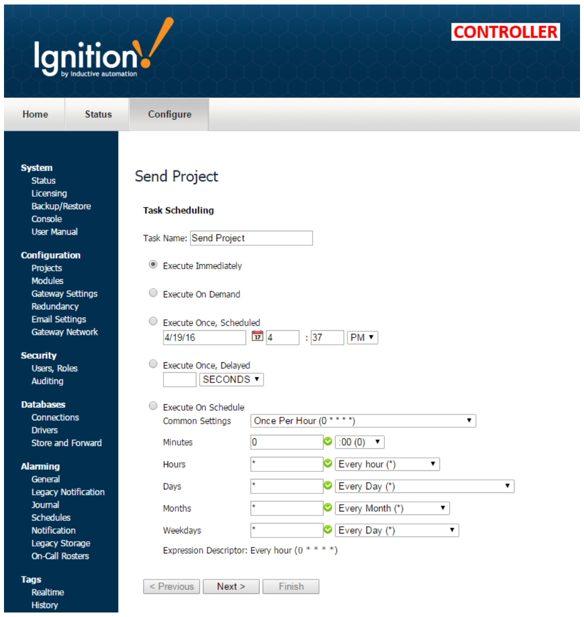
2. Not only will the Send Project send a project from the Controller to the Agent, but if the project already exists on the Agent, it will replace it.
Scroll down the list of Gateway Tasks and select **Send Project**. Press **Next**.



Add Gateway Task Step 1: Choose Type

- ○ **Activate License** Activate a license key on selected agent Gateways.
- ○ **Collect Backup** Collect a Gateway backup from an agent machine.
- ○ **Install Modules** Install one or more modules on selected agent Gateways.
- ○ **Restart agent** Restarts the Ignition service on the agent machine.
- ○ **Restore Backup** Restore a Gateway backup on selected agent Gateways.
- ● **Send Project** Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- ○ **Send Project Resources** Distributes project resources, like windows and templates, to remote servers.
- ○ **Unactivate License** Unactivate a license key on selected agent Gateways.
- ○ **Update License** Update the license on selected agent Gateways.

3. Just like with all Gateway Tasks, the Send Project task needs to be scheduled. Schedule the task to **Execute Immediately** and press **Next**.



Send Project

Task Scheduling

Task Name: Send Project

● Execute Immediately

○ Execute On Demand

○ Execute Once, Scheduled
4/19/16 | 4 : 37 | PM

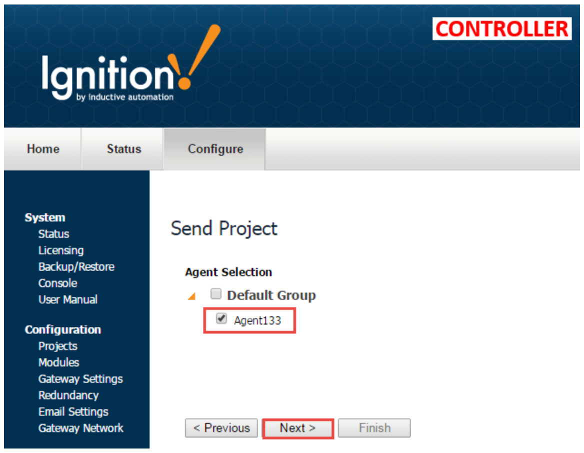
○ Execute Once, Delayed
SECONDS

○ Execute On Schedule

Common Settings	Once Per Hour (0 * * * *)	
Minutes	0	:00 (0)
Hours	*	Every hour (*)
Days	*	Every Day (*)
Months	*	Every Month (*)
Weekdays	*	Every Day (*)

Expression Descriptor: Every hour (0 * * * *)

4. Select the Agent where you want to send your project and press **Next**.



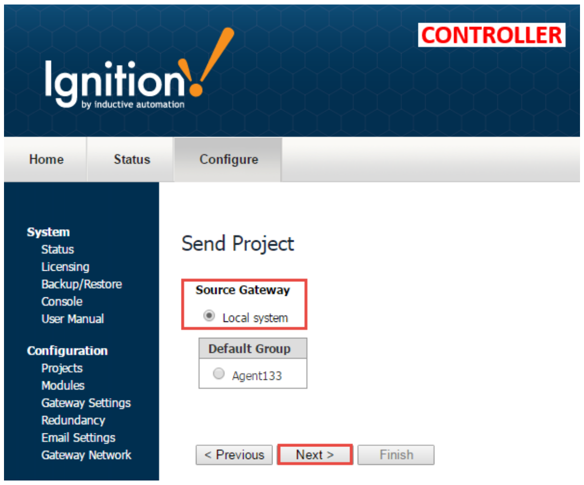
Send Project

Agent Selection

Default Group

[✓ Agent133]

5. The EAM knows your controller and agent configuration. It's going to prompt you to choose the machine where your project is located. In this example, the project that you want to send is located on the Controller or Gateway source so select **Local System** and press **Next**.



Send Project

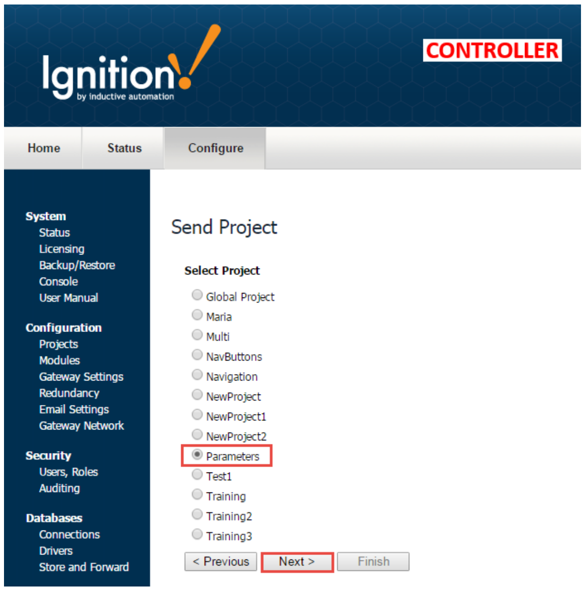
Source Gateway

[✓ Local system]

Default Group

Agent133

6. The wizard displays all your projects from your local system. Select the project you want to send to your Agent. In this example, select the **Parameters** project and press **Next**.
(If you select your Agent on the previous screen, the wizard will display all your projects running on your Agent that were sent over previously from the Controller).

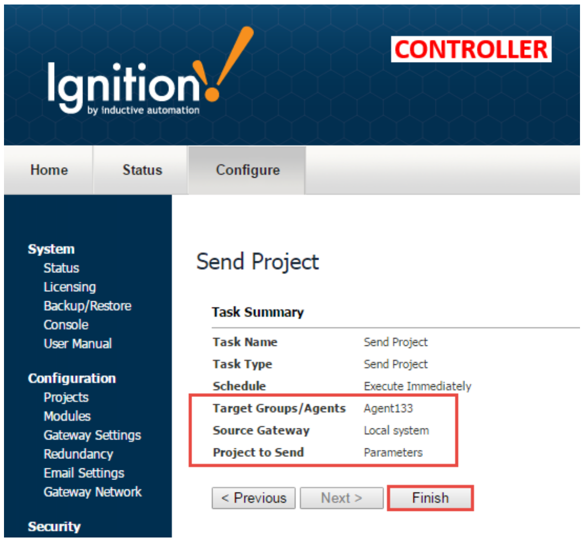


Send Project

Select Project

Global Project
Maria
Multi
NavButtons
Navigation
NewProject
NewProject1
NewProject2
[✓ Parameters]
Test1
Training
Training2
Training3

7. The EAM wizard summarizes all the information for you to review. Press **Finish** and the Parameters project will be sent from the **Controller** to your **Agent.**

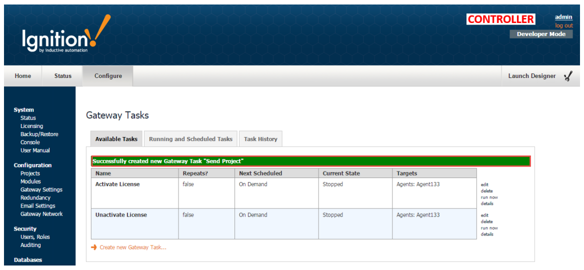


Send Project

Task Summary

Task Name	Send Project
Task Type	Send Project
Schedule	Execute Immediately
Target Groups/Agents	Agent133
Source Gateway	Local system
Project to Send	Parameters

8. The EAM will execute the task, the screen will refresh, and a message will popup stating that your Send Project task was created successfully.



Gateway Tasks

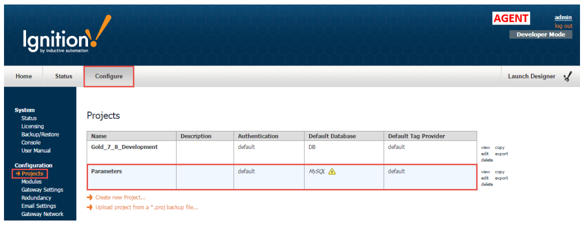
Successfully created new Gateway Task "Send Project"

Name	Repeats?	Next Scheduled	Current State	Targets	
Activate License	false	On Demand	Stopped	Agents: Agent133	edit delete run now details
Unactivate License	false	On Demand	Stopped	Agents: Agent133	edit delete run now details

9. To verify that the Parameters project was sent, go to your **Agent,** click on the **Configure** tab and **select Projects**. You can see that the Parameters project was created by the EAM controller. If you can't see your sent project, refresh the screen by clicking **Projects**.

MySQL Warning Message

Notice that for the Parameters project there is an warning icon next to MySQL database. This means that MySQL database was used on the Controller and currently is not configured on the Agent.



Projects

Name	Description	Authentication	Default Database	Default Tag Provider	
Gold_7_8_Development		default	DB	default	view copy edit export delete
Parameters		default	MySQL ⚠	default	view copy edit export delete

Send Project Resources Task

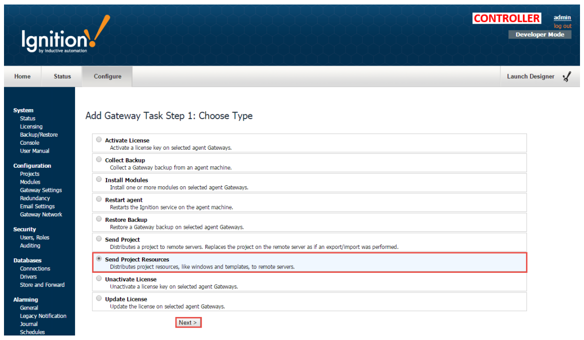
You can also send project resources, such as project templates, windows, transaction groups, pipelines and even script modules from the Controller to an Agent.

1. On the **Controller**, under the **Configure** tab, go to **Enterprise Administration > Agent Tasks**. Let's create another Gateway Task to send resources from the Controller to the Agent.   Click the **'Create new Gateway Task'** link. Once again, the EAM wizard will guide you through each

step of the Send Project Resources task.

> **Reference screenshots in the Send Project section**
>
> You can refer to some of the screenshots in the Send Project Task section as you work your way through the examples in this Send Project Resources Task.

2. Click on the **Send Project Resources** button and press **Next.**



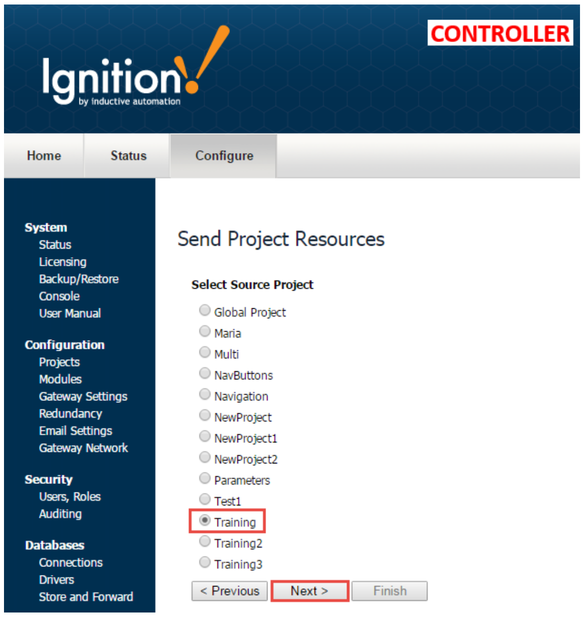
Add Gateway Task Step 1: Choose Type

- ○ **Activate License** — Activate a license key on selected agent Gateways.
- ○ **Collect Backup** — Collect a Gateway backup from an agent machine.
- ○ **Install Modules** — Install one or more modules on selected agent Gateways.
- ○ **Restart agent** — Restarts the Ignition service on the agent machine.
- ○ **Restore Backup** — Restore a Gateway backup on selected agent Gateways.
- ○ **Send Project** — Distributes a project to remote servers. Replaces the project on the remote server as if an export/import was performed.
- [✓ **Send Project Resources**] — Distributes project resources, like windows and templates, to remote servers.
- ○ **Unactivate License** — Unactivate a license key on selected agent Gateways.
- ○ **Update License** — Update the license on selected agent Gateways.

3. Schedule the Send Project Resources task to **Execute Immediately** and press **Next**.

4. Let's select the resources from a project on the Controller. Select **Local System** and press **Next**
.

5. This example sends selected resources from the Training project on the Controller to the Agent. Select **Training** and press **Next**.

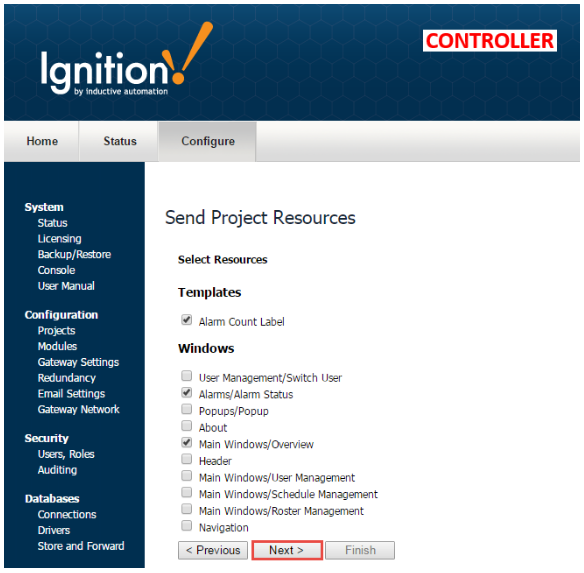


Send Project Resources

Select Source Project

- ○ Global Project
- ○ Maria
- ○ Multi
- ○ NavButtons
- ○ Navigation
- ○ NewProject
- ○ NewProject1
- ○ NewProject2
- ○ Parameters
- ○ Test1
- [✓ Training]
- ○ Training2
- ○ Training3

6. You can see all the resources inside of the Training Project.   Of course, real projects will have many more useful project resources than are shown here. It can include templates, windows, transaction groups, pipelines and script modules. Choose whatever resources you want to send to the agent, and press **Next**.



Send Project Resources

Select Resources

Templates

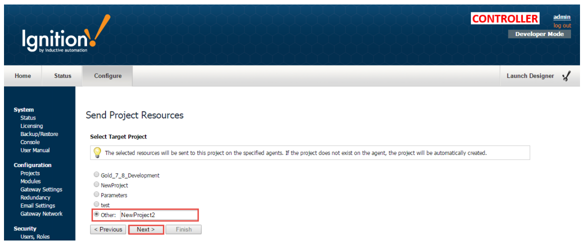
- [✓] Alarm Count Label

Windows

- [ ] User Management/Switch User
- [✓] Alarms/Alarm Status
- [ ] Popups/Popup
- [ ] About
- [✓] Main Windows/Overview
- [ ] Header
- [ ] Main Windows/User Management
- [ ] Main Windows/Schedule Management
- [ ] Main Windows/Roster Management
- [ ] Navigation

< Previous | Next > | Finish

7. You have the option of either merging the project resources into an existing project or creating a new project on the Agent. In this example, let's create a new project. Select **Other,** and enter a new project name. Press **Finish**.



Send Project Resources

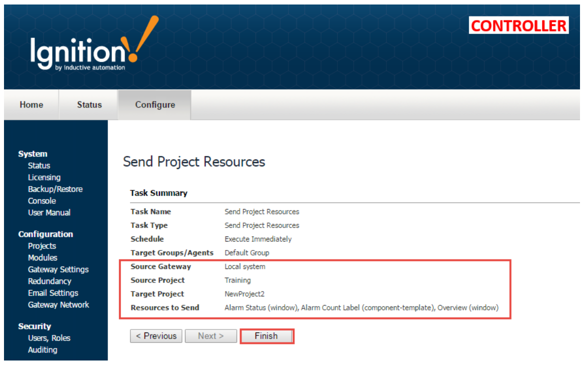
Select Target Project

The selected resources will be sent to this project on the specified agents. If the project does not exist on the agent, the project will be automatically created.

- ○ Gold_7_8_Development
- ○ NewProject
- ○ Parameters
- ○ test
- ● Other: NewProject2

< Previous | Next > | Finish

8. The EAM wizard summarizes all the information for you to review. Press **Finish.**



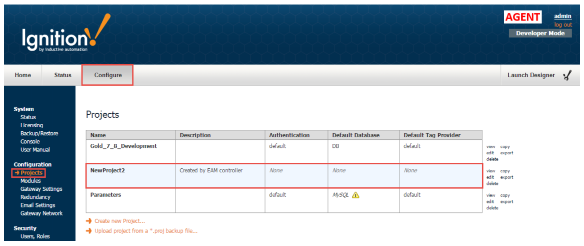
Send Project Resources

Task Summary	
Task Name	Send Project Resources
Task Type	Send Project Resources
Schedule	Execute Immediately
Target Groups/Agents	Default Group
Source Gateway	Local system
Source Project	Training
Target Project	NewProject2
Resources to Send	Alarm Status (window), Alarm Count Label (component-template), Overview (window)

< Previous | Next > | Finish

9. The Gateway Task for Send Project Resources will execute, the screen will refresh, and a successful message will popup stating that your Send Project Resources task was sent.

10. To verify that the Training project resources were sent, go to your **Agent,** click on the **Configure** tab, and **select Projects**. You can see in this example that **'NewProject2'** was created by the EAM controller. If you cannot see your new project, refresh the screen by clicking **Projects**.



Projects

Name	Description	Authentication	Default Database	Default Tag Provider	
Gold_7_8_Development		default	DB	default	view copy edit export delete
NewProject2	Created by EAM controller	None	None	None	view copy edit export delete
Parameters		default	MySQL ⚠	default	view copy edit export delete

Create new Project...

Upload project from a *.proj backup file...

So once your project and project resources are sent to the Agent, you need to complete the installation of your project. You need to setup your authentication profile, database, and tag providers. Once this is completed, you are ready to use the project and project resources on your Agent.

Next...

- Automated Agent Installation

Automated Agent Installation

Automated agent installation

The EAM allows an agent Gateway to be automatically configured from text files the first time the Gateway is started. This capability is enormously useful when installing multiple agent Gateways in your network, as otherwise you must install each Gateway, log in, and install and configure the EAM agent. To automatically configure an agent, you must add two files to the Ignition /data folder before startup: init.properties and eam-install.properties. Example contents are below. After the files have been read, they are renamed to .init.properties.bak and .eam-install.properties.bak. A typical application for this capability would be a script that creates the init files and adds them to an ignition.zip that is distributed to many machines.

Note 1: For init.properties, you are not required to use all the settings shown below. The Gateway will use its standard default settings instead.

Note 2: Multiple Gateway Network connection settings are managed by numbering each setting. The first set of connection settings must contain “0” in the setting name after “gateway.network.” . For example, you can create two outgoing connections with different IP addresses like so:

```
gateway.network.0.Host=10.20.11.18
gateway.network.1.Host=10.20.15.23
```

Example init.properties

```
# SystemName: sets the Gateway name in Gateway Settings

SystemName=Agent1

# AutoDetectLocal: set to true to check the first found network interface to determine local IP address. Set
to false to manually set the IP address of the Gateway.

AutoDetectLocal=false

# LocalInterface: manually sets the IP address of the Gateway. If set, the AutoDetectLocal setting must be set
to false

LocalInterface=10.20.11.17

# UseSSL: set to true to force all connecting web browsers to connect to the Gateway over SSL

UseSSL=false

# PingRate: how often in milliseconds to ping the remote machine

gateway.network.0.PingRate=1000

# Enabled: set to false to disable the gateway connection after it is created

gateway.network.0.Enabled=true

# Host: the address of the remote machine

gateway.network.0.Host=10.20.11.18

# Port: the port of the remote machine.

gateway.network.0.Port=8088

# Enable SSL: set to true to use SSL to connect to the remote machine. Note that you must change the Port
setting to use the Gateway Network SSL port. This port is 8060 by default.

gateway.network.0.EnableSSL=false

# PingTimeout: how long in milliseconds to wait for a ping request to be processed by the remote machine

gateway.network.0.PingTimeout=300

# PingMaxMissed: the number of failed pings allowed before the connection state is set to Faulted.

gateway.network.0.PingMaxMissed=30

# EnableWebSockets: set to true to allow faster asynchronous communications with the remote machine.

gateway.network.0.EnableWebSockets=true
```

```
# WebsocketTimeout: how long to wait in milliseconds for a message to be processed over the web socket

gateway.network.0.WebsocketTimeout=10000

# HttpConnectTimeout: how long to wait in milliseconds to initially connect the http data channel

gateway.network.0.HttpConnectTimeout=10000

# HttpReadTimeout: how long to wait in milliseconds to read data from the http data channel

gateway.network.0.HttpReadTimeout=60000
```

Example eam-install.properties

```
# installSelection: always set this setting to Agent

setup.installSelection=Agent

# controllerServerName: the Gateway Network server name of the controller

agent.controllerServerName=Ubuntu-Controller

# sendStatsInterval: the amount in seconds that the agent will wait before sending new metrics and
configuration data to the controller. Note that if there are currently no interesting metrics or updated
config data to send, the agent will not send any data.

agent.sendStatsInterval=5
```

Next...

- Redundant EAM Configuration (7.8.1)

Redundant EAM Configuration (7.8.1)

Redundant Controller Setup

To set up redundant EAM controllers, you first follow the normal setup process for a redundant master and backup. The main difference with EAM is that each agent requires separate Gateway Network connections to both the master controller and the backup controller. The agent will automatically route messages to the active controller. Some items to keep in mind:

- As with any redundant configuration, you cannot make any changes on the backup controller. This includes approving new agents and running agent tasks.
- Agent metrics are only sent to the active controller. This means that EAM agent system tags will not process data on a backup controller until the backup becomes active.
- Scheduled and running agent tasks on the master controller will be automatically suspended if control is transferred to the backup controller. The tasks will never run on the backup controller. The tasks will resume on the master controller after it resumes control.

Redundant Agent Setup

As with the redundant controller setup, you follow the same redundant master and backup setup process. Then run the Agent Setup wizard in the same way as you set up a standard agent. After the Agent Setup wizard completes, you must navigate to the controller Gateway and approve both the master and backup pending agents. Some items to keep in mind:

- Both the master agent and the backup agent will send agent metrics. The controller will create a Master folder and a Backup folder of system tags, allowing you to create a project that monitors both sets of system tags at the same time.
- Agent tasks list the master agent and the backup agent separately. Tasks that do not change any configuration on an agent can be run without errors on both the master agent and the backup agent (such as the Restart Gateway task and the license tasks). But tasks that do change configuration (such as the Send Project task) will fail on the backup agent, due to the fact that Gateway configuration cannot be changed on a redundant backup.

Versioning

Ignition stores several versions of a project so you can go back to a previous version or test out changes before being sent to the production environment.

In this section ...

Staging and Publishing

Each project can have two distinct versions at once:

- Staging version
- Published version

By default, a new project is configured to be in Auto publish mode, which means that the two versions are always identical. However, if you change a project to be in Manual publish mode, you can then explicitly publish a project in the Designer.

INDUCTIVE UNIVERSITY

Staging and Publishing

Watch the Video

Published vs Staging

Each project can have two distinct versions at the same time: the **Staging** version and the **Published** version. Having a published version and a staging version allows you to make changes to a project, save the changes, and test out the changes before "publishing" it to a production environment.

Usually, Vision module Clients run the published version of a project. However, by launching a Client in a special mode (from the Designer or from the Config section of the Gateway), you can launch a Client that runs the staging version of that project. This staging Client will receive updates on every save, where the production Clients receive updates only on publish. This lets you test out your changes to the project in an actual Client, which is more realistic than the Designer's preview mode.

Not all aspects that comprise a project use this system. It is primarily intended for systems such as the Vision module's Clients. Features that run persistently on the Gateway, such as Tags, the SQL Bridge's Transaction Groups, and Gateway-side scripting always run the most recently saved changes (the Staging version). Since these features by definition must run in exactly one place, they cannot be effectively "tested out" by simultaneously running a staging version alongside a published version.

In the Designer, choose the **Project > Properties** menu item, then go to the **Project > General** page to set the **Publish Mode**. By default, a new project is configured to be in **Auto** publish mode, however you can change it to **Manual** publish mode.

- In the default **Auto** mode, when you want to save your project, Designer lets you do a **File > Save and Publish**, which means both the Staging and the Published versions will be the same and identical.

- In **Manual** mode, when you want to save your project, Designer lets you do both a **File > Save** or a **File > Publish**. **File > Save** updates the Staging version, and **File > Publish** updates the Published version.

Publish Selected Resources

It is also possible to publish individual resources on a project as they are finished instead of publishing the entire project. This is particularly helpful if you want to get resource updates to users quicker rather than waiting for the whole project to be published. To learn more, refer to Project Creation and Publishing.

Project Versioning and History

Each project keeps a log of recent changes. These include both saves and publishes. Every save increments a number called the "edit count" for the project, which can be used like a serial number. The user, time, affected resources, and a commit message (see next) are logged as well.

Next...

- Project Versioning and Rollback

Project Versioning and Rollback

Each project keeps a log of recent changes. These include both saves and publishes. The user, time, affected resources, and a commit message are logged and you can rollback to a previous version. Ignition keeps track of the previous 10 saved versions of your project. You can restore any one of these versions with the roll back feature.

INDUCTIVE UNIVERSITY

Project Versioning and Rollback

Watch the Video

To restore a previous version of your project

1. In the Designer, select **File > Rollback**.
   The Rollback window is displayed showing the edit dates, user name, edit count, and edit type.
2. Select the version you want to roll back to and click **Rollback**.
3. On the **Confirm** dialog box, click **Yes**.
   The Designer reloads the previous version of the project.

Next...

- Production and Development Server

Production and Development Server

Once you complete your project and have it in production, it is a good idea to have a separate Ignition development server for making future changes. That way, the changes will not affect the plant floor and can be completely tested before rolling out to the production environment. Also, it is important to have a development server to perform any Ignition upgrades on so that any problems can be addressed before moving to a production environment.

There are a few things to consider when having a production and development server. First, the development server should have its own PLCs and SQL databases. You do not want the development server to point to the live PLCs because it would be possible to accidentally write values to PLCs that may start or stop equipment. Likewise, you do not want the development server to point to the production database or else you may risk duplicating data.

Second, the development server does not need a license because you can run the server in its two hour trial mode.

Third, you should never move a Gateway backup from the development server to the production server. The development server is pointing to the development PLCs and databases. If you move the Gateway backup, these connections will be saved inside the backup and restored on the production server. Which is bad because the production server should be pointing to the live PLCs and databases. Instead you should backup individual projects and tags separately and move them to the production server.

Ignition projects use database connections and tags but only by name so the production server and the development server will have the same PLC and database names. But behind the scenes, they will point to different places. You cannot simply backup and restore a project from the development server to the production server because the projects contain the windows and the templates and not the tag information. You will have to transport the tags as well in addition to keeping track of changes on the production and the development server to ensure that the tag folder path coincide with the ported project.

INDUCTIVE UNIVERSITY

Production and Development Server

Watch the Video

Next...

Module Descriptions

What are Modules?

Modules are applications that are built on the Ignition platform and integrate into the platform in order to offer functionality. Most of the main features of Ignition are actually provided by different modules such as the Vision and SQL Bridge modules.

Modules integrate seamlessly into the system and provide things like new Designer workspaces, new Gateway settings, new drivers, and much more.

Why Modules?

The modules in Ignition offer many benefits.

- **Flexible Licensing**
  Only license the modules that you need, saving money and reducing complexity compared to big monolithic applications that try to do everything. At the same time, the modules have been designed to offer a broad swath of functionality, to avoid having too many pieces.
- **Hot-Swappable**
  Modules can be dynamically loaded and unloaded, allowing you to install, remove, and upgrade them without affecting other parts of the system. This can have huge implications for big projects where up-time is important.
- **Increase System Stability**
  Building modules on a common platform means fewer bugs, better isolation, and all around increased stability.
- **Pluggable Module Architecture**
  The pluggable module architecture allows quick integration of new modules into the Ignition platform. From time to time new modules will be released which add additional features. Third-party modules that provide a wide range of functionality are also available.

Installing, Upgrading, Uninstalling, or Restarting a Module

All module configuration is done from Gateway in the **Configuration > Modules** section. From here you can add, remove, and restart modules. Modules integrate their settings into the Gateway configuration tree, and therefore do not offer settings in this section.

Installing or Updating a Module

1. On the **Configuration > Modules** page, go to the bottom of the list, find the orange arrow, and click the **Install or Upgrade a Module** link.
2. Click **Choose File**, select the `.modl` file.
3. Click **Install**.
   You can now see the module you installed in the list of modules.

Note: Modules can be running side-by-side in both Trial mode and Activated mode, see About Licensing for more information. Also when installing a module, there is no need to restart the server.

Uninstalling a Module

After you select the module, click the **uninstall** link on the far-right next to the module entry in the table, then click **Confirm**.

Modules are hot-swappable, this means these actions can be performed while the system is running. Furthermore, the isolated nature of modules ensures that performing one of these actions only affects that particular module, and any modules which depend on it. For example, uninstalling the SQL Bridge module will not affect any running Vision module clients.

Restarting a Module

Modules can be restarted by clicking the **restart** button next to their entries on the **Configuration > Modules** page. Because of the isolated nature of modules, the other modules are not affected by the **restart** (unless they depend on that particular module).

Module Status

The installed module list on the **Configuration > Modules** page also provides some basic information about the **State** of the module. The **Version**, **License**, and **State** columns are all displayed in the list. Module licensing is performed centrally in **System > Licensing**, so the values here are only for information purposes.

The module **State** can be either **Running** or **Loaded**. Running means the module programs are actively performing tasks, for example, the Alarm Notification modules shows as Running because it is continually checking for condition changes that need to set the alarm on or off. Loaded means the module program or resources are available but nothing is running, for example, the Symbol Factory module makes all the symbol images available for your use.

Third-Party Modules

Third Party Modules are Ignition modules that were were produced by someone other than Inductive Automation. Our open SDK allows anyone to create and test your own modules. Be aware that loading any modules from an untrusted source could potentially do harm to your projects or your data.

In this section ...

Standard Modules

In Ignition, the Standard modules are those modules that are automatically installed when you choose the **Typical** installation option, they are the core modules.

In this section ...

SQL Bridge Module

The SQL Bridge module is a robust and flexible tool to map between OPC data and Database data. Transaction Groups are the heart of the SQL Bridge module. Transaction groups configured through the Ignition Designer, are units of execution that perform actions such as, storing data historically, synchronizing database values to OPC, and loading recipe values.

Transaction Groups

Transaction groups are configured in the Ignition Designer. Each Transaction Group is associated with a table in a database Ignition is connected to, and is made up of one or more Items. The group will then execute at a specific interval of time, or on a user-defined schedule. Generally each execution will create a new row in the database table with a separate column for each Item in the group. However it is possible for some types of Transaction Groups to take values from the database and write to specific tags.

Additionally the Transaction Group can be configured to conditionally synchronize values by using a trigger. The trigger is evaluated every execution, and if the trigger condition is met, then synchronization will occur. If the trigger condition has not been met then the group will wait until the next execution to re-evaluate the trigger.

There are several types of Transaction Groups, (Standard, Block, Historical and Stored Procedure) and each offers different functionality. For example the Historical Group allows you to quickly configure a group that reads OPC data and push it to the database. While the additional flexibility of a Standard Group allows you take values from the database and write them to

Items

Each Item in the Transaction Group consists of several properties, but the key properties are the Value and Target Name.

- **Value:** The Value can be a reference to an OPC value such as a tag in the Tag Browser. However you can instead use a direct OPC Item Path without having to create a tag in Ignition first. Additionally the Value can be derived using Ignition's Expression Language.
- **Target Name:** In most cases the Target Name is a column on the database table the Transaction Group is associated with. However it is possible to have the Target Name 'Read-only'. When set to 'Read-only' the value of the item will not be tied to any columns in the database, but is still visible from the Transaction Group and can be used as a trigger.

SQL Bridge Limited

The limited version of SQL Bridge only allows the use of Historical Groups. It only passes data from PLC to the database. It is not bi-directional. There is no limit on the number of groups you can create.

For more information about whether a limited license is right for you, contact Inductive Automation.

Vision Module

The Vision module provides the visual elements of Ignition. Vision offers a wide range of functionality, and can be used to create HMI style control systems, data analysis and trending applications, executive dashboards, and more. The projects are designed using the Ignition Designer, and clients are web-launched with zero installation from any Java capable computer.

Create your Ideal Control System in Minutes

Combined with the power of Tags, it's never been easier to build effective status and control systems. Drag and drop tags on the screen to create automatically bound buttons, HOA toggles, LED displays, value entry fields, and more. Drag tags directly onto component properties to bind bi-directionally in seconds. The innovative overlay system provides intuitive data quality feedback with no additional configuration.

Vector Graphics

Powerful vector-graphics drawing tools allow you to create inviting graphics for your project. Vector graphics are screen-resolution independent, allowing screens to look great on any size monitor. Advanced graphics features like gradients, Bézier curves, transparency, are easily accessible with the intuitive drawing tools. Create your own symbols, import them from *.svg files using drag-and-drop, or use the Symbol Factory module to access nearly 4,000 ready to use, professional-quality vector symbols.

World-Class Charting Capabilities

The Ignition Vision module offers a variety of charting and trending options. The Easy Chart, as its name suggests, makes it incredibly easy to create useful and powerful charts. The charts support multiple axes, sub-plots, many pens, and hundreds of thousands of data points. Using Tag Historian, creating charts is as simple as drag-and-drop, and charts intelligently pull just the data they need, making clients more efficient.

Integrated Database Connectivity

The Ignition Vision module is the world's most database friendly HMI/SCADA application. Working with SQL databases is integrated into many aspects of the project design process, allowing you to integrate process and business data effortlessly.

Unlimited Growth Potential

Web-launched clients, the ability to seamlessly connect multiple projects through Retargeting, and no licensing restrictions on screens, tags, components or clients means the system can grow over time.

Vision Limited License

Under Limited Licensing, the Vision Module can support scaling down to running five (or fewer) concurrent Clients. There are no restrictions on the number of tags, providers or projects. The limited version does not come with the Symbol Factory image library.

For more information about whether a limited license is right for you, contact Inductive Automation.

Symbol Factory Module

The Symbol Factory module integrates Symbol Factory 2.5 from Software Toolbox into the Ignition Designer. The intuitive interface allows you to browse and search through nearly 4,000 high quality vector graphics symbols. Once you find the symbol you're looking for, simply drag-and-drop it onto a Vision window to use it right away. Double-click on the symbol to change it, or dynamically animate parts of it to bring your project to life.

Reporting Module

The Reporting module adds dynamic reporting functionality to the Vision module, allowing you to display reports to Vision clients or to generate PDF files. The reporting module offers flexible report generation, with a variety of components, charts and tables. Additionally, it supports the import of existing forms and images, allowing you to migrate from paper-based tracking systems to an electronic system. Easily create dynamic, database-driven PDF reports with pixel-perfect layout. Import existing PDF-based forms and overlay dynamic data on top to achieve a paperless factory.

Dynamic PDF Reports

Data changes quickly, so why not your reports? With the Reporting module, putting together comprehensive reports is a snap. Want a report for a different time range or a different equipment, no problem You can create a report to suit your needs and make it dynamic enough to be changed on the fly without a redesign.

Automatic Reports

Create reports that automatically show your users the information pertinent to their shift, access level, area, or pre-determined times-of-interest.

Utilize Existing Forms

Already have existing forms or reports? You can bring those into the report designer and keep your paperwork consistent.

Paperless Factory

Achieve a greener, paperless facility by using Ignition to generate an unlimited number of PDF reports, which can easily be stored digitally.

Reporting Module Limited Version

Limited Licensing

- With a limited version of the Reporting Module, you can define as many reports as you want, but you will only be able to run (schedule) 2 reports. You must be at your limit or under before the Reporting Module knows which report to run. If you go over the limit, you will get a warning message, and it will not run any reports until you delete enough reports to be at the limit.
- Saving, printing, FTP and email schedules all count as running a report.

On the Vision Module

- A report that you create with the Reporting Module can be displayed using the Vision component either in the Designer or the Client.
- You can print and save a report through scripting. You can also right click on the component to print or save which is very useful in the Client. To learn more, refer to the Report Viewer.

On the Trial Version

- All Report functionality is available.
- There is no limit on the number of reports or how you can view or distribute them.
- The only limitation is the watermark on each page.

For more information about whether a limited license is right for you, contact Inductive Automation.

OPC-UA Module

The Ignition OPC-UA module offers OPC-UA server functionality with a variety of device drivers and a robust, open driver API.

OPC-UA Server Functionality

This module turns Ignition into an OPC-UA server, capable of handling connections from any OPC-UA compatible client. Outgoing connections can also be made to 3rd-party OPC-UA servers. In this way Ignition functions as both an OPC-UA server and an OPC-UA client.

Built-in Device Drivers

The OPC-UA module offers a number of device drivers for common protocols out-of-the box, and is easily expandable thanks to the hot-swappable module architecture in Ignition. New drivers can be downloaded and installed in minutes without requiring a system restart or otherwise affecting other parts of the Ignition platform.

Redundancy Support

The OPC-UA module ties into the Ignition redundancy in order to provide efficient access to device data along with failover redundancy, with no additional configuration.

Public Driver API

Anyone can create new drivers thanks to the open driver API, and users can download and install drivers created by other developers. This is the first time such an API has been made publicly available for a product like Ignition. For more information about creating drivers for Ignition, visit the Inductive Automation website.

Driver Modules

Drivers for the OPC-UA module are deployed as modules themselves. While they don't add a visible element to the system, they are loaded and upgraded in the same manner as other Ignition modules.

Available Drivers:

Allen-Bradley Drivers Module

The Allen-Bradley brand of programmable logic controllers (PLC) is a popular choice in many industries. Ignition can connect to the following Allen-Bradley controllers:

PLC	Capability
CompactLogix	Connect to CompactLogix firmware v20 and prior processors.
ControlLogix	Connect to ControlLogix firmware v20 and prior processors.
Logix Driver	Connect to Allen-Bradley Logix family devices, including devices with firmware v21+.
MicroLogix	Connect to MicroLogix 1100, 1200, and 1400 series PLCs.
PLC5	Connect to PLC5s via Ethernet.
SLC	Connect to SLC 5/05s via Ethernet.

See the following links for detailed step-by-step instructions on how to connect to these PLCs:

- Connecting to CompactLogix
- Connecting to ControlLogix
- Connecting to ControlLogix v21
- Connecting to MicroLogix
- Connecting to PLC5
- Connecting to SLC
- ControlNet Example
- An ENBT Example
- A DH+ Example

Modbus Driver Module

The Modbus Driver module allows the Ignition OPC-UA server to communicate with any device that supports the Modbus protocol, that is, the Modbus TCP and Modbus RTU over TCP protocols.

The Modbus Driver module can connect directly to devices that support Ethernet communications or it can also connect to Modbus devices through a Gateway device.

See the following sections for detailed step-by-step instructions:

- Connecting to Modbus Device
- About Modbus Addressing
- About Modbus Address Mapping

Siemens Drivers Module

The Siemens Drivers module provides support for connecting to S7-300, S7-400, and S7-1200 PLCs via TCP/IP using the S7 protocol.

See the following sections for step-by-step instructions on how to connect to these PLCS:

- Connecting to S7 Devices
- About Siemens Addressing

UDP and TCP Drivers Module

The UDP is a strictly passive listener. The UDP driver is configured to *listen* to one or more ports on a given IP address.

The TCP driver is configured to *connect* to one or more ports on a given IP address. This driver can listen, as well as write back to the connected port.

Rules are configured that dictate how the incoming data is interpreted.

For step-by-step instructions on how to connect to TCP Drivers, see Connecting to TCP Device.

DNP3 Driver Module

The DNP3 module allows the Ignition OPC-UA server to communicate with other devices that support the DNP3 protocol (DNP3 outstations).

To connect Ignition to a DNP3 device

1. Go to the **Configure** section of the **Gateway** webpage.
2. Scroll down and select **OPC-UA > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **DNP3 Driver**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:

   Name:**DNP3**
   Hostname: type the IP address, for example 74.125.224.72
   Check the box for **Show advanced properties?** to see the additional settings, but you can keep all the defaults.

6. Click **Create New Device**.
   The **Devices** page is displayed showing the **DNP3** device is successfully created and added to Ignition. The **Status** will show as Disconnected and then Connected.

Gateway Settings

General	
Name	
Enabled	
Main	
Hostname	
Port	The port used, default is 20000.
Source Address	The address of the master station, default is 3.
Destination Address	The address of the outstation, default is 4.
Integrity Poll Interval	The interval at which to perform an integrity poll, in millis, default is 3,600,000.
Direct Operate Enabled	When true, the Direct-Operate function code is used on a write, otherwise Select-Operate is used, default is true.
Unsolicited Messages Enabled	When true, the outstation may send unsolicited messages for Class 1, 2, and 3 data, default is false.
Advanced	
Message Fragment Size	The maximum size of a message fragment in the application layer, default is 249.
Message Timeout	The amount of time to wait for a message response from the outstation, default is 5,000.
Retries	The number of retries on a message timeout, default is 0.
Link Layer Confirmation	When true, a link layer confirmation will be required from the outstation when sending messages, default is false.

Aliased Points

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

Point Address	The group, variation, and index that fully describe a point. A full address consists of all three parts: • Group – An integer prefixed with `g`. For example, **`g40`** • Variation – An integer prefixed with `v`. For example, **`v2`** • Index – An integer prefixed with `i`. For example, **i5** Example: **`g30v1i20`**

Path	A “/” separated folder hierarchy in which to create the aliased point. Example: `Facility1/Voltage`
Description	A user-defined description of the point mapping.

Point Types

Type Name	Group	Supported Variations
SinglBitBinaryInput	1	1 - Packed format 2 - With flags
DoubleBitBinaryInput	3	1 - Packed format 2 - With flags
BinaryOutput	10	1 - Packed format 2 - With flags
Counter	20	1 - 32-bit with flags 2 - 16-bit with flags 5 - 32-bit 6 - 16-bit
FrozenCounter	21	1 - 32-bit with flags 2 - 16-bit with flags 5 - 32-bit with flags and time 6 - 16-bit with flags and time 9 - 32-bit 10 - 16-bit
AnalogInput	30	1 - 32-bit with flags 2 - 16-bit with flags 3 - 32-bit 4 - 16-bit 5 - Float with flags 6 - Double with flags
FrozenAnalogInput	31	1 - 32-bit with flags 2 - 16-bit with flags 3 - 32-bit with time of freeze 4 - 16-bit with time of freeze 5 - 32-bit 6 - 16-bit 7 - Float with flags 8 - Double with flags
AnalogOutput	40	1 - 32-bit with flags 2 - 16-bit with flags 3 - Float with flags 4 - Double with flags
OctetString	110	0 - 255

When the driver (master) connects to a device (outstation), an integrity poll is performed. Any DNP3 objects returned in the response that fall under the above point type categories are mapped to the OPC server with the appropriate index. For example, g40v1i2 corresponds to an analog output point, variation 1, index 2.

Internal Indicators

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

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

Terminology

unsolicited response: An Application Layer message from an outstation to a master for which no explicit request was received. The request is implied by the act of a master enabling unsolicited reporting of various points within an outstation.

integrity poll: Requests all event data, followed by the static data of all points assigned to one of the four classes (static Class 0 or event Class 1, 2, or 3).

DNP3TIME: Univeral Coordinated Time (UTC) time expressed as the number of milliseconds since the start of January 1, 1970. The effective date for using the UTC time base is January 1, 2008. Prior to this, DNP3 did not require a specific time reference.

Omron Driver

The Omron Driver module allows the Ignition OPC-UA server to communicate with NJ-series controllers.

To connect Ignition to an Omron device

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

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

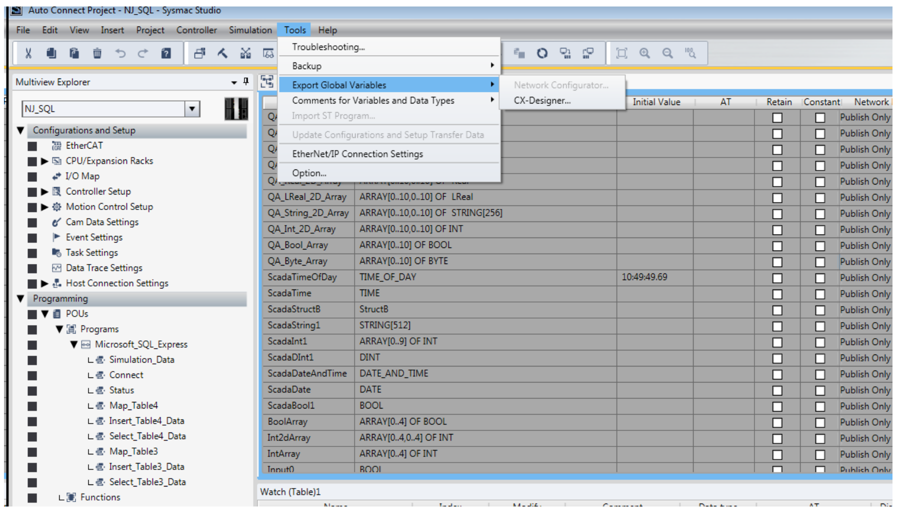
6. Click **Create New Device**.
   The **Devices** page is displayed showing the **Omron** device is successfully created and added to Ignition.
7. On the **Devices** page, click the tags link next to the newly created device.
   The **Manage Tags** page is displayed, allowing you to configure which variables in the device will show up as tags in Ignition.

Gateway Settings

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

Exporting from the Device

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

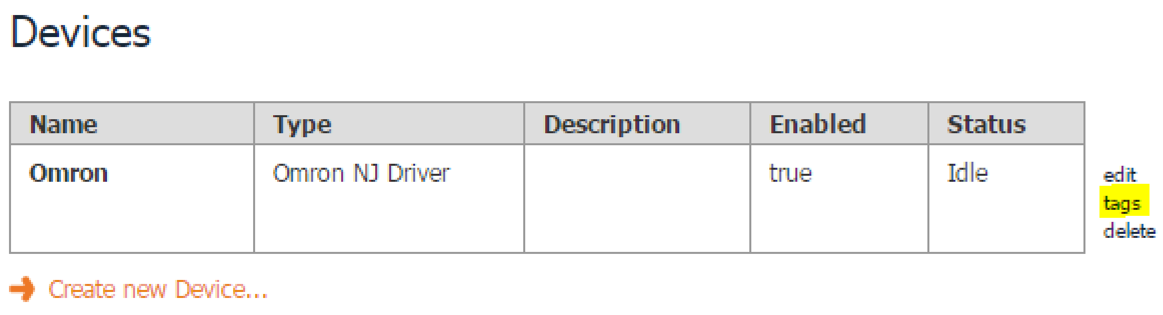


		Initial Value	AT	Retain	Constant	Network
QA_LReal_2D_Array	ARRAY[0..10,0..10] OF LReal			☐	☐	Publish Only
QA_String_2D_Array	ARRAY[0..10,0..10] OF STRING[256]			☐	☐	Publish Only
QA_Int_2D_Array	ARRAY[0..10,0..10] OF INT			☐	☐	Publish Only
QA_Bool_Array	ARRAY[0..10] OF BOOL			☐	☐	Publish Only
QA_Byte_Array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

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

Managing Tags

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

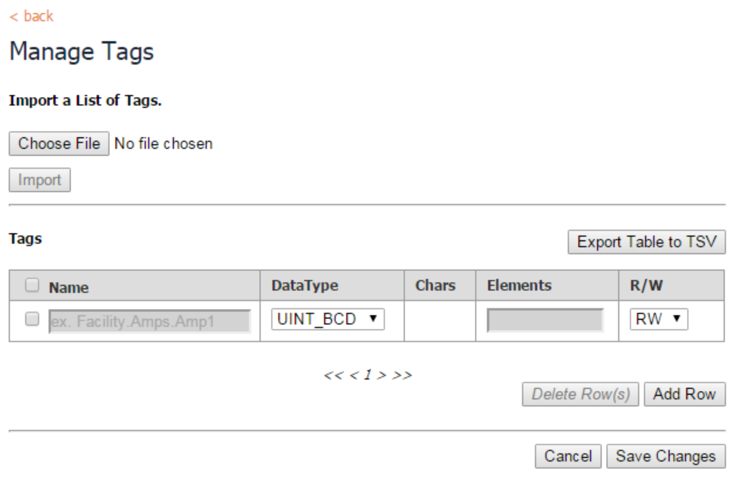


Name	Type	Description	Enabled	Status
Omron	Omron NJ Driver		true	Idle

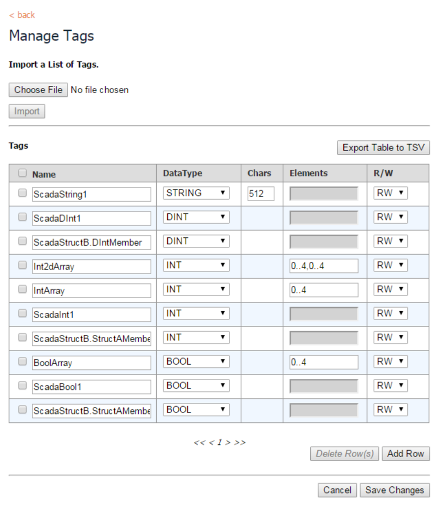
Importing Tags

Once on the *Manage Tags* page, you can manually enter the tags, or import them from a tab-separated file.

When importing, first choose a file, then click the *Import* button. The default option when importing is to replace the *Tags* table with tags from the import. Select the append option to append tags to the table.



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



< back

Manage Tags

Import a List of Tags.

Choose File No file chosen

Import

Tags

Export Table to TSV

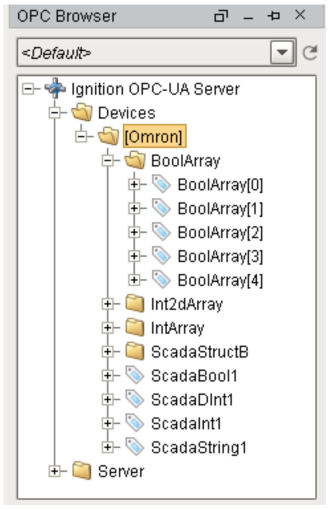
	Name	DataType	Chars	Elements	R/W
☐	ScadaString1	STRING	512		RW
☐	ScadaDInt1	DINT			RW
☐	ScadaStructB.DIntMember	DINT			RW
☐	Int2dArray	INT		0..4,0..4	RW
☐	IntArray	INT		0..4	RW
☐	ScadaInt1	INT			RW
☐	ScadaStructB.StructAMembe	INT			RW
☐	BoolArray	BOOL		0..4	RW
☐	ScadaBool1	BOOL			RW
☐	ScadaStructB.StructAMembe	BOOL			RW

<< < 1 > >>

Delete Row(s) Add Row

Cancel Save Changes

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



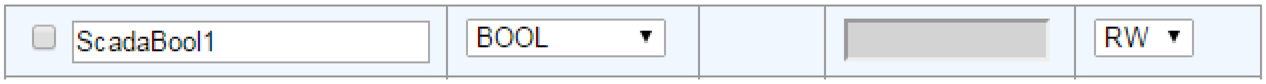
Addressing

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

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

Scalars

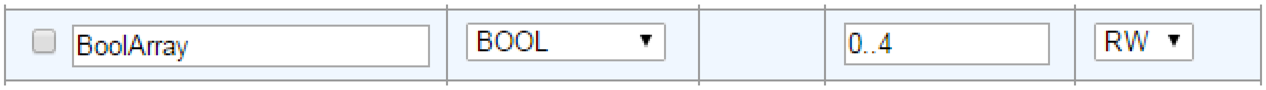
Leaving the *Elements* column blank will result in a scalar tag. When reading from the device, only one element will be requested.



☐	ScadaBool1	BOOL			RW

Arrays

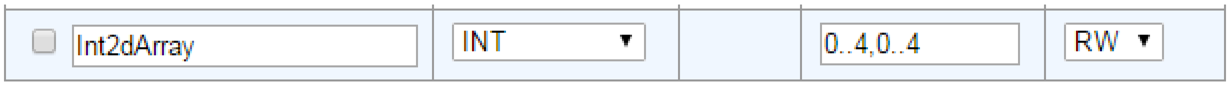
Specify the number of elements in an array in the form of **0..N** . The initial index 0 is always included, so an array mapped with 0..4 elements is a 5 element array.



☐	BoolArray	BOOL		0..4	RW

Multi-Dimensional Arrays

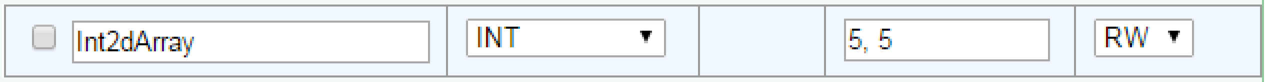
Multi-dimensional arrays are specified in the same way as arrays, with each group of indices separated with a comma.



☐	Int2dArray	INT ▼		0..4,0..4	RW ▼

Optional Format

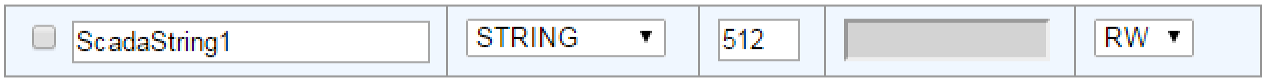
Array elements may also be specified with a single number equaling the total number of elements.



☐	Int2dArray	INT ▼		5, 5	RW ▼

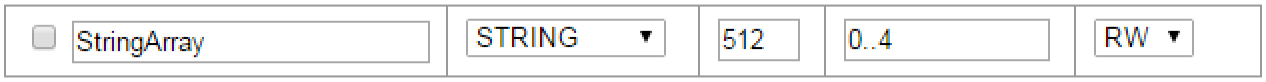
Strings

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



☐	ScadaString1	STRING ▼	512		RW ▼

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



☐	StringArray	STRING ▼	512	0..4	RW ▼

Mobile Module

The Mobile module adds the ability to launch Vision module projects on modern smartphones. This lets you keep track of your control system while moving around your facility. The Mobile module can be combined with remote-access networking architecture to allow global on-the-go access to your control system.

How it works

Normally, you can't launch Vision projects on mobile devices. This is because of the technical limitation of Java SE (Standard Edition) which does not run on mobile devices. The Mobile module gets around this limitation by launching the Client on the Gateway in a special headless (invisible) mode, and then using HTML5 and AJAX to show the Client's screen on the mobile device's browser.

Networking

Typically, the mobile device connects to the Ignition Gateway via the facility's wireless LAN (802.11) infrastructure. To launch a mobile client, the mobile device simply connects to the Ignition Gateway by pointing its web browser to the Gateway's LAN address. It is important to understand that normally, the traffic is not going over the device's cellular connection. This wouldn't work, because the cellular connection connects to the internet, and without explicit setup, an Ignition Gateway is not accessible from the outside internet.

Remote (as in, beyond the reach of 802.11 wireless LAN) mobile access can be enabled through the same networking strategies that enable remote access for standard Vision clients. Somehow, the mobile device must be able to access the Ignition Gateway via its cellular connection. One strategy would be to set up a VPN router and configure the mobile device as a VPN client. This way, the mobile device could directly access the LAN address of the Gateway as if it were on-site. Another technique would be to put the Ignition Gateway in a DMZ so that at least one NIC had a public IP address. Or, an edge router could be configured to port-forward the HTTP and HTTPS ports to the Gateway. Coordination with your IT is advised when attempting to set up remote access.

Alarm Notification Module

The Alarm Notification module adds the ability to send out email notification messages when alarms become active, cleared, or shelved. This module includes several features.

Product Data Sheet
To see the Product Data Sheet, click on Alarm Notification Modules.

Notification Pipelines

Alarm notification pipelines are logic diagrams that control how alarm notifications are sent out, including who they are sent to and when. They can be used to achieve many advanced alarming scenarios, such as delays, escalation, and parallel delivery. They are designed using a drag-and-drop designing interface through the Ignition Designer.

On-call Roster

An on-call roster (or simply 'roster') is a group of users that will receive the alarm notification. The Alarm Notification Module allows you to configure multiple rosters so only the users you want to receive the notifications will be notified. Rosters are created on the Gateway webpage, but can be modified from a Client with the Roster Management component.

User Schedule Integration

The Alarm Notification Module takes User Schedules into account. Users on a roster that are considered active will be notified, while inactive users will be skipped. This way you do not have to change your roster every time a user starts or ends their shift.

Two-Way Acknowledgement

The Alarm Notification Module includes the ability to send alarm notifications via email. Users who receive these emails are also able to acknowledge the alarm by visiting a link contained in the email.

SMS Notification Module

The SMS Notification module adds the ability to send alarm notifications via SMS. The SMS messages are sent through a cellular modem, which must be purchased separately and configured with a SIM card belonging to an active cellular account. If SMS alarm notification is enabled, recipients of these messages can reply with a special code in order to acknowledge the alarm.

Note: The Alarm Notification module is required for the SMS Notification module to work.

Product Data Sheet
To see the Product Data Sheet, click on Alarm Notification Modules.

Alarm Notification Module Compatibility

The SMS Notification Module is fully compatiblewith the Alarm Notification Module, so all pre-existing on-call rosters and pipelines will work with SMS notification.

Two-Way Acknowledgement

The SMS Notification Module can be configured to allow users to acknowledge alarms by simply responding to the message. With the Two-Way property enabled, the notification is sent out with a response code in the message. The user simply replies with the response code and the alarm will be acknowledged.

Audit Log Integration

The SMS Notification module can be configured to log successful message delivery and user acknowledgement into any connected database. Once recorded, a simply query will return all notification activity.

SFC Module

The SFC module adds a powerful programming ability to Ignition called "Sequential Function Charts" or SFCs. A SFC is a series of elements that execute in sequential order. The most common of these elements is the Action Step, which can execute any number of python scripts. The chart can be configured to execute each Action Step only once, or loop indefinitely.

Easy Flow Control

The chart only proceeds through each element after all tasks in that element have finished. This means that SFCs are perfect for long running tasks that must complete in sequence.

Multiple instances

Multiple instances of a chart may be invoked simultaneously, and each instance of the chart runs independently of the other instances. Furthermore each chart may be invoke with different starting parameters that may be passed in at will.

Visualize Work Flow

The SFC module also includes a Vision component: the SFC Monitor. This component allows you to view the realtime status of all running charts, as well as the value of any parameters. This way you can easily determine the status of all running charts.

Parallel Tasks

Controlling the execution of multiple scripts that must run in parallel is normally very difficult to achieve, especially if any of those scripts take an inconsistent amount of time to complete. However the Parallel Section allows you to run multiple elements simultaneously, and only move on once all are finished. You'll never have to worry about checking the status of each element before moving on.

Enterprise Administration Module

Overview

The Enterprise Administration Module (hereafter abbreviated to EAM) allows a user to control and monitor any number of Ignition Gateways from a central Gateway. In EAM terminology, the central Gateway is called a controller, and the monitored Gateways are called agents. The EAM takes advantage of the Gateway Network system to seamlessly deliver data between Gateways.

Task Management

EAM allows you to schedule common Gateway tasks to execute against remote agent Gateways. Tasks include agent Gateway restart, backup collection, system restoration, module installation, license key activation/unactivation, and project distribution. Tasks can be scheduled to run once at a later time, or as part of a recurring schedule.

Agent Events

Every EAM agent sends information to the controller about its general health status. Whenever problems occur on an agent, the controller is notified of the agent's status in the form of agent events. Problems can include high CPU usage on the agent, large numbers of system errors in a short timeframe, and unusually high usage of database connections. Loss of connectivity to the agent also triggers an agent event on the controller. Agent events are recorded in an external database for analysis and reporting later. Agent events can also be directly configured in the controller to automatically issue alarms to a specified alarm pipeline.

Archival and System Recovery

The Collect Backup gateway task does more than simply collect a gateway backup from agents. Whenever the task runs, the gateway backup, all installed modules and the current license are collected from the agent and stored in the archive system on the controller. This system allows for quick recovery later if an agent machine catastrophically fails and must be reinstalled.

Tag Historian Module

The Tag Historian module allows users to quickly and easily setup history logging on any tag. At its core, the Tag Historian gives a user the ability to select which tags to store history on, then drag those tags onto a chart or table for viewing. The Tag Historian will automatically partition its data into multiple tables, allowing for faster loading of large time spans of data. The Tag Historian module includes the functionality of the Tag History Splitter module, allowing you to store history to two separate database connections at the same time.

Tag History

Tag History is configured in the designer on each tag. By opening the tag editor for a tag and navigating to the History section, a user can quickly setup tag history by selecting a History Provider, which are automatically created for each database connection that is set up, and opting to store history on that tag. There are other settings there such as the Historical Scanclass that determines how often the historian will scan the tag, or the Historical Deadband, that is used to determine when a value has actually changed enough to store a historical value.

The Tag Historian uses the scanclass configured on each tags history setting to determine how quickly to scan the tag. When the tag does get scanned, the historian uses the Historical Deadband, Historical Deadband Mode, and the Value mode to determine if the value has changed enough to store a value to the database. If the value has changed enough, the tag historian will automatically create any necessary tables that have not been created, and store the history appropriately.

Gateway Configuration

In addition to the configuration that can be done on an individual tag by tag basis, the historian allows you set some properties that pertain to that entire historical provider so that all data that goes through a certain tag provider and onto the database all follow a set of rules. By default, the history providers start with partitioning turned on, but this can be turned off, or modified to change how big a partition should be in regards to time. The history providers also allow data pruning to be setup, so that all data that is older than a specified time will be deleted automatically. This makes maintaining space on the database a little easier, knowing that the historian will always keep itself in check with how large it grows.

Tag History Splitter

The Tag Historian now comes with the Tag History Splitter included in the module. The Tag History Splitter works in the same way that a regular historical provider linked to a database works, in that you get to change some settings like data pruning and table partitioning. The difference is that when setting up a tag history splitter tag provider, the splitter provider allows you to select two other history providers to forward data to instead of a single database connection. So, when setting up history on a tag, the newly created history splitter provider can be selected for that tags historical provider allowing that tags history to automatically be stored in two databases at the same time.

The Tag History Splitter is unique in that it allows you to select any two other historical tag providers, even other splitter providers. You could have a splitter provider looking at two other splitter providers, allowing for data storage in four different database connections at once.

Additional Modules

When we speak of "Addition Modules", we are referring to those Modules that belong to the following two groups of Modules:

1. Modules that are included with the installer but are not automatically installed with the **Typical** installation option.
    - Voice Notification Module
    - OEE Downtime Module
    - SPC Module
    - Recipe/Changeover Module
    - Track and Trace Module
    - Web Service Module
    - Barcode Scanner Module
    - HASP
2. Modules that are not included with the installer but are available for download.
    - Voice Notification Module
    - OPC COM Module
    - OPC COM Tunneller Module
    - Web Browser Module
    - Barcode Scanner Module
    - WebDev Module
    - Tag History Splitter Module
    - Twilio Integration and Alarm Notification
    - Serial Modules

In this section ...

Voice Notification Module

The Voice Notification module adds the ability to deliver alarm notifications to users via telephone, using any SIP compatible phone system. Messages are constructed in text, and are delivered through a high quality text-to-speech engine. The engine supports multiple voices and languages . Therefore, in order for the voice notification module to work, a compatible voice module must also be installed.

Product Data Sheet
To see the Product Data Sheet, click on Alarm Notification Modules.

Alarm Notification Module Compatibility

The Voice Notification Module is fully compatible with the Alarm Notification Module, so all pre-existing on-call rosters and pipelines will work with Voice notification.

Two-Way Acknowledgement

The Voice Notification Module can be configured to allow users to quickly acknowledge alarms. With the Two-Way property enabled, the notification is sent out a response code in the message. The user simply replies with the response code and the alarm will be acknowledged.

Custom Notification Messages

The High-Quality Text-to-Speech Engine allows you to customize the alarm message the user hears, so you are not limited to a number of pre-recorded messages.

Audit Log Integration

The Voice Notification module can be configured to log successful message delivery and user acknowledgement into any connected database. Once recorded, a simply query will return all notification activity.

Multi-Language Support

The following is a list of the different languages in both female and male voices that are offered by the Voice Notification module:

- Australian Voice Module (Nicole)
- Dutch Voice Module (Ruben)
- English Voice Module (Salli)
- English Voice Module (Jennifer)
- English Voice Module (Kimberly)
- British English Voice Module (Amy)
- Spanish Voice Module (Miguel)
- Spanich Voice Module (Penelope)
- European Spanish Voice Module (Conchita)
- European Spanish Voice Module (Enrique)
- French Voice Module (Celine)
- French Voice Module (Mathieu)
- German Voice Module (Hans)
- German Voice Module (Marlene)
- Italian Voice Module (Carla)
- Italian Voice Module (Giorgio)
- Russian Voice Module (Tatyana)

OPC COM Module

The OPC-COM module offers Ignition the ability to connect to legacy ("classic") COM-based OPC-DA servers. It supports OPC-DA 2.0 and 3.0.

Tip

The OPC Com Tunneller Module simplifies connecting to remote OPC-COM servers.

Connecting to Classic OPC Servers

With the OPC-COM module installed, there will be a new option for COM-based OPC servers when creating a server connection in the Gateway. The OPC-COM module is primarily intended for use with local OPC servers, although it also provides basic support for remote connections.

Even when connecting locally, the application may run into the traditional difficulties of connecting to OPC servers. DCOM security settings on the machine can interfere with connections, and the OPC Core Components package must be properly installed before connections can be established.

Using Data from Classic OPC Servers

After a connection to a server has been defined, the server will appear along side of other OPC servers (both COM and UA based) in the OPC Tag Browser. You can use these tags like any other ones - bring them into Tags, use them in Transaction Groups, and so on.

OPC COM Tunneller Module

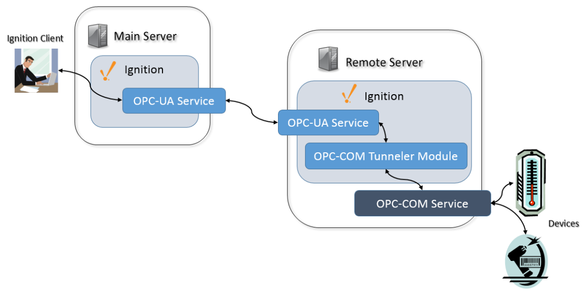
The OPC-COM Tunneler module eliminates many difficulties associated with establishing remote OPC-COM connections. The OPC-COM Tunneler module exposes OPC-COM connections to Ignition's OPC-UA server. The connections appear and behave as devices connected to Ignition's OPC-UA server. In this way, Ignition manages the OPC-COM connection in the context of the OPC-UA standard resulting easier access over the network, eliminating the need to establish remote DCOM connections, and incorporating the security benefits of the OPC-UA standard such as authentication and encryption. This module can be considered a "UA wrapper" for classic OPC servers.

The following picture illustrates how this module can fit into your architecture.

OPC Com Tunneller

Learn how to configure the OPC Com Tunneller module.

Link to Knowledge Base Article



Web Browser Module

The Web Browser module adds a web-browser component to the Vision module. This component allows you to embed other webpages from the internet or from your local intranet inside of your projects. The browser is based on the open source project Chromium, and supports modern web technology such as HTML5, CSS, and Javascript.

Web Browser Module

About

This module allows users have a web browser component in the Vision module.

System Requirements

Client machines need to meet the following minimum requirements to use this component. The component may not work properly if the requirements are not met.

Windows

- Microsoft Windows XP (SP2), 7, 8, Vista, Server 2003 (SP1), Server 2008/2012, 32-bit and 64-bit.
    - Windows version 8 and 8.1 require Java 6 update 38 or greater
- Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Linux

- Ubuntu 12.04+, Debian 7.7, RedHat Enterprise Linux 7, openSUSE 13.1, Fedora 20, 32-bit and 64-bit
- Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Mac OS X

- Mac OS X 10.7.x - 10.10.x (Intel)
- Apple or Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Barcode Scanner Module

This Barcode Scanner module presents a simple webpage meant for handheld barcode scanners that have an integrated web browser, for example, the Intermec CK71. When items are scanned, they queue up to be sent back to the Ignition Gateway, where a python script is called to handle the incoming scan. The scanner page is designed to be tolerant of wireless dead zones, queueing up scans until connectivity is restored.

Unique Identifier for each Scanner

Each scanner that navigates to the webpage can be given a unique identifier which ties to a specific scanner. This way multiple scanners can be out in the field at once, and the scanned data can be easily tracked to a certain identifier

Custom Actions

Received scans trigger a python script. From this script you can achieve any number of tasks, such as writing to a tag, or inserting a new row into a database table.

WebDev Module

This module allows users to directly program against the web server inside the Ignition Gateway. Webpages can be build by hand using a combination of python programming and static web resources such as images, css files, javascript files, and html files. Likewise, this module allows you to build RESTful web service APIs that allow external systems to interact with the Ignition server. This module follows the normal installation process.

> **Disclaimer**
>
> The Web Dev module requires specialized web-programming knowledge. The Inductive Automation support team is unable to provide detailed advice about creating a particular site. Furthermore, they are unable to provide troubleshooting beyond the basic functionality of the module.

Basic Usage

Each type of resource may specify its *content type*. It is important to specify the correct content type for the contents of the resource. When the WebDev module is installed, a new kind of project resource heading will apear in the Designer's project browser called "Web Dev". Right-clicking on this heading will allow the creation of three new types of project resources:

Python Resource

Python resources are dynamic web resources. Each time a user browses to the URL associated with a python resource, the script will run and generate a response.

File Resource

A file resource is a static resource, usually a binary resources such as an image.

Text Resource

A text resource is a static resource much like a file resource, except that its contents may be directly edited from within the Ignition designer. These are useful for static HTML, CSS, and JavaScript files.

URL

Each resource will be directly accessible over HTTP and mounted beneath the/main/system/webdev path.

For example, if you created a Text Resource directly beneath the "Web Dev", it would be mounted at:

```
http://host:port/main/system/webdev/project/resource_name
```

Notice that the project name and resource name are part of the path. If your resource is nested inside a folder, it will be part of the path too, for example:

```
http://host:port/main/system/webdev/project/folder_name/resource_name
```

Web dev resources may have periods in their name. This meas that if you upload an image file, you may include its extension directly in its name so that its path is more natural, for example, you might name an image resource "my_image.png" so that its URL is:

```
http://host:port/main/system/webdev/project/my_image.png
```

The Web Dev module respects the published vs. staging system of Ignition. If your project is set up to have separate staging and published verions, you may access your staging version resources via the URL:

```
http://host:port/main/system/webdev_staging/project/resource_name
```

Requests to the root of your project, i.e. will attempt to load a resource named "index.html". If no such resource exists, a 404 response code will be returned instead.

```
http://host:port/main/system/project
```

Python Resources

Python resources are the heart of the functionality of the Web Dev module. These resources are completely dynamic, and can handle all parts of the HTTP protocol, formulating any type of response.

Each time an incoming HTTP request arrives an a URL where a python resource is mounted, the corresponding python script will be run. Each python resource may have up to seven scripts, one for each HTTP method (GET, POST, HEAD, TRACE, etc.). In practice, most python resources will probably only implement one or two of these, usually doGet or doPost at a minimum.

Parameters

Each do* method receives the same two parameters: 'request' and 'session'.

Request Parameter

The request parameter is a dictionary with a bunch of information about the incoming web request.

>request['params']

Any URL parameters such as the following:

```
/main/system/webdev/project/foo?param1=value&param2=other_value
```

This will be contained in a dictionary accessible via request['params']. For the given example, request['params'] = {'param1':'value', 'param2':'other_value'}

request['postData']

This parameter is only present for the doPost method, and its value is different based upon the value of the incoming HTTP request's Content-Type header. If the content type is application/json, then the request['postData'] will be a python dictionary structure which is the result of parsing the json document that was posted. If the content type starts with text/, then the value of request['postData']will be the text which was posted, as a string.

request['headers']

The HTTP request headers will be in a dictionary under request['headers']. For example, you could read the User-Agent string with request['headers']['User-Agent'].

request['scheme']

The scheme is available via request['scheme']. The value will be 'http' or 'https'.

request['remoteAddr'] / request['remoteHost']

These two parameters give the remote IP address and host of the entity that made the web request. Note that these are from the perspective of the web server, and so may not be what you expect due to the effects of things like NAT-ing routers.

request['servletRequest'] / request['servletResponse']

These two parameters give you direct access to the underlying Java servlet request and response objects.request['context']. This provides direct access to the Ignition GatewayContext. More information about this object may be found in the Ignition SDK developer guide and associated JavaDocs.

request['context']

This provides direct access to the Ignition GatewayContext. More information about this object may be found in the Ignition SDK developer guide and associated JavaDocs.

Session Parameter

The session parameter is a Python dictionary which may be used to hold information which persists across multiple different calls to your Web Dev Python Resource, or across multiple Python Resources. Session management is handled automatically, and this dictionary is created for each new client session (the client must have cookies enabled for sessions to work). You may place any key-value pairs in the session dictionary you'd like, just make sure that the values are things that can be serialized. All normal python types (scalar, dictionary, lists, and tuples) are serializable.

Return Value

Each do* function on a python resource must generate some sort of HTTP response. This is done by returning a value from the function. The return value should always be a dictionary, in which the following keys are recognized. The keys are listed here in the order they are evaluated. For example, if you have both file and bytes, only file will take effect. The exception is the contentType key, which may be included with any of the other keys to override the default content type.

File: The value should be a string, which should be the path to a file on the server's filesystem. If no contentType is specified, then the system will attempt to probe the content type from the operating system using java.nio.Files.probeContentType. If thefile key is present, but the value points to a file that doesn't exist, an HTTP 404 will be returned.

Bytes: The value should be a byte array. The default content type is application/octet-stream, but you probably want to specify your own.

HTML: The value should be a string, which should be the source of an HTML document. Content type is assumed to be text/html.

JSON: The value is assumed to either be a string (which should be valid json) or to be a python object, which will then be encoded into a json string. Content type will beapplication/json

Response: If none of the other keys are present, the system will look for the key response which will be stringified and then returned with the content type text/plain.

If your implementation of the do* function returns a dictionary with none of the above keys, an HTTP 500 error will be returned. However, if you return None, no HTTP 500 error will be returned. In this case, it is assumed that you used therequest['servletResponse'] parameter to directly formulate your HTTP response.

Security Settings

Each python resource can configure its own optional security settings.

Require HTTPS

If this is checked, then the resource will only be accessible via an SSL connection. If a non-secure HTTP transport is used, the browser will be sent a redirect to the the gateway's SSL port. The gateway must have SSL enabled, of course.

Require Authentication

If this is checked, the resource will require authentication before it executes. This uses HTTP BASIC auth, and so should really be combined with the Require HTTPSoption so that the credentials are encrypted. The username/password combination sent through the HTTP BASIC authentication headers will then be passed through the chosen User Source. If roles are specified, the user must have at least one of the roles. Specify multiple acceptible roles using a comma separated list. If the credentials are missing, an HTTP 401 will be returned with the WWW-Authenticateheader. If the credentials are present but incorrect, an HTTP 403 will be returned.

If the credentials succeed, the python resource will execute. In addition, the authenticated user object returned by the User Source will be accessible inside the session object as session['user']. Since the user is stored in session, if the client has cookies enabled, then further requests against the same session will use the stored user object and will not require additional authentication.

Tag History Splitter Module

The Tag History Splitter module adds a new type of tag history provider that simply points to two other providers at the same time. Data stored to the new provider is "split" to both, storing the same data in each provider.

Record Tag History to Multiple Databases

The Tag History Splitter Module allows data recorded by the Tag Historian module to be inserted into multiple databases simultaneously by way of a new "Splitter" Historical Tag Provider. Normally this type of architecture would require scheduled bulk-inserts from one database to another at regular intervals, or duplicate tags.

Nested History Providers

It is possible for Splitter providers to record data against other Splitter providers. This allows the history to be recorded in more than two separate database connections.

Twilio Integration and Alarm Notification

The Twilio Integration and Alarm Notification module adds integration with Twilio via an SMS Alarm Notification Profile and scripting functions to send SMS.

Send SMS messages through Scripts

This module unlocks the system.twilio functions. These allow you to send SMS message via python script, and without the use of Alarming.

Serial Modules

Serial support

The Serial Modules module add serial comm support and system.serial functions to python scripts. These functions allow you to open, read from, write to, and close serial ports. More information on these function can be found in the system.serial section of the appendix.

Multiple Scopes

There are two variants of the Serial Module. Both perform similar functionality, but at different scopes:

- The Serial Module Client allows the system.serial functions to be accessed from client based scripts, as well as connect to serial devices plugged into the computer running the client
- The Serial Module Gateway allows the system.serial functions to be accessed from gateway based scripts, as well as connect to serial devices plugged into the gateway server

HASP

HASP module is a hardware activation key.

OPC-UA Drivers

OPC-UA Driver modules are different than the regular modules because they require the OPC-UA module, and they are focus on establishing communications to specific devices.

OPC-UA Drivers:

TutorialsRoot

Tutorials & Helpful Tricks

In this section ...

Best Practices when Upgrading

Steps to Upgrade

The Ignition installer also doubles as an upgrader. Simply download the executable installer from our website for your operating system. See the the upgrade pages for instructions specific to Windows, Linux, and Mac OS X operating systems.

Reminder

The bit architecture of the Java and Ignition have to match; servers with only a 64-bit version of Java should use a 64-bit Ignition installer.

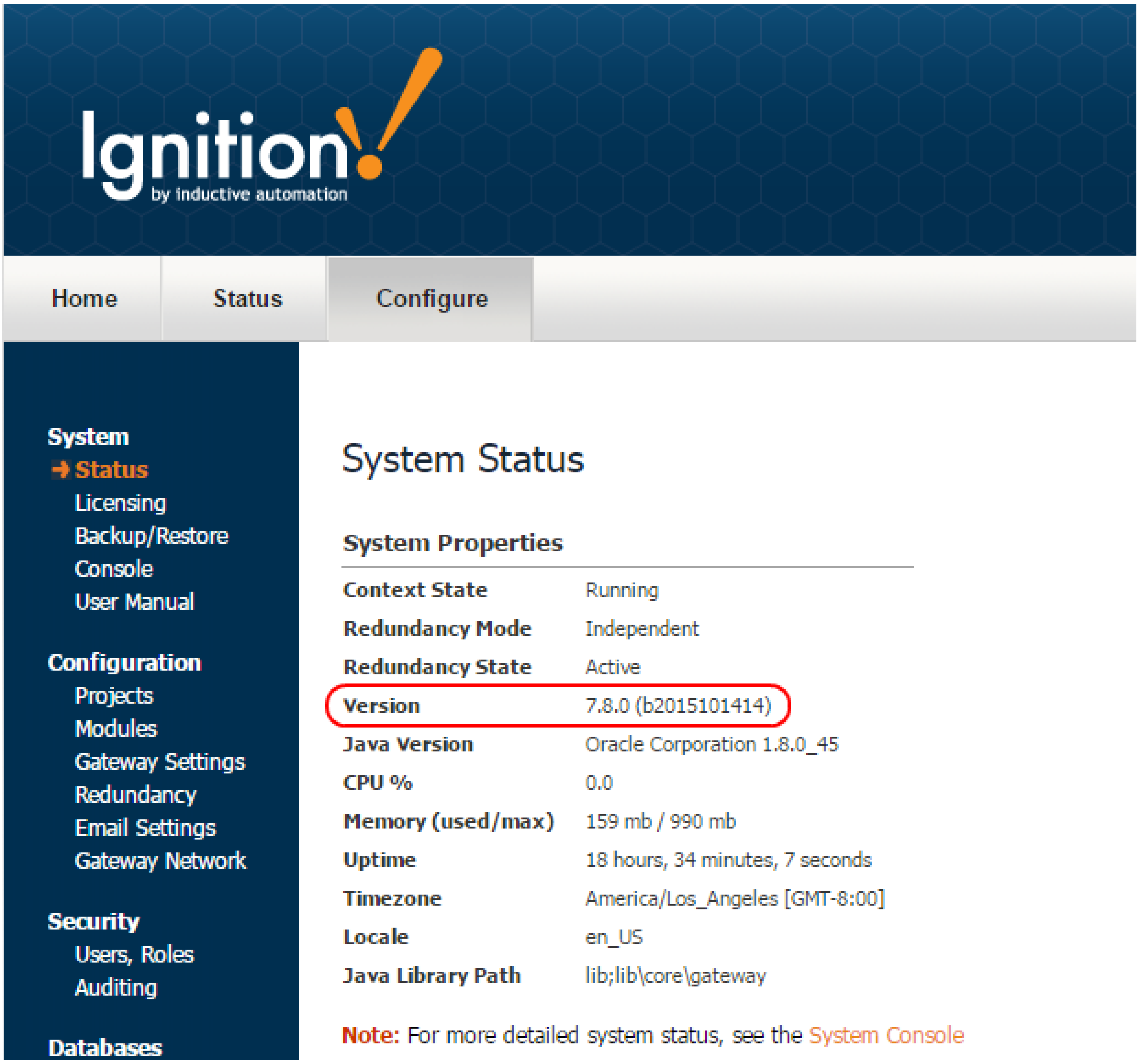
Regardless of which Operating System you are using, you should follow these steps prior to running the Ignition installer:

1. Check your versions
2. Update your License (if needed)
3. Make a Gateway Backup (most important!)
4. Update on a Test Server
5. Upgrade the Production Server

For more information about the changes in each version, see the New in this Version page.

Check Versions

Is this is a Major version update or a Minor version update? You can check the version of your Ignition in several ways. The easiest is to log into the **C onfigure** section of the **Gateway Webpage**. The version is listed in the main Status page.



System Status

System Properties

Context State	Running
Redundancy Mode	Independent
Redundancy State	Active
Version	7.8.0 (b2015101414)
Java Version	Oracle Corporation 1.8.0_45
CPU %	0.0
Memory (used/max)	159 mb / 990 mb
Uptime	18 hours, 34 minutes, 7 seconds
Timezone	America/Los_Angeles [GMT-8:00]
Locale	en_US
Java Library Path	lib;lib\core\gateway

Note: For more detailed system status, see the System Console

It is a **MINOR VERSION** upgrade if only the last number changes. IE: from **7.8.1** to **7.8.3**.

Minor version changes are mostly bug fixes. These are minor changes to Ignition and will fall under your existing license.

You **do not** have to update your license.

It is a **MAJOR VERSION** upgrade if one of the first two version numbers changes. IE: from **7.6** to **7.7**, or from **7.8** to **8.1**.

Major version changes are very large changes to Ignition. These will add functionality or change the way some of the systems in Ignition work.

You **do** have to update your license.

Update your License

1) Let us know.

You can reach Inductive Automation by phone at **(800) 266-7798**.

Some Major upgrades include new modules or have modified the functionality available in each module. If your license needs to be updated, call us and we can guide you through it. If you are updating from one version to the next, or from 7.0.0 all the way to the latest version, we can talk you though what is different and what new functionality is available. See the New in this Version page for more specifics about what was updated in the various versions.

There are several ways to get discounted or free Major upgrades, ask your rep for more information.

2) Reload your license.

Once your license is updated in our system, go to your Gateway webpage and click the **Update License** link there.

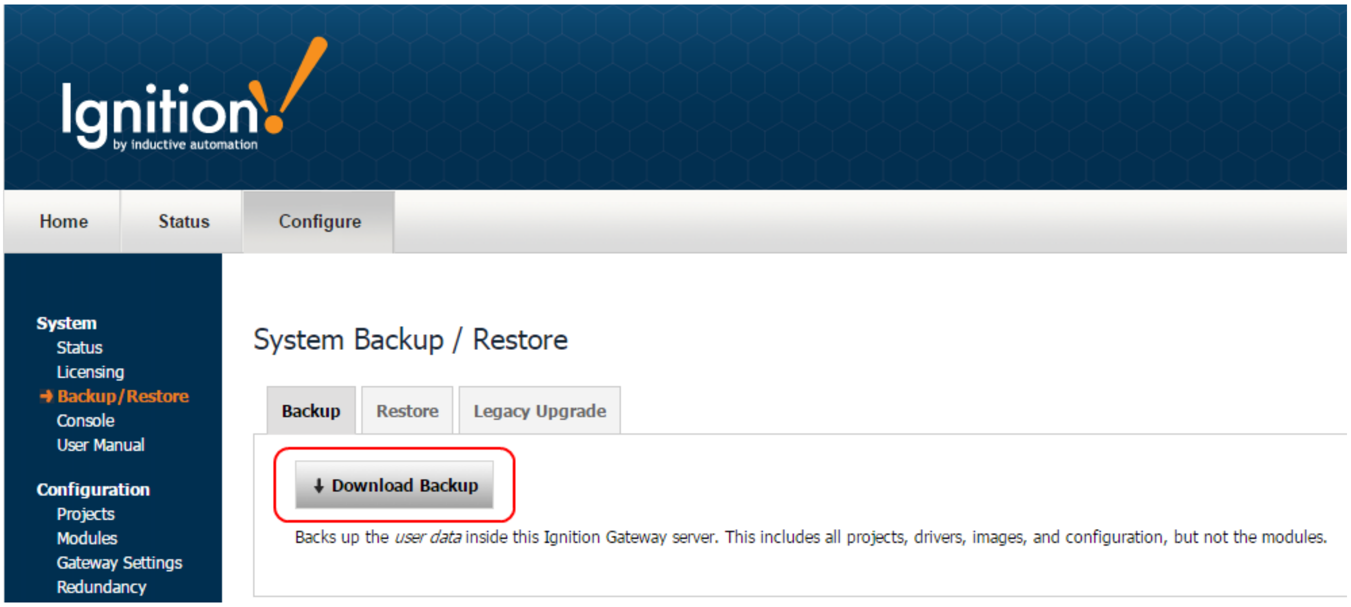
See this page for a detailed guide.

Make a Gateway Backup

This is the most important step!

Upgrading Ignition is very easy, but you always want to make a backup before any major changes.

Go to the **Configure** section of the **Gateway Webpage** and make a backup before changing anything.



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

Update on a Test Server

It is always a good idea to test your changes before running them in a production environment!

One of the great things about Ignition is that you can run any Gateway in Trial mode (without a license). That means all you need to do is install on another computer and load your backup.

You can download any version of Ignition from the Inductive Automation downloads archive page.

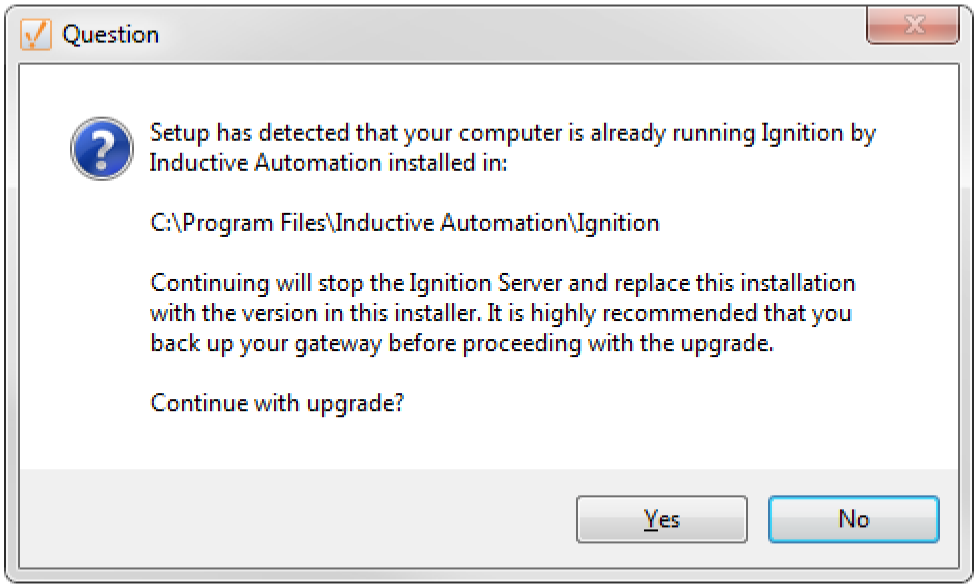
To load the project on another computer, follow these steps:

1. Install your original version of Ignition.
2. Restore the Gateway Backup you just made.
3. Run the new installer to upgrade your test system (see next step).
4. Check you projects for errors.

Upgrade the Production Server

Once you've made a backup, just run the Ignition installer!

The Ignition installer doubles as an upgrader and will automatically check for an existing installation and upgrade any installed modules.



Setup has detected that your computer is already running Ignition by Inductive Automation installed in:

C:\Program Files\Inductive Automation\Ignition

Continuing will stop the Ignition Server and replace this installation with the version in this installer. It is highly recommended that you back up your gateway before proceeding with the upgrade.

Continue with upgrade?

Mapping a Network Drive

Windows makes it possible to map drives on network servers to local drive letters so they can be accessed by users as if they were a local drive. The problem, however, is that Windows is not very consistent about how it handles such mapped drives when accessed from a Windows Service, such as the Ignition Gateway. When the service is started manually, the drives will be available, but when the system is rebooted, the service will no longer be able to access them. There may be users that wish to read or write data in Ignition using shared drives, and don't want to manually set up shared drives each time.

⚠ Must be using Ignition at least 7.5 and Java Service Wrapper 3.5.4

To make shared drives available to Ignition on startup, place the following lines in the ignition.conf file, which is located in the data folder of the main Ignition installation folder (usually C:\Program Files\Inductive Automation\Ignition\data for Windows users):

```
//Note that target "Z:" is the drive letter assigned to the mapped drive.

wrapper.share.1.location=\\fileserver\folder
wrapper.share.1.target=Z:
wrapper.share.1.type=DISK
```

Change the appropriate data for location and target to match the computer's actual setup. If your shared drives require authentication, add the following lines, filling in the appropriate data for user, domain, and password:

```
wrapper.ntservice.account=user
wrapper.ntservice.password=password
wrapper.share.1.account=domain\user
wrapper.share.1.password=password
```

To turn on debugging to see what is causing network share connection issues, make sure to enable wrapper.debug = TRUE by removing the pound (#) sign in front of that line of code. The location of the log file is in the main Ignition installation folder as wrapper.log.

Other Notes:

Un-map drive on shutdown

```
wrapper.share.1.shutdown.unmap=TRUE
```

How often to retry server connection:

```
wrapper.share.1.startup.max_retries=2
```

What interval (seconds) between retries:

```
wrapper.share.1.startup.retry_interval=10
```

Set wrapper to fail to startup if network share not found:

```
wrapper.share.1.startup.failure=SHUTDOWN
```

Troubleshooting:

Here a couple of the common problems that are encountered when mapping network shares:

- **Server not found**
  The debug output will show something like this if a drive can't be reached

  ```
  wrapper | Attempting to map the "\\fileserver\folder" share to "S:"...
  wrapper | Unable to map "S:". Attempt #1 (The network name cannot be found. (0x43))
  wrapper | Attempting to map the "\\fileserver\folder" share to "S:"...
  wrapper | Unable to map "S:". Trying to continue. (The network name cannot be found. (0x43))
  ```

- **Incorrect Login Data**
  If the configured account or password are incorrect (or are missing) then the mapping will fail with a message like the following:

  ```
  wrapper | Attempting to map the "\\myfileserver\commonshare" share to "S:"...
  wrapper | Unable to map "S:". Trying to continue. (Access is denied. (0x5))
  ```