

Ignition 7.9 User Manual — Tag Historian

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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

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

Historian

Recording History

Ignition has two main approaches to recording historical data:

- **Tag Historian:** Individual or groups of tags can be configured to record history based on scan class execution.
- **Transaction groups:** Groups of OPC items that are recorded on an execution cycle. More information on Transaction Groups can be found in the [SQL Bridge \(Transaction Groups\)](#) section.

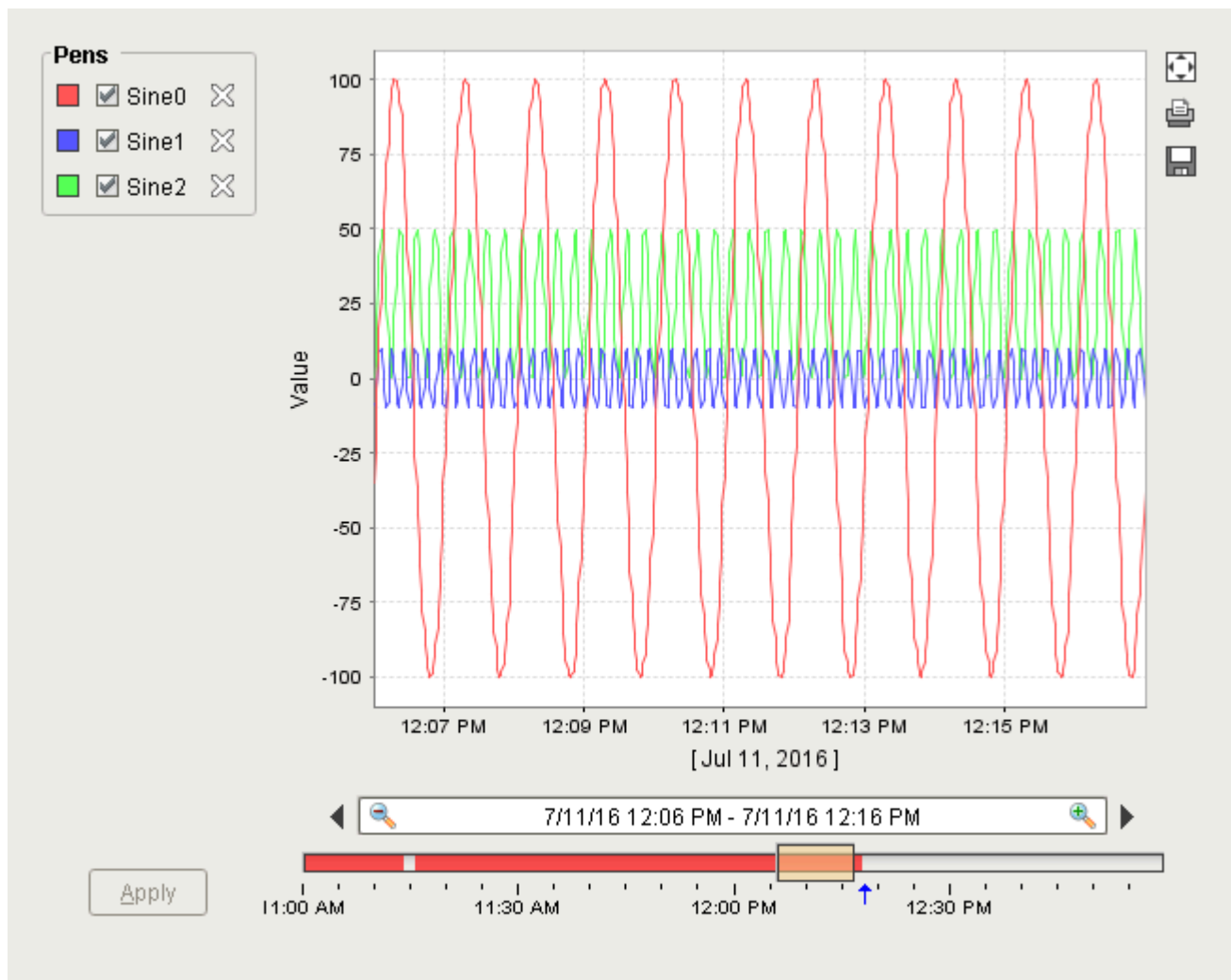
What is the Tag Historian?

Tag Historian makes it easier than 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 component to create trends or onto a table to display historical values. Tag Historian's robust querying features provide you great flexibility in how you retrieve the stored data.

Historian Charts

Creating charts to display historical data or realtime data is a snap! The [Easy Chart component](#) allows you to make powerful, runtime-configurable time series charts from the Tag Historian data. Simply drag and drop history-enabled tags onto a chart to create chart pens and data trends. Your charts and graphs can include subplots, axes, digital offsets, and a moving average pen. You can quickly and easily turn your historical and realtime data into dynamic charts and graphs for your users.

To see all the chart options and features, refer to the section on [Using the Easy Chart](#).



Historian Tables

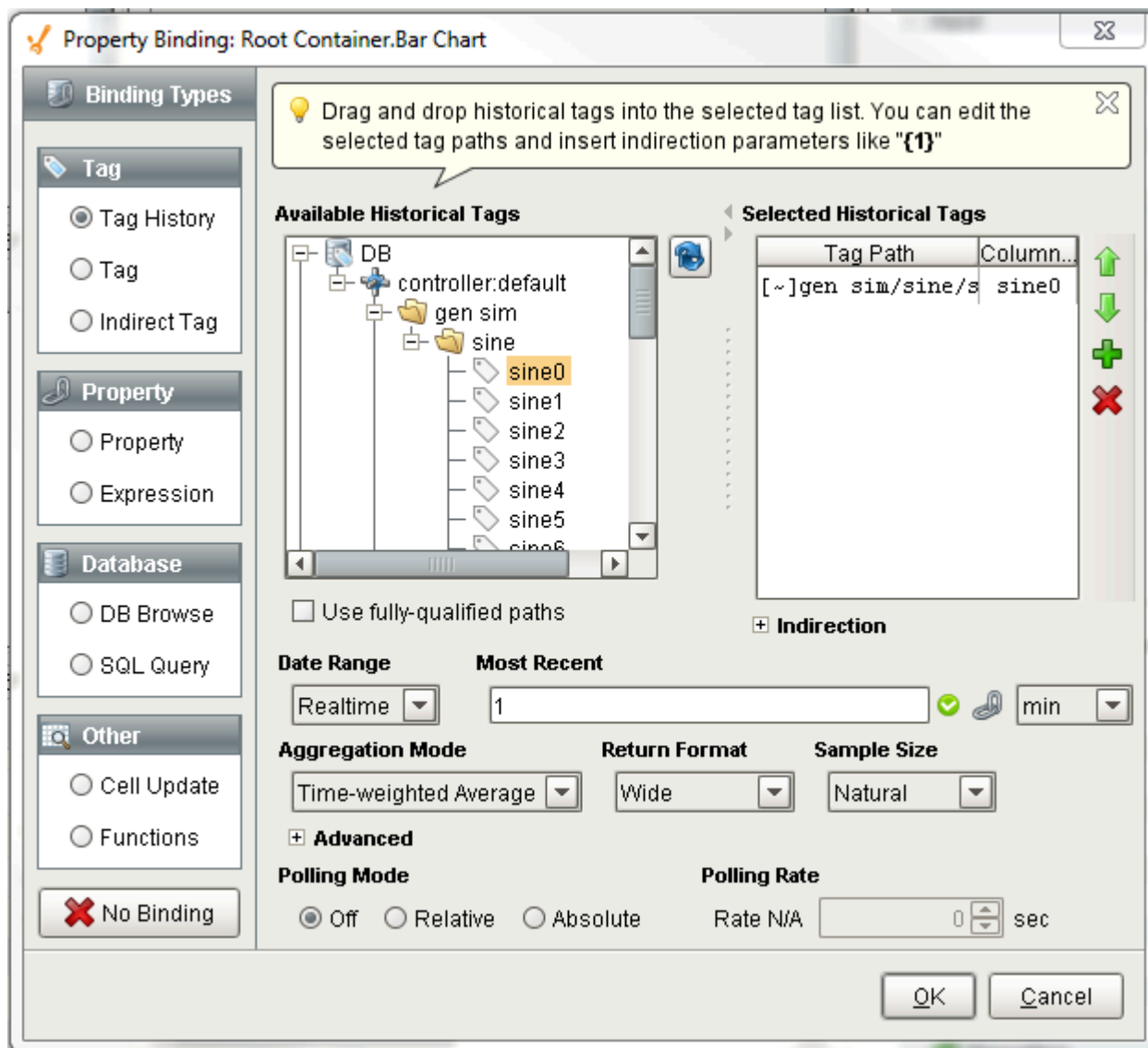
Just like charts, creating tables using the [Table component](#) is just as easy! Drag and drop historian enabled tags onto a table component to display historical values in many different ways depending on how you customize your table. You can bring back the most recent history, a specific date range, fixed sample size, interval sample size, and more. To see all the table options and features, refer to the section on [Tag Historian and Tables](#).

t_stamp	Tag1	Tag2
Jul 11, 2016 11:57 AM	10	25
Jul 11, 2016 11:58 AM	10	25
Jul 11, 2016 11:58 AM	10	25
Jul 11, 2016 11:58 AM	10	25
Jul 11, 2016 11:58 AM	10	25
Jul 11, 2016 11:58 AM	10	25
Jul 11, 2016 11:58 AM	10	25
Jul 11, 2016 11:59 AM	10	25
Jul 11, 2016 11:59 AM	10	25
Jul 11, 2016 11:59 AM	10	25
Jul 11, 2016 11:59 AM	10	25
Jul 11, 2016 11:59 AM	10	25
Jul 11, 2016 11:59 AM	10	25
Jul 11, 2016 11:59 AM	10	25
Jul 11, 2016 12:00 PM	10	25
Jul 11, 2016 12:00 PM	10	25
Jul 11, 2016 12:00 PM	10	25

Tag Historian 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 built-in 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. You can also query tags from the [Reporting Module](#).



Other Methods of Storing Historical Data

SQL Bridge (Transaction Groups) perform a variety of tasks, one in particular, to store data historically. In their simplest form, transaction groups read values from the OPC addresses, and store them into a SQL database. There are four types of Transaction Groups; Standard, Block, Historical, and Stored Procedures. Each transaction group handles data a little differently.

- **Standard Group** - is the most flexible, and can also write database values to OPC addresses or synchronize data changes between both the database and PLC. This allows you to create true realtime values tables in a database, and push values to a PLC.
- **Historical Group** - can quickly store data from the plant floor into any kind of SQL database.
- **Block Group** - transfers large amounts of data very efficiently.
- **Stored Procedures** - uses PLC data as inputs and outputs from your existing stored procedures.

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.

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

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

Configuring Tag Historian

Set Up Tags to Log

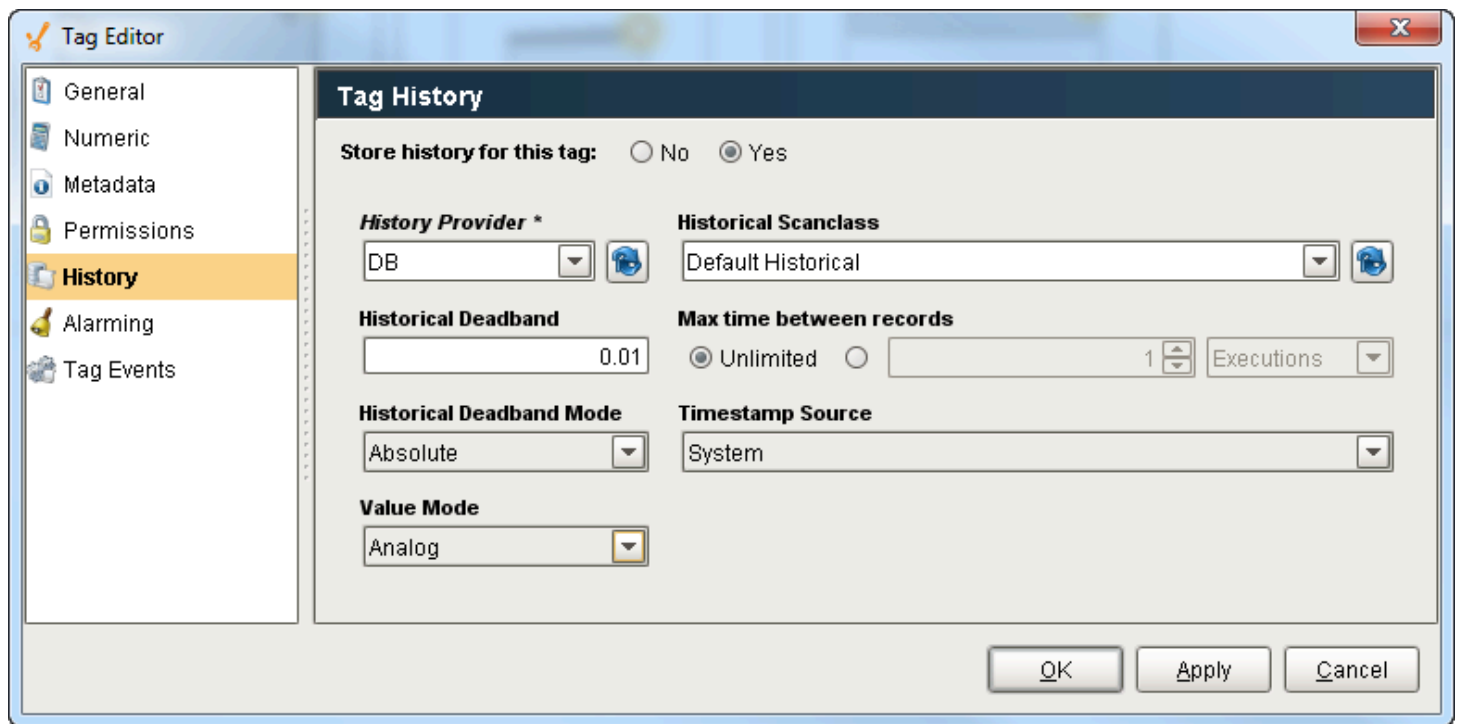
Logging data is easy with Tag Historian Module. Once you have a database connection, just set your Tags to store history. Ignition creates the tables, logs the data, and maintains the database. To learn how to log history for your Tags, go to [Set up Tags to Log](#).

NOTE

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.

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.



Historical Scanclass

Historical Scanclass determines how often to record the value on the Tag. It uses the same [Scan Classes](#) that dictate how often your Tags should execute. Typically, the Historical Scanclass should execute at the same rate as the Tag's Scan Class or slower: if a Tag's Scan Class is set to update at a 1,000ms rate, but the Historical Scanclass is set to a scan class that runs at 500ms rate, then the Tag History system will be checking the Tag's value twice between normal value changes, which is unnecessary.

Evaluate on Change

Additionally, the Historical Scanclass may be set to the `<Evaluate On Change>` option, which records every time the Tag changes value. This is useful if you wish to log every new value Ignition sees on the Tag.

NOTE

This scanclass setting will ignore the Max Time Between Records setting.

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: $|V_1 - V_0| \geq \text{Deadband}$.
- **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

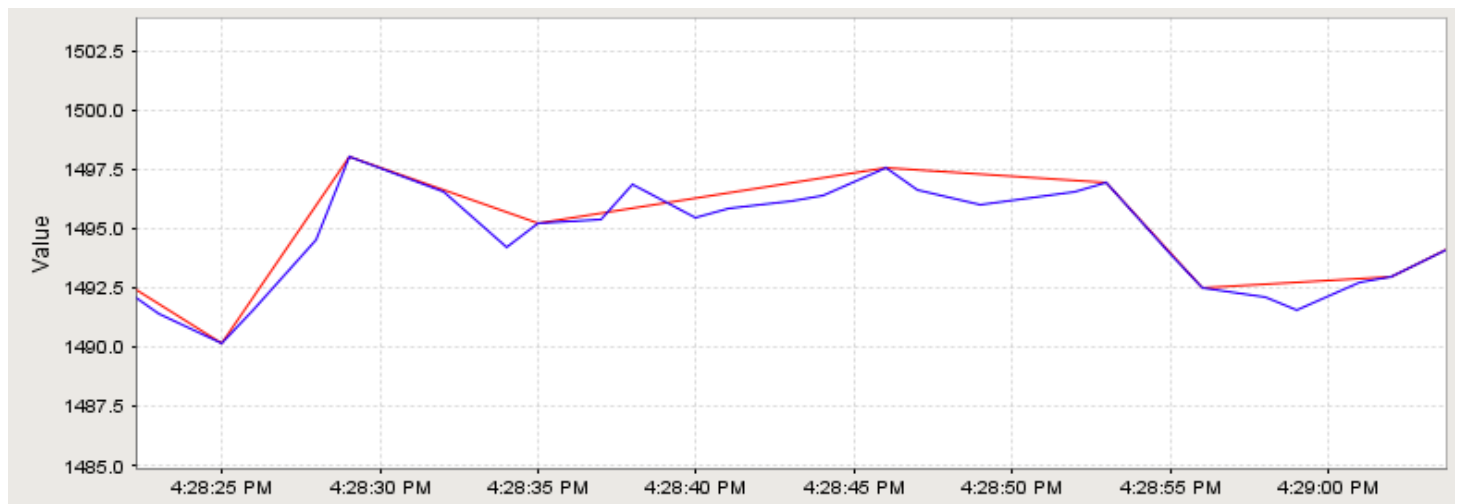
When a Tag executes, there are two possible timestamps that can be observed: the time associated with the data, and the time that the Tag was evaluated. The first case is generally only interesting when the value is provided by an OPC server. In most cases, the time provided by OPC, which in Ignition 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.

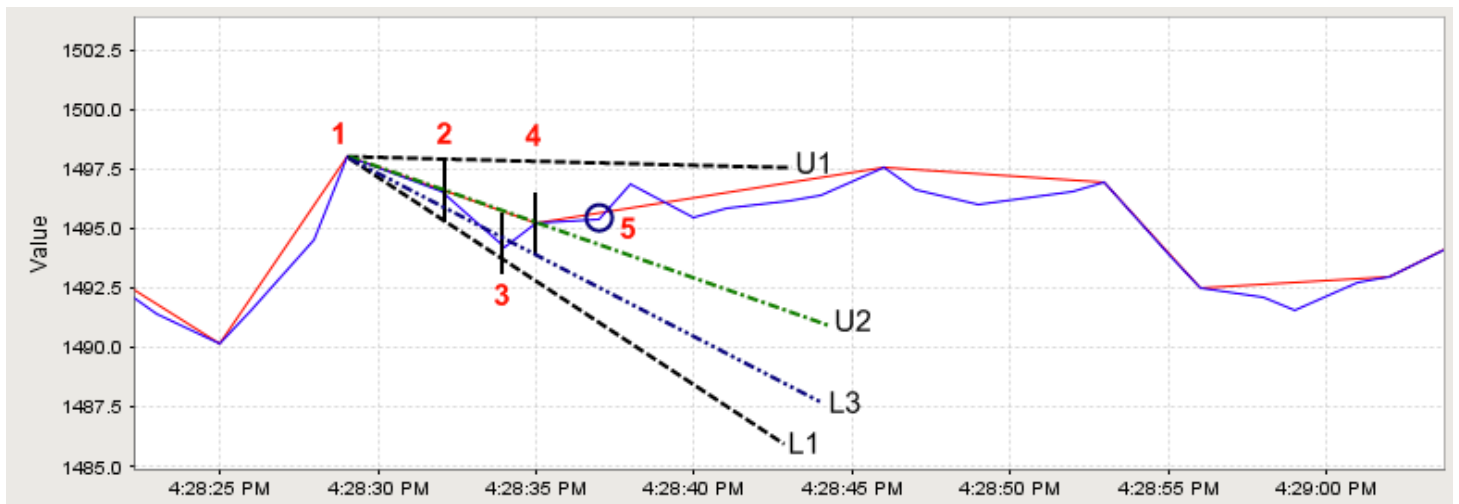
The Deadband, and Analog Compression

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 the process is started again from that point.

Seeded Values

Tag history queries sometimes use seeded values (occasionally called "Boundary Values"). When retrieving tag history data, the system will also retrieve values just outside of the query range (before the start time, after the end time), and include them in the returned result set. They're generally used for interpolation purposes. If the tag is storing history with an Analog Value Mode, or "Prevent Interpolation" is enabled on the calling query, then these seeded values will not be included.

Pre-Query Seed Value

These are a single value taken from just before the start of the query range. The value and timestamp for this value is typically the first row in the resulting query. Pre-query seed values are always included when not using a [raw data query](#).

An exception to this rule can be found with the [system.tag.queryTagHistory](#) function. Setting `includingBoundingValues` argument to True and `returnSize` to -1 will return a raw data query with a pre-query seed value.

Post-Query Seed Value

These extra values are added to the end of the result set, representing the next data point after the query range. Post-query seed values are only included when interpolation is requested/enabled for the query. Thus, values stored with a Discrete **value mode** will not include post-query seed values in the query results.

If the system knows the query is retrieving records for a tag on the local system, this value will be determined by the current tag's value instead of retrieving the last recorded value in the database. The current tag's value is also used in cases where the time range extends to the present time.

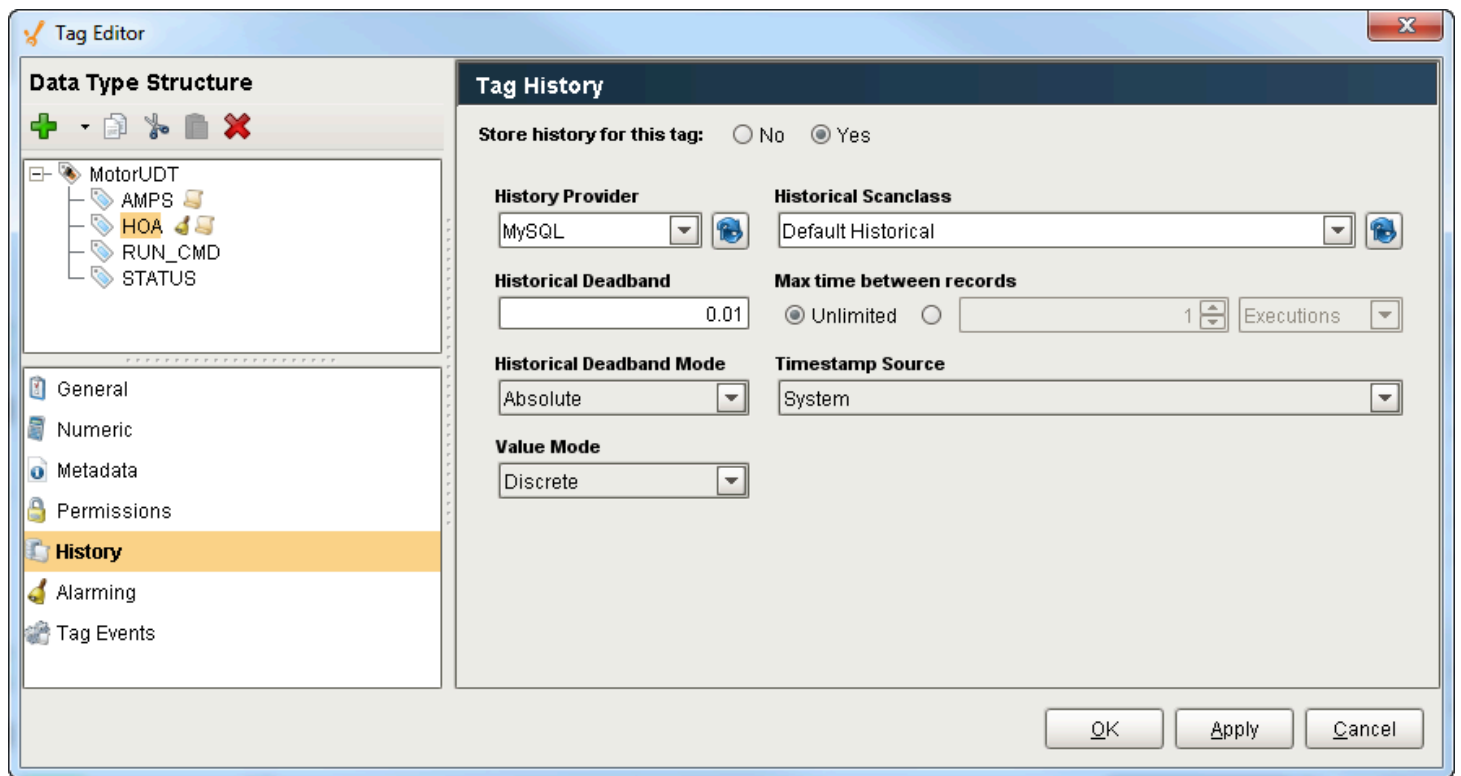
Raw Data Queries

In most cases, queries returned by tag history will apply some form of aggregation, as determined by the aggregation mode. However it is possible to get a "raw data query", which is a result set that contains only values that were recorded: meaning no aggregation or interpolation is applied to the results. A raw data query can be obtained by using one of the following options:

- Setting the Sample Size on Vision Tag History bindings to **On Change**
- Setting the **returnSize** parameter on **system.tag.queryTagHistory** to -1

Add History to Tags in a UDT

Setting up UDTs to log history data works on the same principle as **creating UDTs**. You dramatically reduce the amount of work when setting up UDTs to log history data so that all instances of that UDT will log data without having to edit each individual instance. To learn how to set the UDT to log history, refer to **Add History to Tags in UDT**.



Historian Providers

Tag History Providers are what drive the historian in Ignition. They typically require some sort of database that stores the data to be recalled at a later time in a trend or report. They are setup or configured in the Configure tab off the Gateway Webpage by going to **Tags > History**.

Historical Tag Providers

Provider Name	Enabled	Type	Description	Status	
NewConnection	true	Datasource History Provider		Running	edit

[→ Create new Historical Tag Provider...](#)

Datasource History Provider

The default history provider that automatically gets created for each new database connection that gets established and will be removed if that database connection is deleted. While it can be configured to allow for some different storage options, it is setup with a 1 month partition by default.

Remote History Provider



INDUCTIVE
UNIVERSITY

Remote History Provider

[Watch the video](#)

This provider provides a link to a provider on another Gateway. To use this type of provider, there must be an established Gateway Network. It can be used just to read history values that the remote provider has logged, or it can be configured to allow the local Gateway to write values to the remote Gateway.

Tag History Splitter

This provider combines two separate providers into a single new provider. When setting up a Tag to store history, selecting this provider will write the same data to both providers that it has selected. The Tag History Splitter is useful for automatically creating a backup of your data, or for reading history from two separate providers. Learn more about setting up the [Tag History Splitter](#) here.

OPC-HDA Provider

Establishes a connection to an [OPC-HDA Server](#) to read history data that may be stored there from a third party. Ignition can't write to this type of history provider.

NOTE

This requires the OPC COM module be installed.

Store and Forward



INDUCTIVE
UNIVERSITY

Store and Forward

[Watch the video](#)

The Store and Forward system provides a reliable way for Ignition to store historical data to the database. The Store and Forward system is not exclusively part of Tag History, but systems such as the Tag Historian and Transaction Groups use this to prevent data loss and store data efficiently using a record cache.

It is important to configure your Store and Forward system when using Tag history. For more information, see the following sections:

- [Store and Forward](#)
- [Using Store and Forward](#)
- [Configuring Store and Forward](#)
- [Controlling Quarantine Data](#)

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

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

Data Partitioning and Pruning



INDUCTIVE
UNIVERSITY

Data Partitioning and Pruning

[Watch the video](#)

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 **Historical Tag 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.

Historical Tag Provider Properties

Main

Property	Description
Provider Name	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.
Description	An optional user created description of the provider.

Data Partitioning

Property	Description
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

Property	Description
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 section will quickly cover how to do it in MySQL and MSSQL.

Both examples below assume the partition table is named 'sqlt_data_1_2016_08'.

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
```

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

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

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.



INDUCTIVE
UNIVERSITY

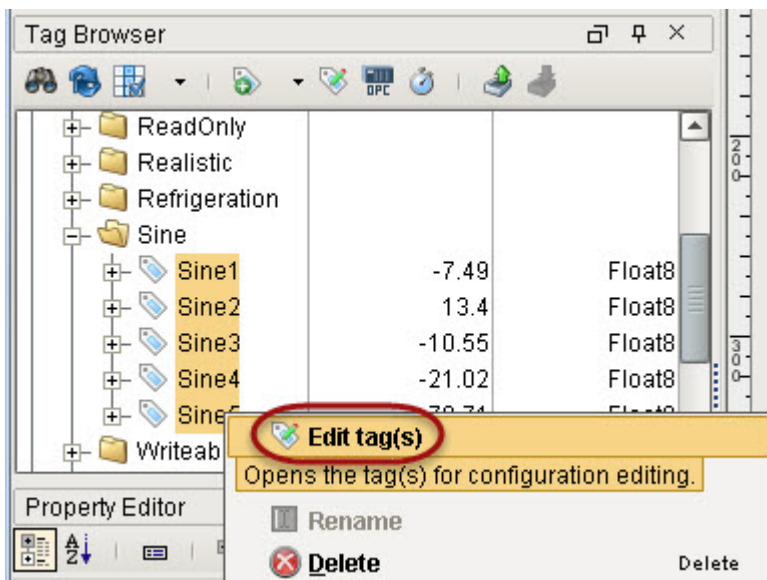
Set up Tags to Log

[Watch the video](#)

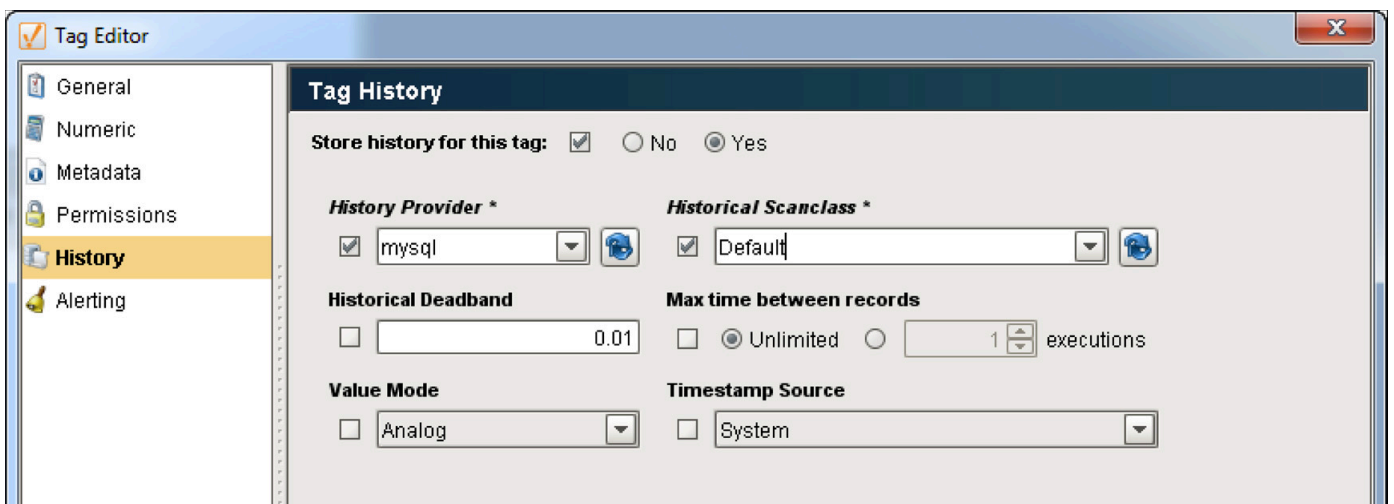
Log tag history data

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

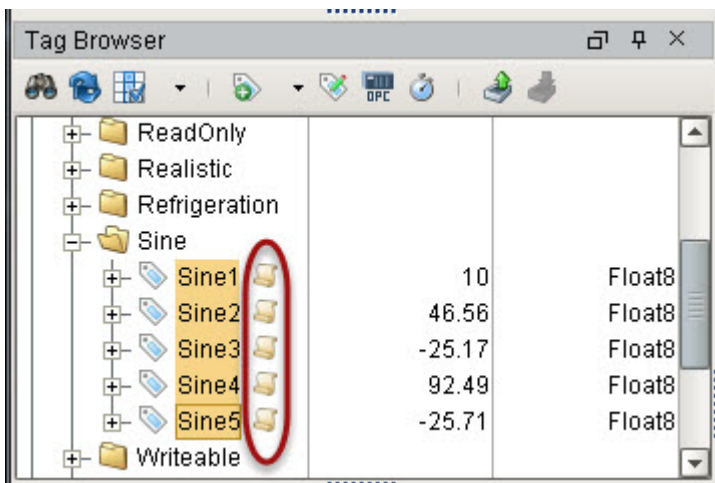
1. Go to the Designer, from **Tag Browser**, select one or more tags. For example, select all the **Sine** tags in the Sine folder.
2. Right-click on the selected tags, and then select **Edit tag(s)**. The Tag Editor window is displayed. Here, you can edit the tag and change the name, data type, scaling options, metadata, permissions, history, and alarming.



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.



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.



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

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.

Set the UDT to log history data

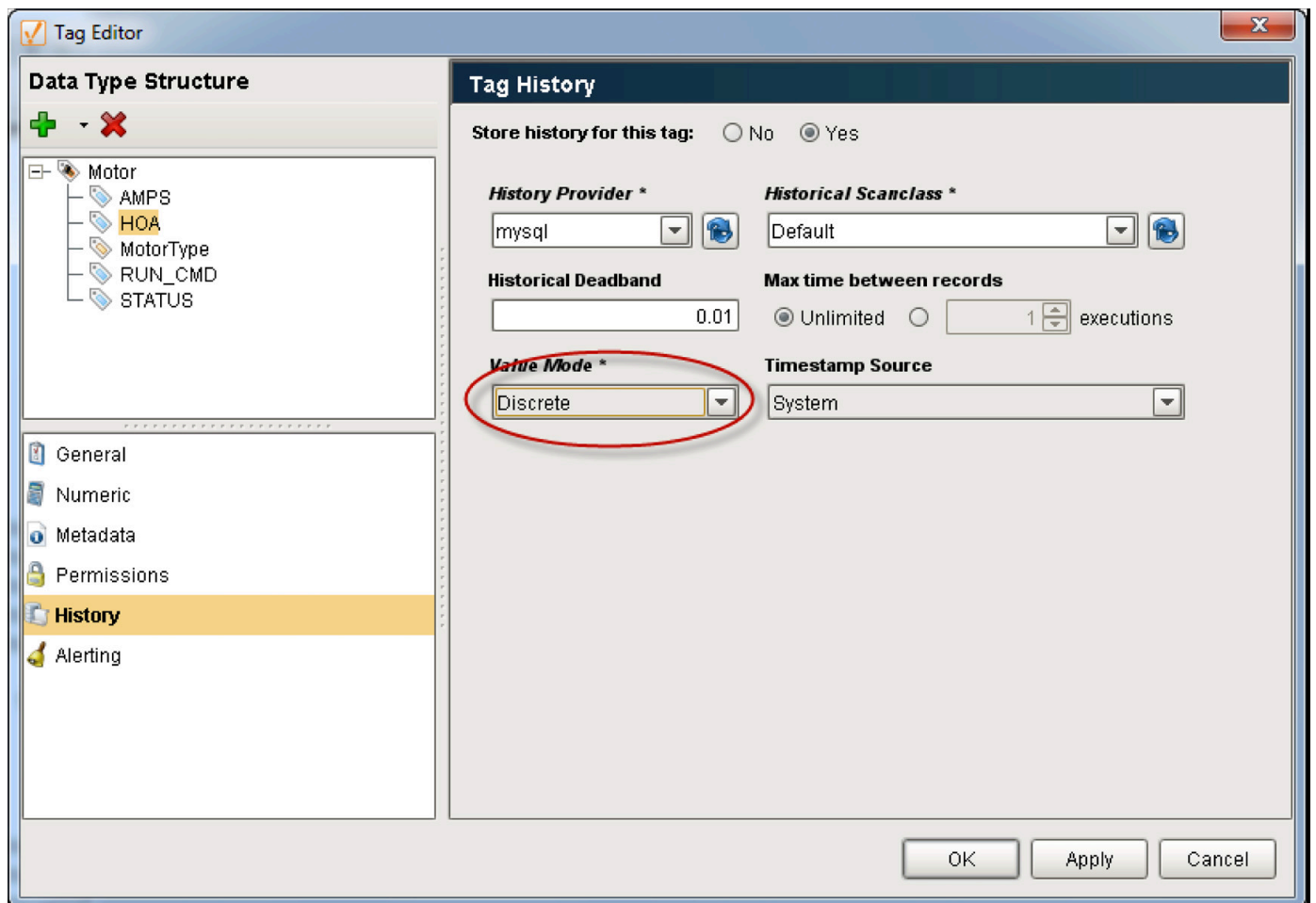


INDUCTIVE
UNIVERSITY

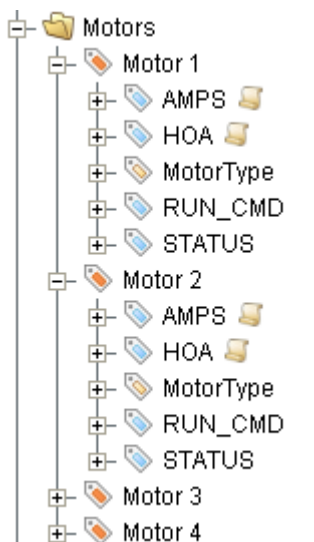
Add History to Tags in UDT

[Watch the video](#)

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.



5. Click **OK** to save.



Now every motor instance automatically starts logging data.

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

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

Tag History Gateway Settings

Configuring Tag History Settings

Storing Tag History is easy to get setup quickly, but there are also some settings that can be adjusted to allow for differences in database storage space and performance needs.

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. 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. See Value Mode, in History Properties for more information about the difference between Discrete and Analog values.



INDUCTIVE
UNIVERSITY

Discrete vs. Analog Mode

[Watch the video](#)

Datasource History Providers

Datasource History Providers can't be created or deleted, but are instead tied to a database connection. They are automatically added when connecting to a new database and removed after the database connection is removed. It comes pre-configured to partition every month, but the provider can be edited to change its behavior. There are two other major options to configure on the provider: pre-processed partitions and data pruning. With pre-processed partitions, the data that is stored is then summarized and then placed into another table in the database. While this takes up more space in the database, it can improve query speed by reducing the amount of data points that must be loaded. Data pruning will automatically remove old data from your system after it reaches an age that you set. It will only remove whole tables though, so if each partitioned table represents a month and the pruning system removes data that is 3 months old, it will wait until all the data in the oldest table is 3 months old before pruning it.

Datasource History Provider Properties

Property	Description
Provider Name	Name of the Tag History Provider. By default, this will match up with the name of the database connection.
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.
Description	A description of the provider.
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.
Enabled Pre-processed Partitions	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.
Enable Data Pruning	Partitions with data older than a specific age are deleted. The check box is not selected/enabled by default.

Property	Description
	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.
Enable Stale Data Detection	If enabled, tracks scan class executions to determine the difference between unchanging values, and values that are flat due to the system not running.
Stale Detection Multiplier	The multiplier for scan class rate used to determine when values are stale. If scan class execution is not recorded within this amount of time, values will be considered bad on query.

Remote History Provider

A Remote History Provider is a link to a historical provider on another gateway. Since it is grabbing historical tag data from another provider, its only configuration is to ensure it is pointed at the correct tag provider. You can't change any of the settings like partition length and prune age, but would instead have to change those settings on the original history provider on the remote gateway. By default, the remote history provider will fall under the **Default Security Zone** and be read only.

Remote History Provider Properties

Property	Description
Provider Name	Name of 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.
Description	A description of the provider.
Remote Gateway Name	The name of the remote Gateway.
Remote History Provider	The name of the provider on the remote Gateway. This does not have to match the provider name on the local Gateway.
Allow Storage	If false, the provider will only be used for querying historical data. If true, the provider will create a store and forward pipeline for sending data to a remote Gateway.

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

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

Sample Size and Aggregation Mode

In places where the Tag History system can be queried, a Sample Size and Aggregation mode can be selected that will determine how the results will be queried out and how the raw values will be aggregated.



INDUCTIVE
UNIVERSITY

Tag History Aggregates

[Watch the video](#)

Sample Size

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 $[[A0, B0], [A1, B0], [A1, B1]]$. 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.

Aggregation Mode

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).

- **Time-weighted Average** - The values are averaged together, weighted for the amount of time they cover in the interval.
- **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.
- **Closest Value** - The value closest to the ending time of the interval will be returned.
- **Basic 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.
- **Count** - Returns the number of times a value was recorded
- **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.

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

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

Custom Tag History Aggregates

Python Aggregation Functions

The Tag History system has many built-in aggregate function, such as Average, Sum, and Count. However a custom aggregate may be defined via Python scripting. 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.

CAUTION

When calling a custom tag history aggregate, the `returnSize` argument must be set to natural (`returnSize = 0`). If the `returnSize` is set to `-1`, or left with it's default value, the the custom aggregate will be ignored.

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 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.

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

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.

Examples

Using a Shared Script

This example assumes a Shared Scripts named "aggregates" contained the function listed below.

Example

```
# this is a simple count function, called for each value in a time window
def myCount(qval, interpolated, finished, blockContext, queryContext):
    cnt = blockContext.getDefault('cnt',0)
    if qval.quality.isGood():
        blockContext['cnt']=int(cnt)+1

    if finished:
        return blockContext.getDefault('cnt', 0)
```

The custom function could be used like in the example below:

Example

```
#Return tag history using a custom aggregate function you wrote.

system.tag.queryTagHistory(paths=['MyTag'], rangeHours=1, aggregationModes=
['shared.aggregates.myCount'], returnSize = 0)
```

Creating an Aggregate Function on the Fly

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], returnSize = 0)
```

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

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

Tag History Splitter



INDUCTIVE
UNIVERSITY

Tag History Splitter

[Watch the video](#)

Tag History Splitter Provider

The Tag History Module has a provider type called the Tag History Splitter. Like the Remote History Provider, it doesn't store history on its own, it relies on having other providers already set up. A Splitter provider simply logs Tag History into multiple existing History Providers.

Some users prefer to have the data that is recorded by the Tag Historian sent to multiple databases. For example, project specifications require redundant logging and users at another facility may like to have a copy of the data in their local database. In cases like this, the Tag History Splitter Provider is ideal.

The Gateway that these Tags reside in must have multiple Tag History Providers configured. Should one of the providers fault, the Store and Forward system will kick in to maintain the data on the faulted connection. Since each database connection has its own Store and Forward engine, the data is always forwarded to the correct database.

To use the Tag History Splitter

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 **Historical Tag 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 option. On the Tag Editor window go to the History tab and see the new splitter provider in the dropdown of the History Provider.

Tag History Splitter Provider Properties

Below are the properties available on the Historical Tag Provider.

Main

Property	Description
Provider Name	Name of the connection.
Enabled	Enables and disables the connection.
Description	Description of the connection. The description appears on the Historical Tag Providers page of the Gateway.

Storage

Property	Description
First Connection	Data is stored to both connections equally. However, all tag history queries (tag history bindings, <code>system.tag.queryTagHistory()</code> calls, reporting tag historian queries, etc.) execute against the first connection, unless a limit is imposed using the settings below, or the first connection is unavailable.
Second Connection	The second connection to store tag history.

Querying

Property	Description
Limit First Connection Query	If enabled, only queries that are within the time frame specified below will be executed against the first connection. Queries that go further back will only execute against the second connection.
Limit Length and Units	The unit and length of the time frame limitation mentioned above.

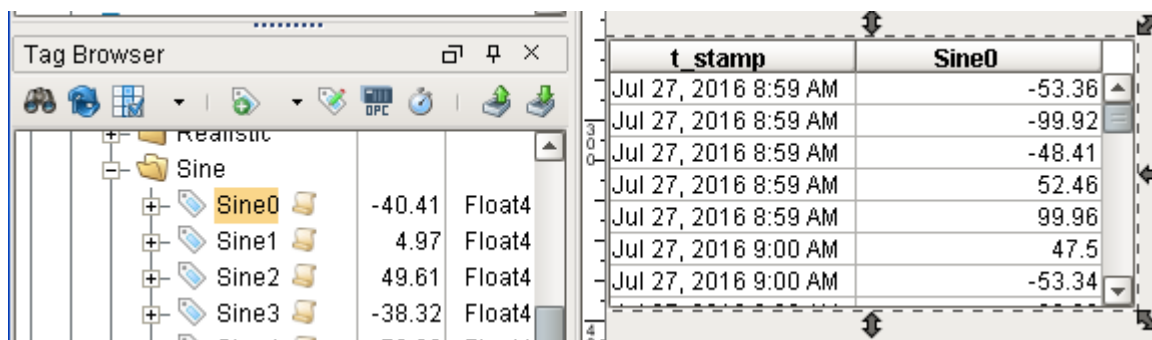
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

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

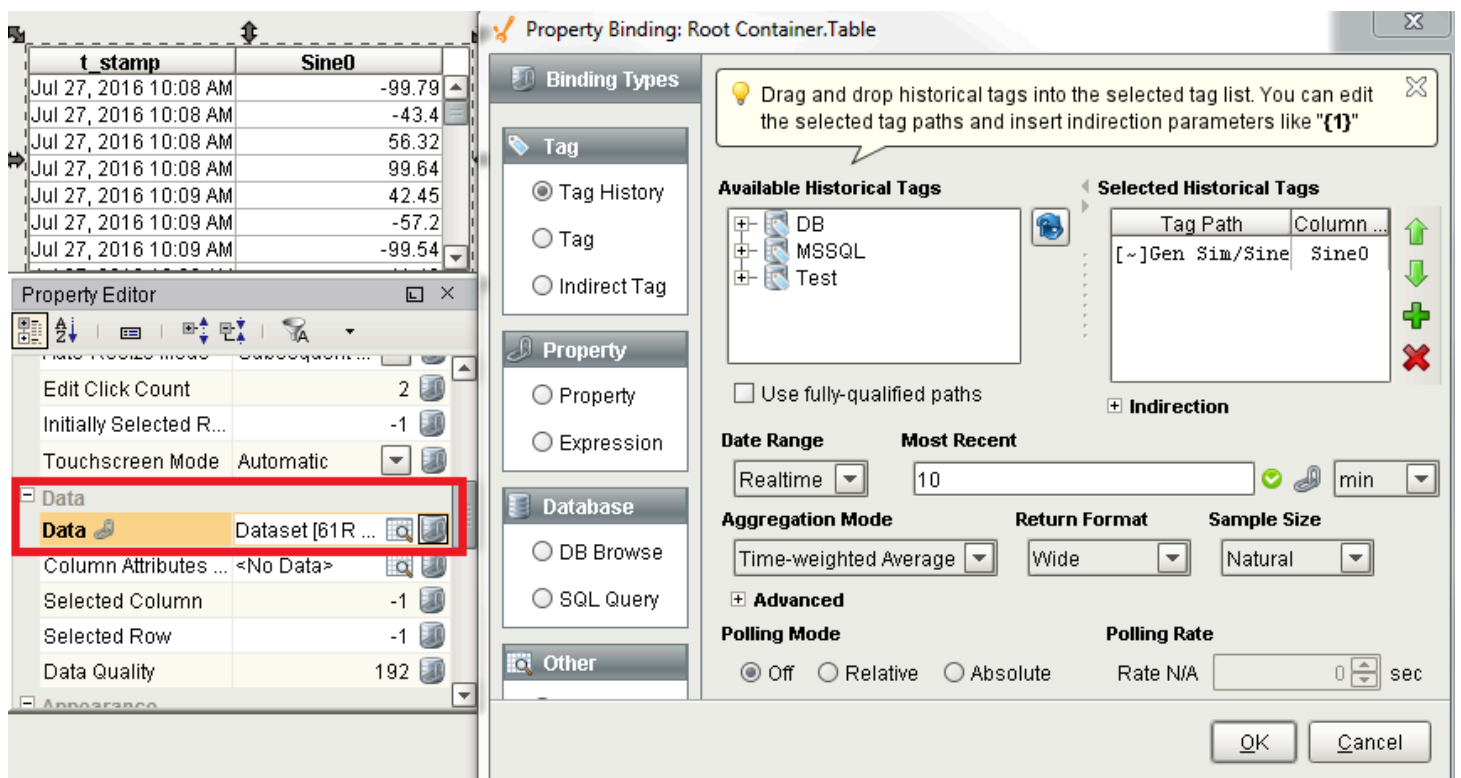
Tag Historian and Tables

Displaying Tag Historian Data

Displaying Tag Historian data on a Table component is easy! When the Designer is in design mode, simply drag and drop a historical tag (a tag with history enabled) on to a Table component.



When the tag was dropped, the Designer will recognize the action and create a **Tag History Binding** on the Table's Data property. The default configuration on the binding uses a **Realtime** date range looking at the most recent **10 minutes**. The Sample Size is **Natural**, and polling is **off**.



Manually Creating the Binding

Instead of dragging-and-dropping the tag, you could instead create the binding manually by editing the Data property on the table. While this approach is slightly slower to implement than dragging-and-dropping the tags, it has some benefits, such as filtering out non-historical tags, and allowing immediate access properties on the binding. Additionally, components that do not support drag-and-drop could be used instead.

To manually create a 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. 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**.

From the Binding window we can make several configurations changes to modify how the data is displayed. The following information on this page demonstrates several different configurations of the Tag History Binding applied to a Table component. However, these examples could easily be adapted to other components that display data by timestamp, such as the standard Chart component. Even a dataset-type custom property could be used.

Date Range Property

Once the binding has been created, the displayed data can be altered and tweaked in several ways. The following sections on this page give some examples of such alterations.

Date Range - Realtime



INDUCTIVE
UNIVERSITY

Table – Most Recent

[Watch the video](#)

The Realtime date range will show history between now, meaning the last time the binding polled for data, and some unit of time in the past. This mode is great if you need to show history spanning the last several X units of time. Note that tag history bindings update when polling, so creating a a real-time chart is a breeze when polling is turned on.

Tag Path	Column Name
[~]Gen Sim/Sine/Sine0	Sine0

Date Range - Historical



INDUCTIVE
UNIVERSITY

Table – Data Range

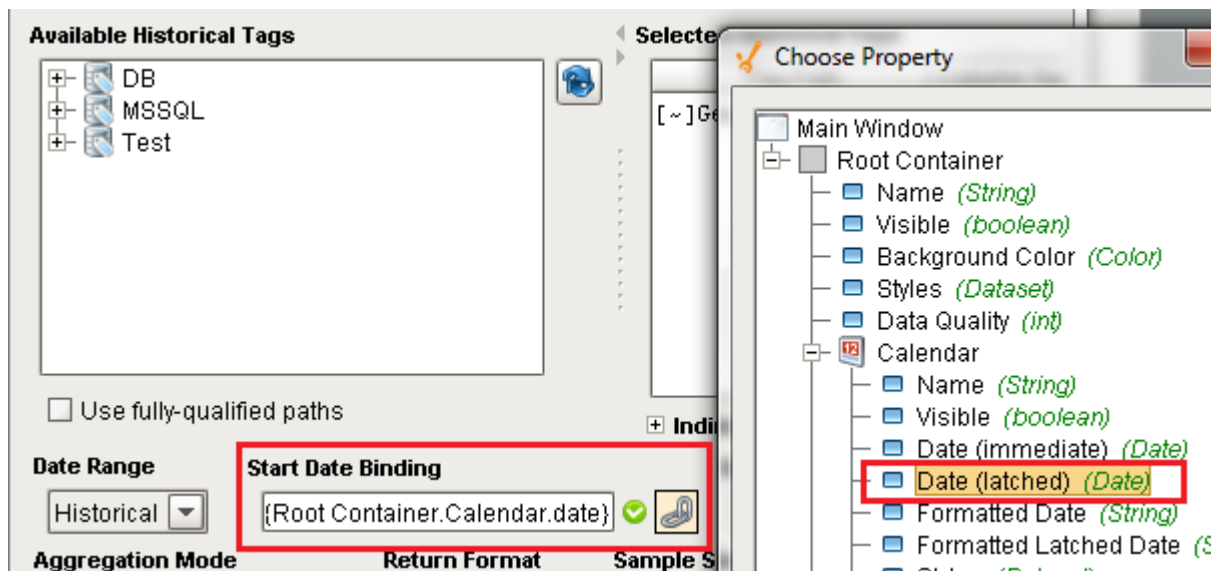
[Watch the video](#)

The Historical date range will show historical data between two points in time.

Date Range	Start Date Binding	End Date Binding
Historical		

When selected, a Start Data Binding and End Date Binding property will appear. There must be a date property on the window for these property to bind  to. Commonly a Calendar or Date Range component is used, so users can change the date range from the client. Below is an example of a binding that is making use of a Calendar component. The component is in the same container as the Table. The Start Date Binding property on the Table is bound to the Calendar's Date(latched) property.

A similar binding should be make with End Date Binding, but with a different calendar. This way, users will be able to select the exact range for the table to display.



Sample Size Property

The following examples focus on the Sample Size property to determine how many datapoints the binding should return. More details can be found on the [Sample Size and Aggregation Mode](#) page.

Sample Size - Fixed



INDUCTIVE
UNIVERSITY

Table – Fixed Sample Size

[Watch the video](#)

You can use the **Sample Size** and **Aggregation Mode** on the **Tag History** binding type to fix the number of records that are retrieved. The Fixed sample size will cause the binding to retrieve all records from the date range, and aggregate them evenly between a fixed number of points. This will ensure that the number of rows will remain the same without regard to the size of the data-set. In windows where users are able to select a large range of data, Fixed is recommended as it will prevent the property from loading an excessive number of records.

In cases where the number of points can not evenly represent the data from the date range, an extra point will be added, making the final size of the dataset the fixed value + 1.

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

The following image shows a Tag History Binding pulling data from the last 1 day. The **Sample Size** is configured to **Fixed** with a value of **100**, and the **Aggregation Mode** is set to **Basic Average**. This means that the binding will query for data from the last 1 day, regardless of how many records there are, and create 100 time-slices that are evenly dispersed between the start and end periods of that range. Then, a basic average of the tag values are calculated for each time-slice. The resulting values are then returned to the property.

Available Historical Tags

- DB
- MSSQL
- Test

☐ Use fully-qualified paths

Selected Historical Tags

Tag Path	Column Name
[~]Gen Sim/Sine/Si	Sine0

Date Range Most Recent

Realtime 1 day

Aggregation Mode Basic Average

Return Format Wide

Sample Size Fixed 100

Advanced

Polling Mode Off Relative Absolute

Polling Rate Rate = 10 sec

Note the binding icon  next to Sample Size. This allows a property binding to determine the number of data points, so you could change the size to increase or reduce the amount datapoints on the chart from the client.

Sample Size - Interval



INDUCTIVE
UNIVERSITY

Table – Interval Sample Size

[Watch the video](#)

Whereas the **Fixed** sample size will calculate time-slices based on the date range, the **Interval** sample size allows you to determine the size of the time-slices. This sample size will divide the date range by the interval size to determine the size of each slice. Because of this, it is recommended to use an interval that is evenly divisible by the date range. However, in the event that the date range is dynamic or user driven, interpolation will handle any partially built slices. Even though the binding may attempt to evenly distribute the slices, there may be an extra row that represents the current values as they are building an interval.

The image below shows a **Realtime** range of **60 minutes**. The Aggregation Mode is set to **Time-weighted Average**, and the Sample Size is set to **Interval 5 minutes**. This means that the binding

will query for data ranging from 60 minutes ago to now (or whenever the binding last executed, in the case that polling has been turned off). That 60 minute window will be divided as evenly as possible into 5 minute time-slices, so there should around 12 time-slices. Each time slice will aggregate its value based on the time-weighted average of all values within that slice.

The example uses a Realtime range, but a Historical range could easily be used instead.

Available Historical Tags

DB

MSSQL

Test

Selected Historical Tags

Tag Path	Column Name
[~]Gen Sim/Sine/Sine0	Sine0

↑

↓

+

×

☐ Use fully-qualified paths

☒ Indirection

Date Range

Most Recent

Realtime

60

✓

min

Aggregation Mode

Return Format

Sample Size

Time-weighted Average

Wide

Interval

5

✓

min

Advanced

Polling Mode

Polling Rate

☐ Off

☐ Relative

☒ Absolute

Rate =

10

sec

Again, the values of Most Recent and Sample Size can be bound using the binding icon  , so the size of the slices can change in the client.

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

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

Table - Indirect Tag History Binding



INDUCTIVE
UNIVERSITY

Indirect Tag History Binding

[Watch the video](#)

You can use indirection to configure the tag history binding to point to a set of tags based on any parameter on the screen.

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**.

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

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

Using the Easy Chart



INDUCTIVE
UNIVERSITY

Basic Easy Chart

[Watch the video](#)

Trends Made Easy

There are several types of components that are capable of trending historical data. The most popular component would be the Easy Chart. This component is simple to initially configure, and contains many ways to customize the look and behavior of the chart.

The Easy Chart was developed with the Tag Historian system in mind, so once an Easy Chart has been created, historical tags can be dragged-and-dropped onto the chart. The chart will immediately fetch the results and trend the history. Non-Tag-Historian can also be displayed on the chart as well: as long as the data has timestamps associated with the values, database pens can be created.

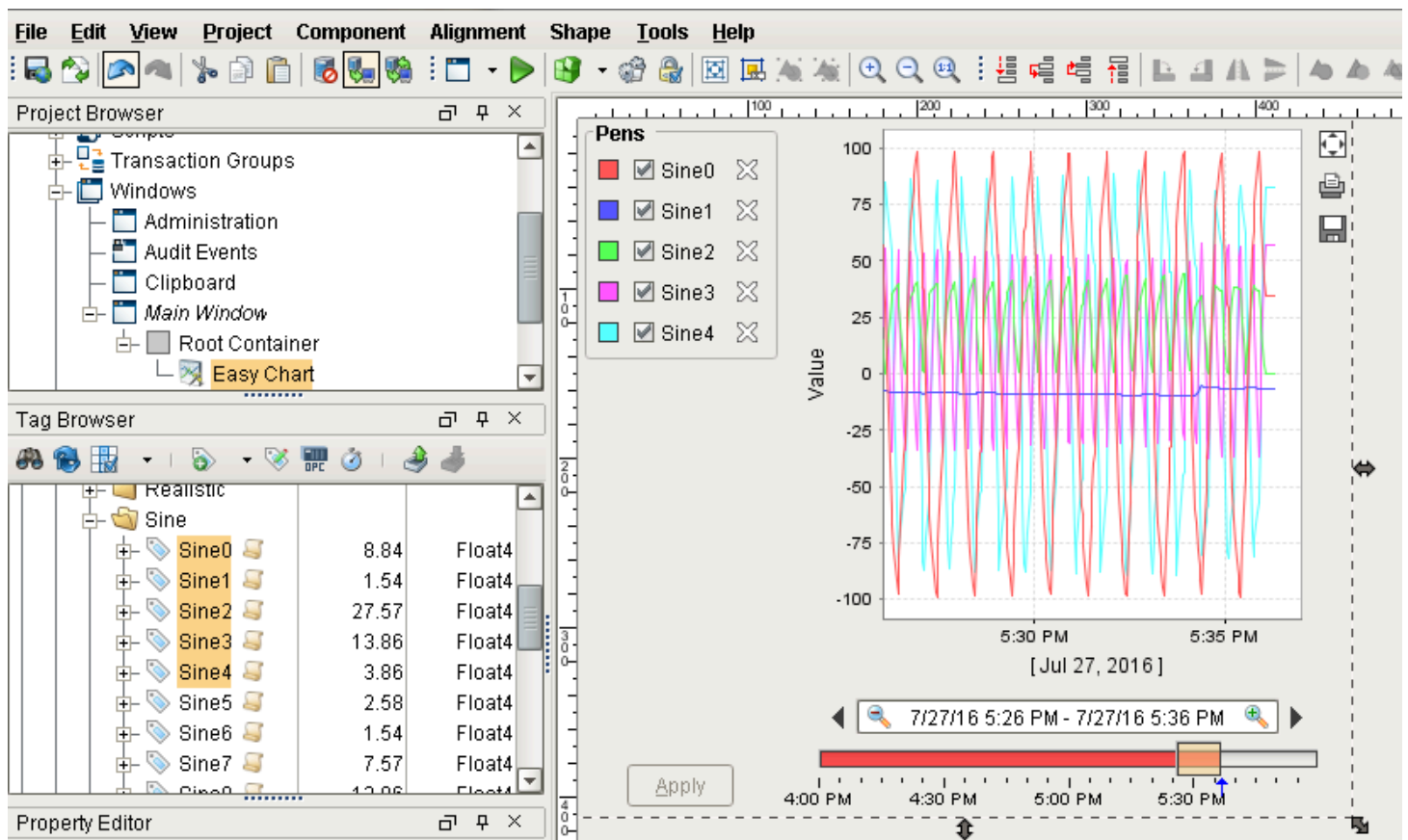
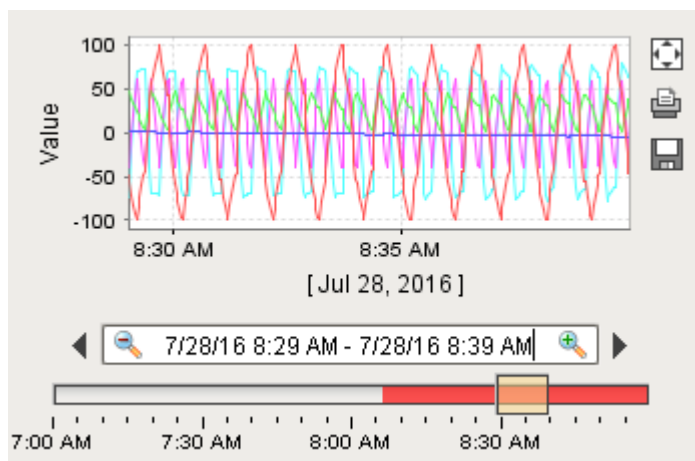


Chart Modes

The Easy Chart has a Chart Modes property that changes the behavior of the chart in several ways.

Historical

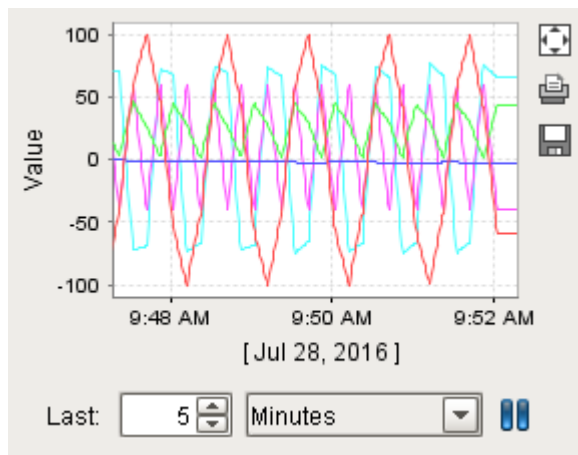
This mode places a date range selector at the bottom of the chart. This mode allows users to select a start and end date for the trends. The data density is shown at the bottom of the chart: the more vibrant the color, the higher the density. This is the default mode, and commonly used in situations where users need to look at specific date ranges. It is important to remember that the chart does not poll in this mode. New values are only added to the chart when the selection box is moved or re-sized.



Realtime

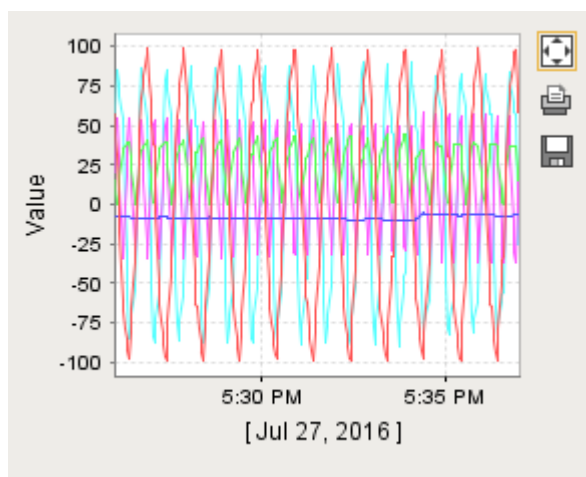
Displays the most recent data for each pen. Users are able determine how far back in time the trend should display with the spinner and drop down at the bottom of the chart. In this mode, the chart polls for data at the rate specified by the Poll Rate property.

In some cases, you may notice that the most recent values on tag pens tend to flat-line, and then 'snap' to a different value. This is generally due to how often the chart polls versus how often history is being generated. If the chart polls at a 1,000ms rate, but history is only recorded at a 10,000ms rate, then the chart will extrapolate the last recorded value for 9,000ms. After a new value is recorded, the next poll will return the latest value, and the flat-line will change position.



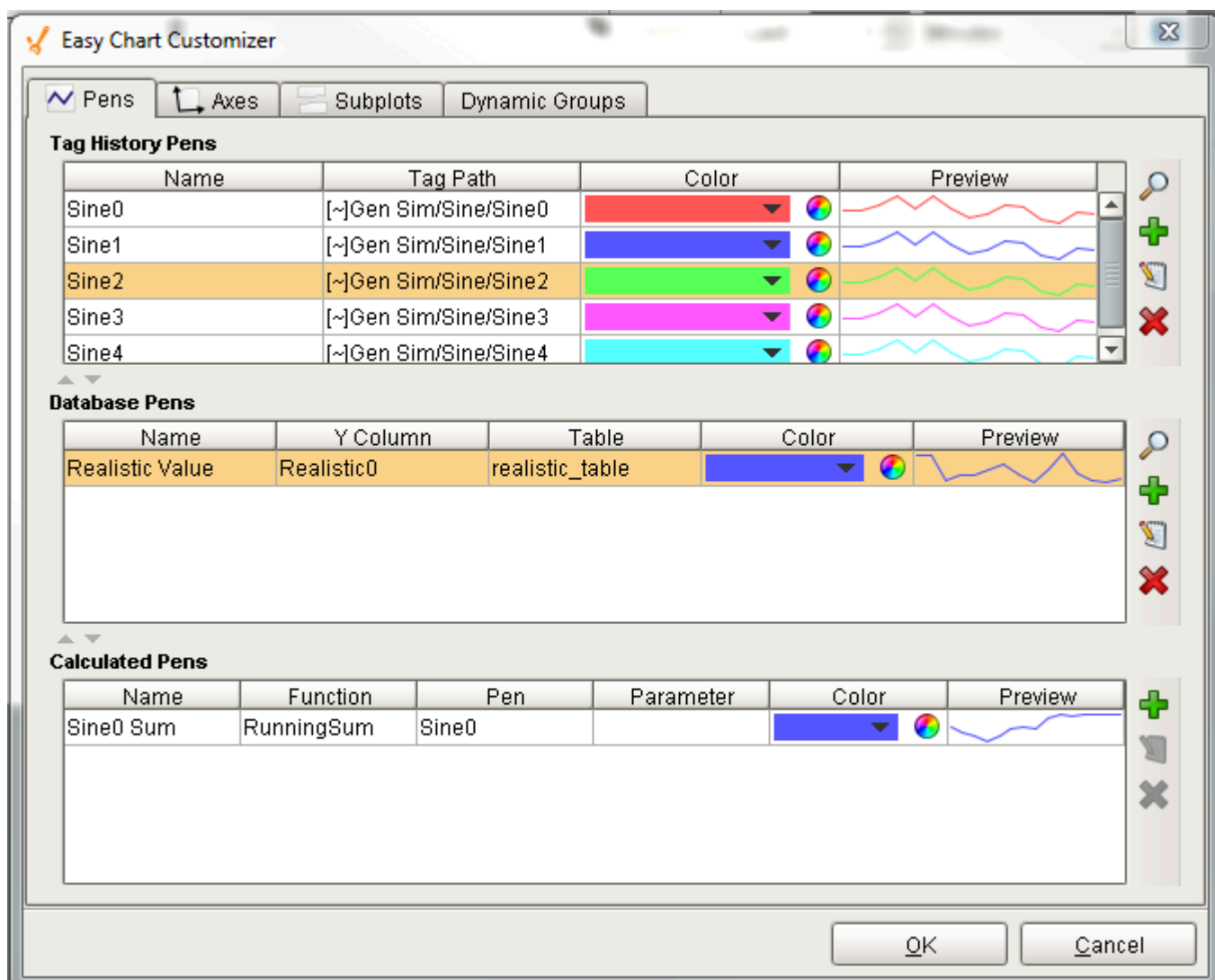
Manual

Similar to the Historical mode in that trends range from two points in time. However there is not a built-in method for users to change the data range. Instead, some sort of binding can be applied to the chart's Start Date and End Date property. This mode is generally used in situations where only certain date ranges should be shown, such as values recorded during the previous day, or shift.



Pens

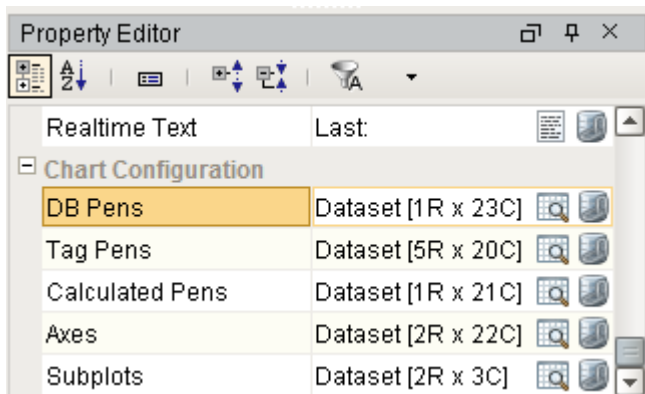
Pens on the chart, or each series of data points on the chart, can be customized to take on a number of different styles and colors. There are three types of pens, and each pen functions in a similar manner. What makes them different is how their data is collected.



- **Tag Pens** - These pens are driven by the tag history system. Data from any historical provider can be used, and tag history from different databases can be shown on the same chart. These are the type of pens that are created when tags are dragged onto the chart. Since the Tag History system is being used, an Aggregation Mode must be selected, and the Tag Path needs to be specified for each pen.
- **Database Pens** - These pens are driven by a SQL query, so they are ideal to use when trending Transaction Group data. However they can query for data in any connected SQL database, so it is possible to show historical data recorded by other systems on the Easy Chart.
- **Calculated Pens** - Pens that derive their data from calculations performed on other pens. Data for calculated pens is not stored directly into a database, but rather calculated in the runtime

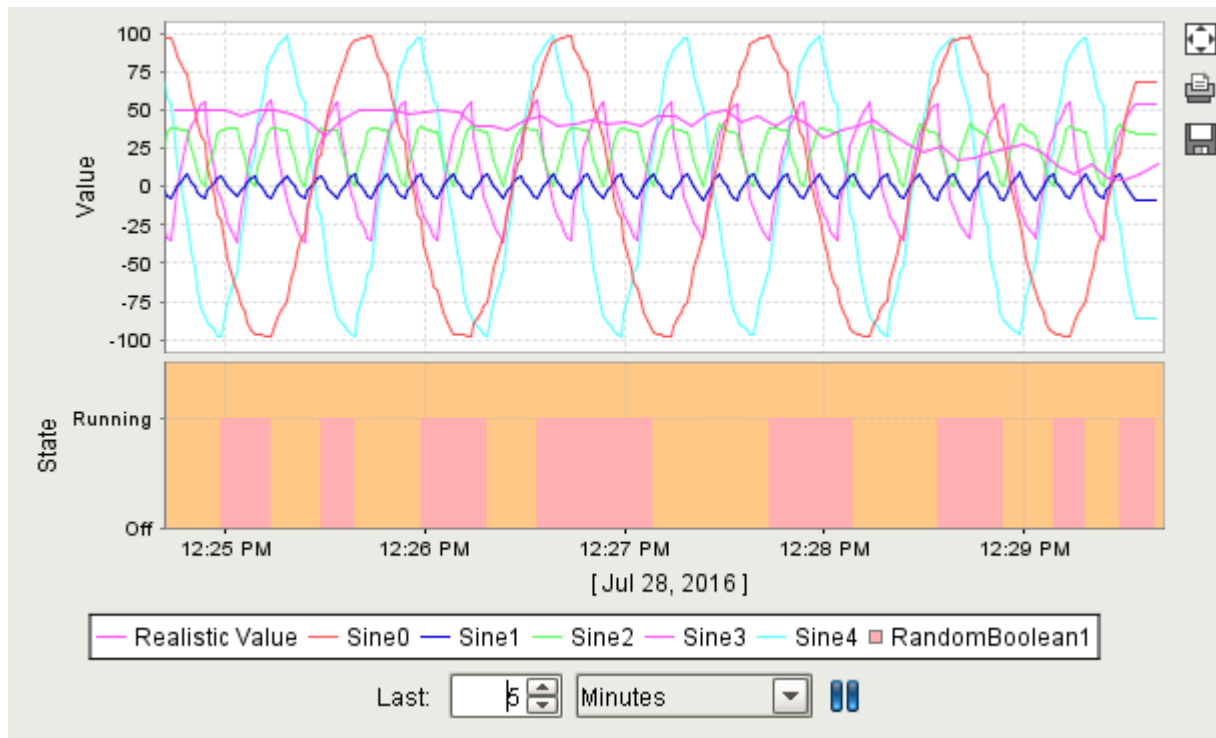
based on data from another pen. These type of pens are great for display running totals, control limits, or specification limits.

Pens can be added manually to the chart with the Easy Chart Customizer, or dynamically by modifying the various pens properties listed under Chart Configuration on the chart. These properties contain the configuration of each type of pen, and can make use of the binding system. Because of this, pen preferences can be saved to a database table, and then queried in the runtime with a SQL binding. Additional adjustments can be made with Cell Update Bindings or scripting to create a dynamic-yet-robust chart.



Easy Chart Customizer

Aside from the properties on the component, the Easy Chart has its own **Customizer** that allows modifications to be made to the chart. Aside from configuring pens, the customizer can be used to create **subplots** and new **axes**. Once created, each pen can be assigned to any available axes or subplots. This way different values can be shown on different plots, with an axis that is specific to data at hand.



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

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

Easy Chart - Axes



INDUCTIVE
UNIVERSITY

Easy Chart - Axes

[Watch the video](#)

ⓘ THIS SECTION ASSUMES THAT TAGS AND TAG HISTORY HAVE BEEN CONFIGURED

To learn more, go to the [Tag](#) and [Set up Tags to Log](#) pages. The examples below use [OPC tags](#) from the [Generic Simulator](#) driver, but [Memory, Expression, and Query Tags](#) can be used instead.

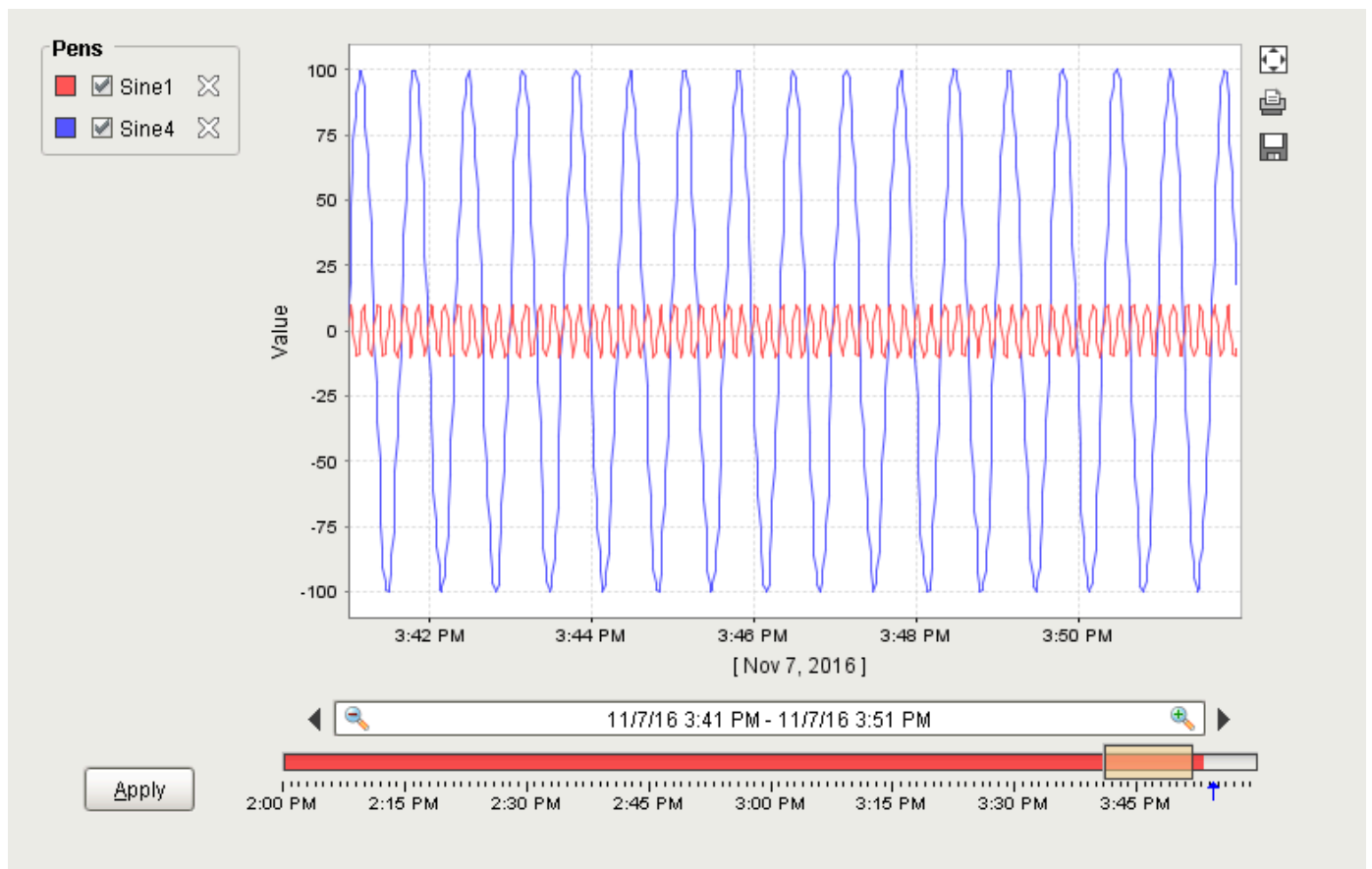
Configuring Multiple Axes on an Easy Chart

The Easy Chart supports the use of multiple axes for displaying data from the Tag Historian.

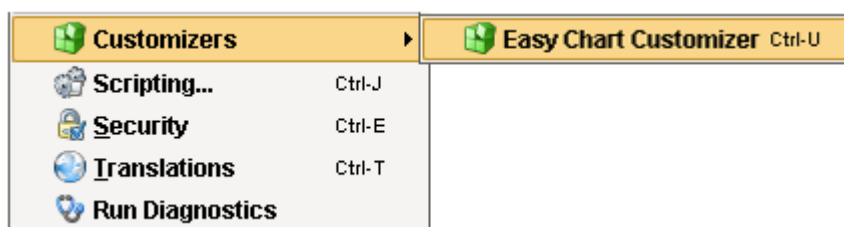
Now, let's configure multiple axes on an Easy Chart component:

1. From the component palette, drag an Easy Chart component to your workspace.
2. Drag your Tags over from the Tag Browser to your Easy Chart that you want to display.

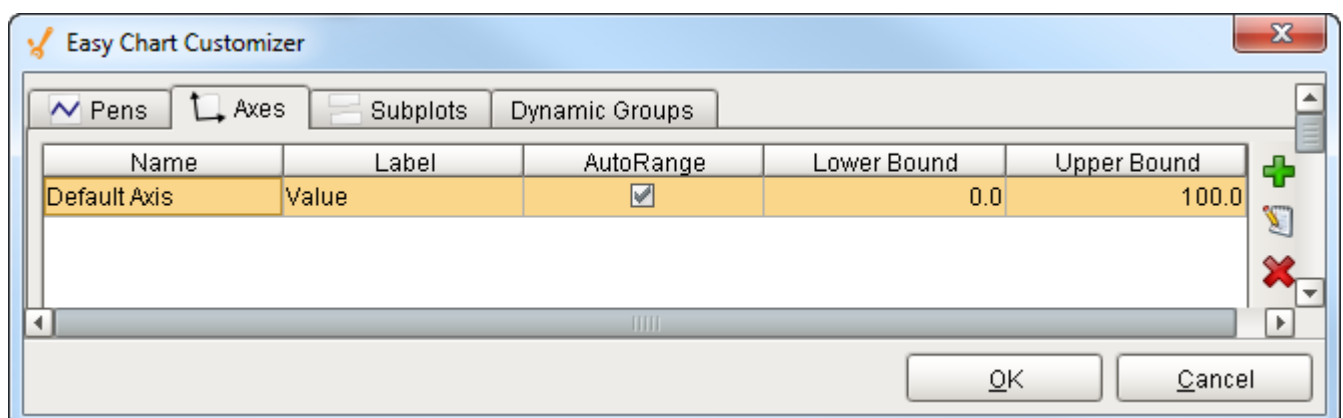
The following chart shows two Tags: Sine 1 and Sine 4. They both have completely different value ranges. Sine 1 values range between -10 and 10. Sine 4 values range between -100 and 100. Since both Sines are on the same axis, it is hard to see the details of Sine 1 values because Sine 4 is throwing off the axis due to its wide range of values.



3. You have the option of putting 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**.



4. Under the **Axes** tab, you'll notice that there is already one axis showing called Default Axis which both Tags are sharing. To add an axis to a component, click on the **green plus icon** .



5. The Edit Axis window will be displayed. Assign the **Axis** a name. In this example, it's the name of the Tag that is being used.
6. Enter a **Label** name, which is a name that you want users to see on your chart.
7. Select the **Type** of axis from the dropdown: Numeric, Logarithmic, or Symbol. This example uses the default Numeric. Specify the Label, Tick Label and Tick Colors that you want to set for your axis.

The **Position** property determines which side of the chart the Axis should be drawn on. By default, this property is disabled because the Easy Chart automatically attempts to position each Axis. To manually determine the position of an Axis, locate the **Auto Axis Positioning** property in the Property Editor of the Easy Chart component, and set it to 'False'.

By default, the **Auto Range** is set to 'true' and will apply padding so the pens do not draw at the top and bottom of the axis. Instead of having the Easy Chart automatically determine the range, Auto Range could be set to 'false', in which case the **Lower Bound** and **Upper Bound** properties will determine the full range of the axis.

Edit Axis

General

Name: Sine 4

Label: Sine 4

Type: Numeric

Position: Left

Label Color: [Blue]

Tick Label Color: [Blue]

Tick Color: [Blue]

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

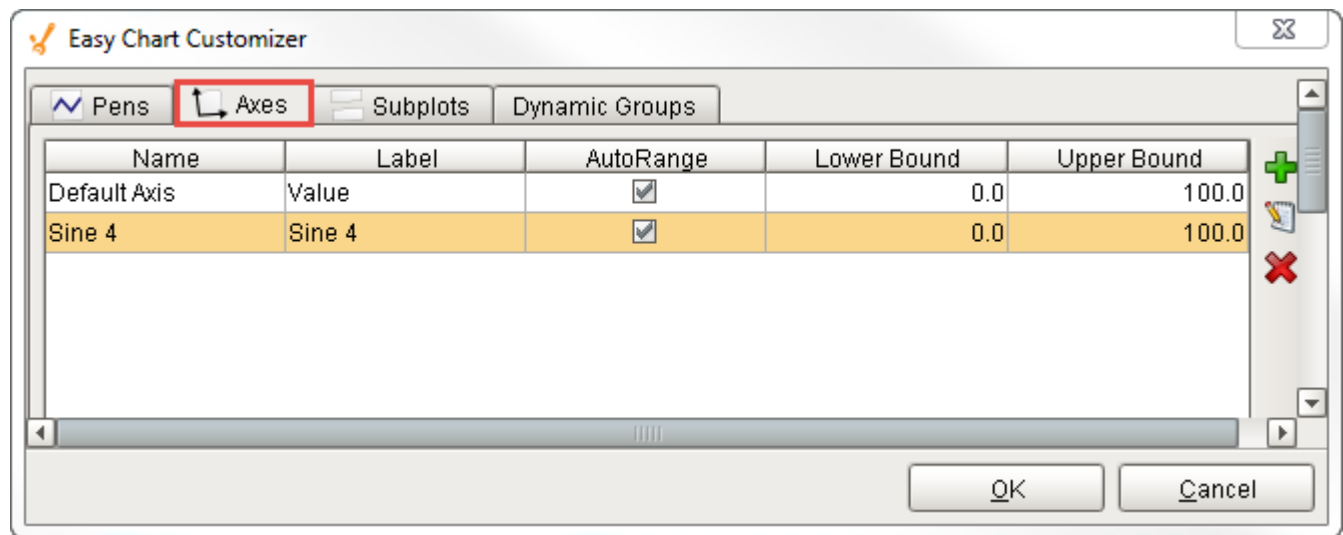
Tick Units: 5

Gridline Units: 5

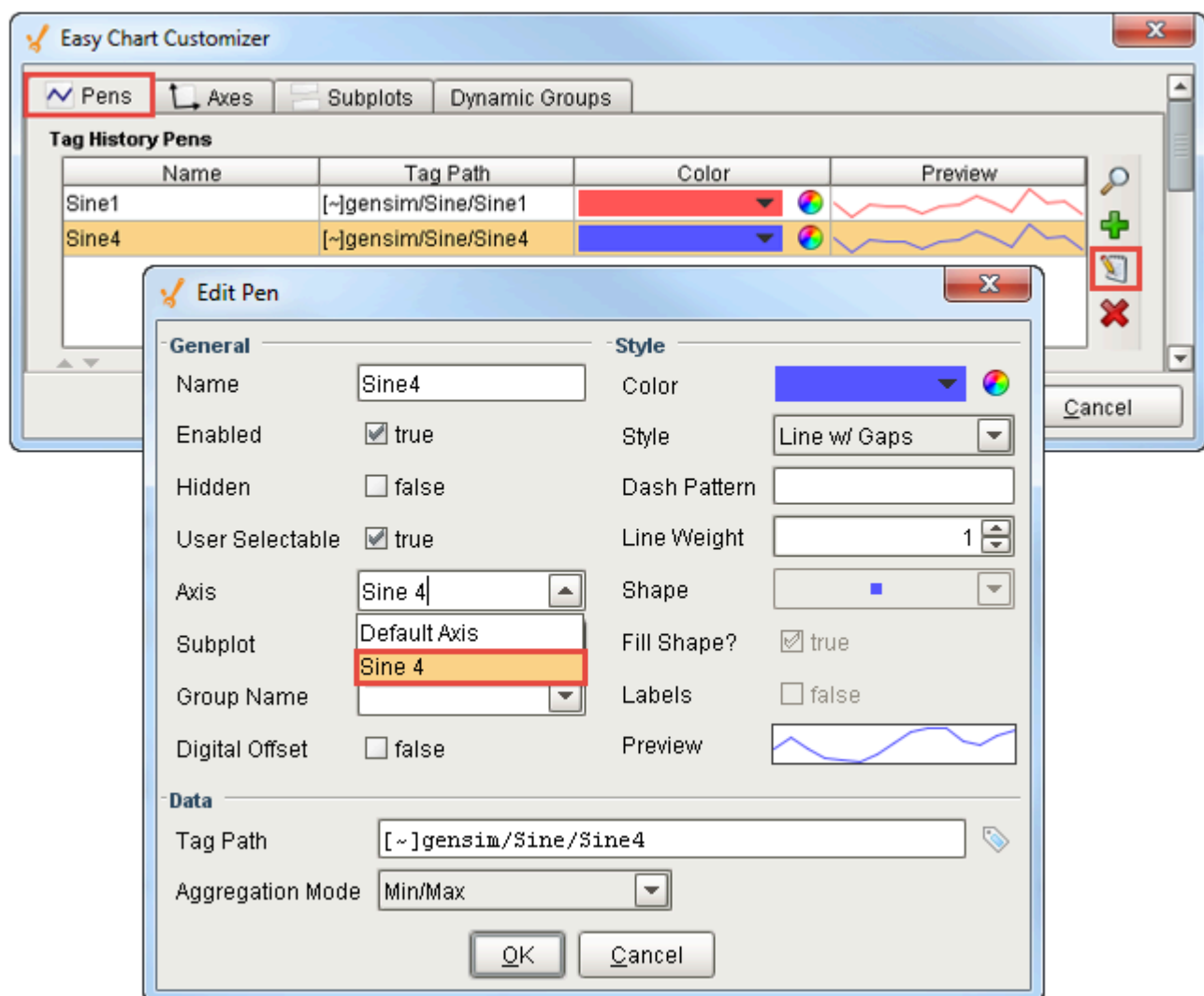
Number Format Override:

OK Cancel

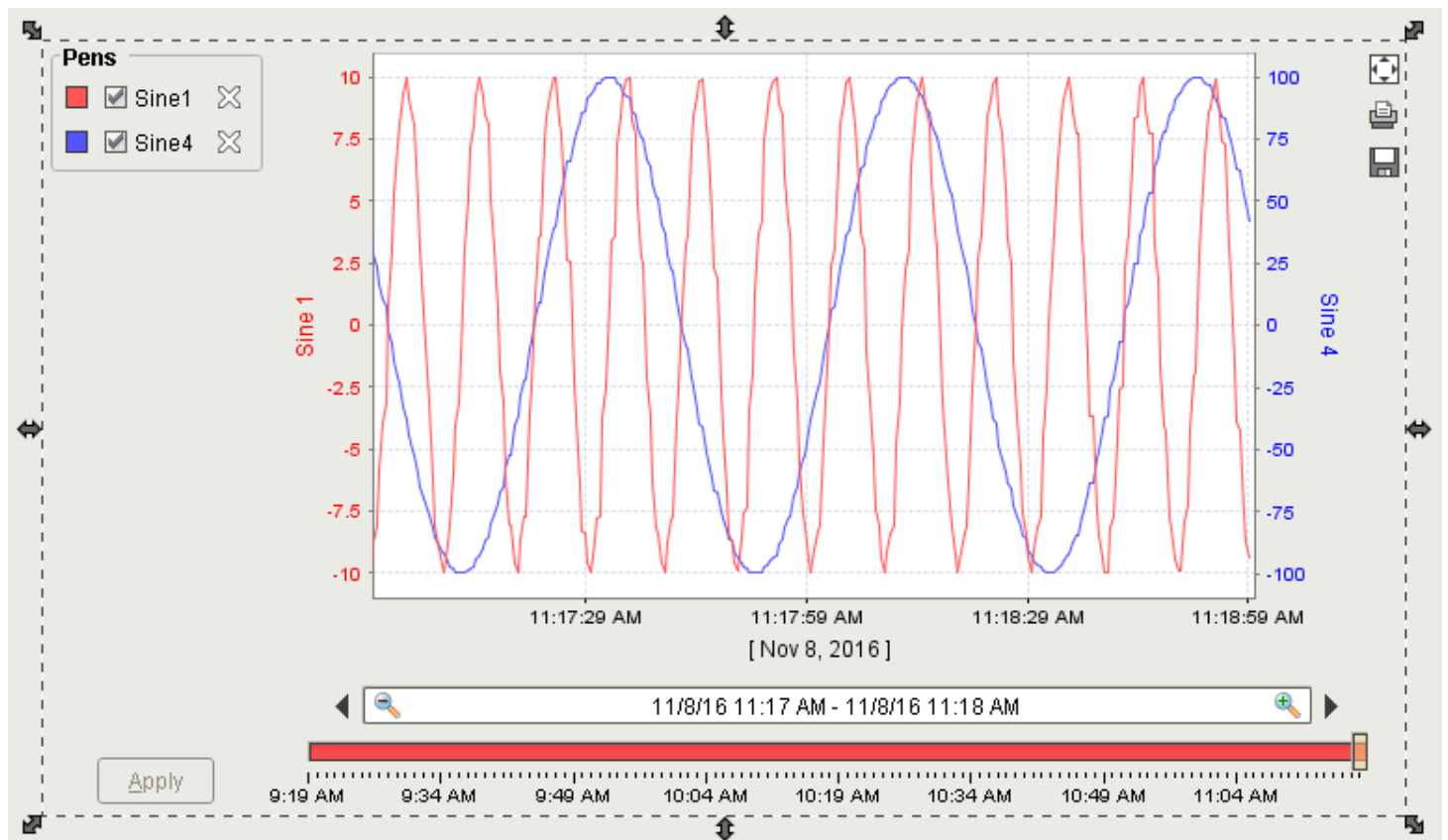
Now, you have two axes: Default Axis and Sine 4 axis.



8. Once a new axis has been created, you need to assign a pen to the axis. Press the **Pens** tab, select the pen row you want to change, and click the **Edit icon** . This example uses the 'Sine 4' pen. In the **Axis** field, select the 'Sine 4' axis from the dropdown menu, and press **OK** to close the Edit Pen window, and press **OK** again to close the Easy Chart Customizer.



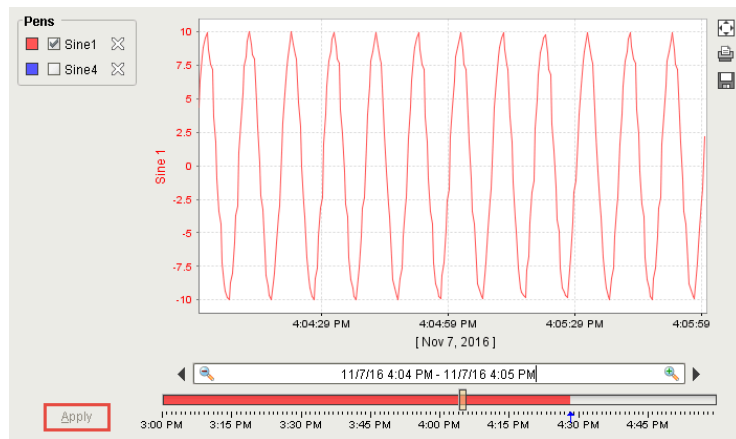
You'll notice that each pen is now in a different axis: Sine 1 is in the Default Axis, and Sine 4 is in the Sine 4 axis. On the right side can see Sine 4 axis and its values. On the left side, you can see the Default Axis (Sine 1) axis and its values. Now, you are using a different axis for each pen, and one pen is not going to throw off the values for the other pen.



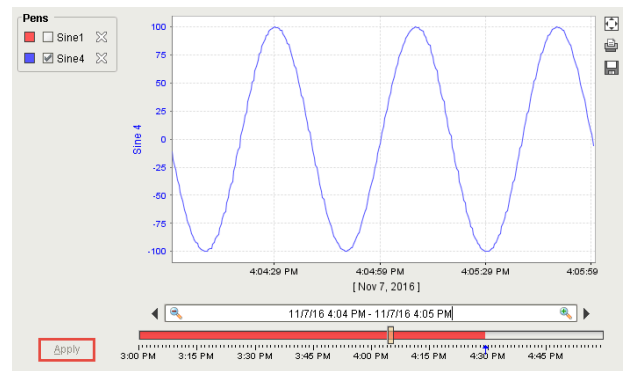
Hide Pens

You can also hide or turn off pens so they are not displayed on the Easy Chart. To only see values for the Sine 1 axis, go to [Preview Mode](#), uncheck the **Sine 4** pen, and click **Apply**. To see the values for only the Sine 4 axis, check the **Sine 4** pen, uncheck the **Sine 1** pen, and click **Apply**. Also important to note, auto positioning on the Easy Chart will automatically move the axis should pens assigned to an axis be removed.

Sine 1 Pen



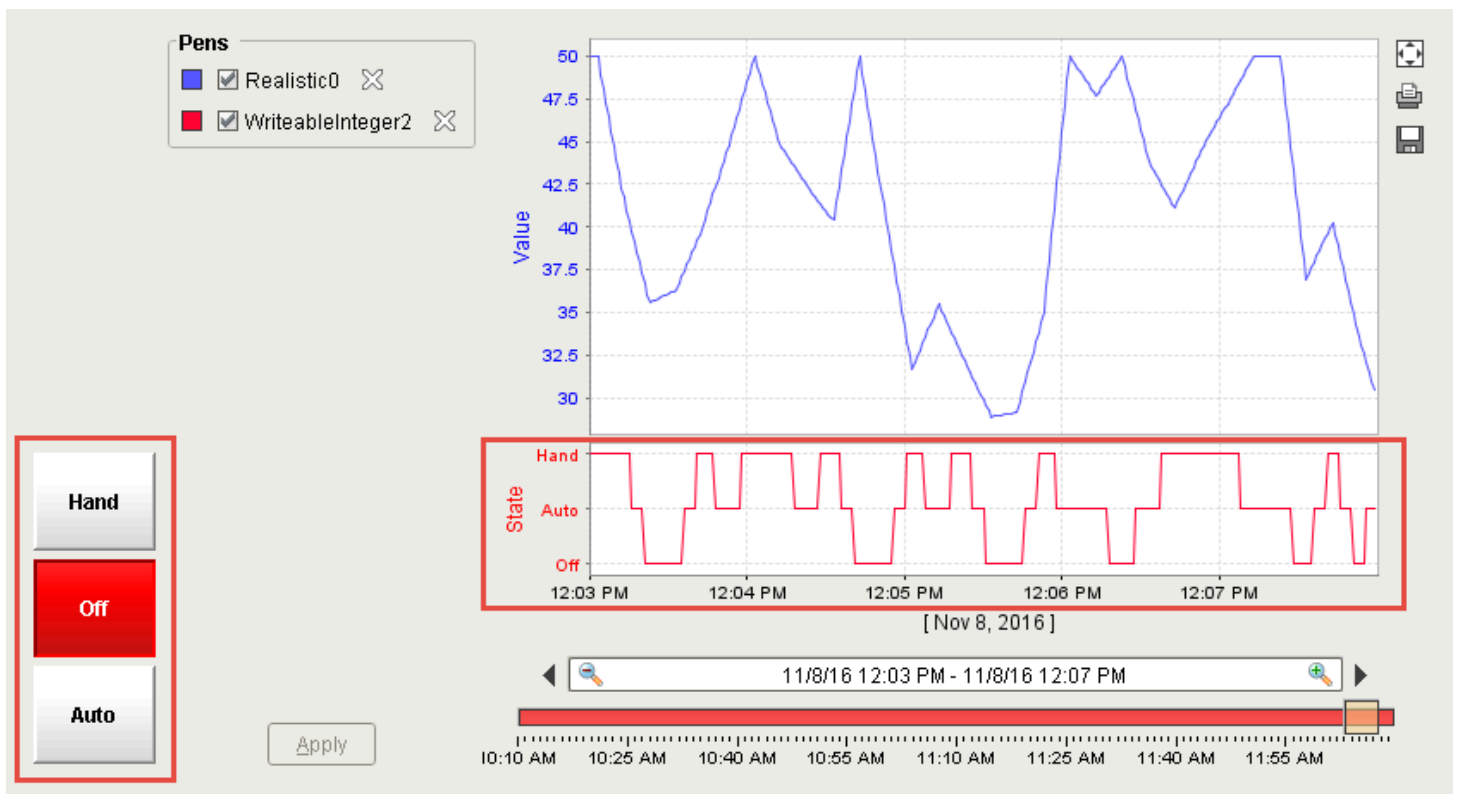
Sine 4 Pen



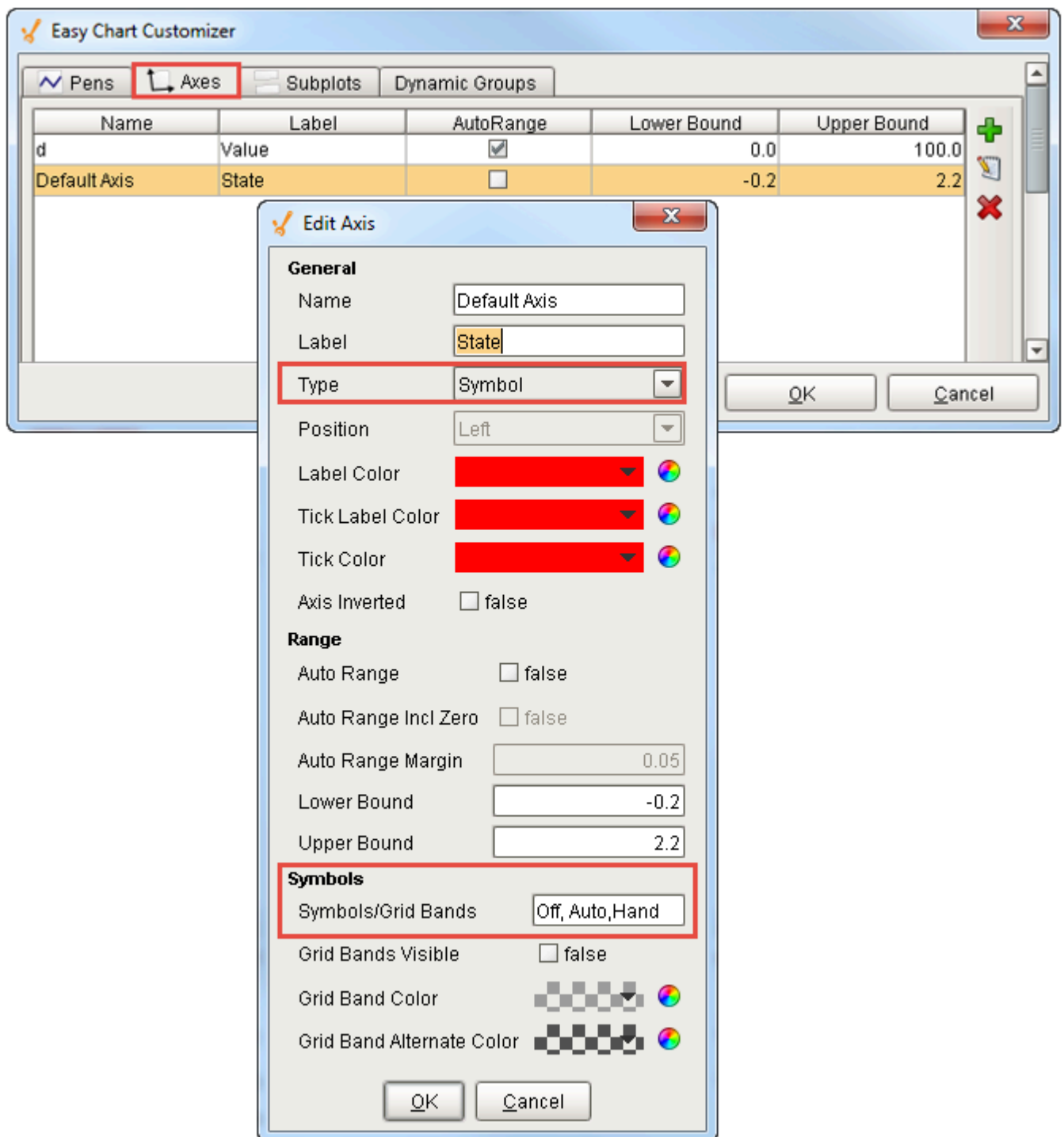
Configuring an Easy Chart using the Symbol Axis

Another feature of the Easy Chart is the use of the **Symbol Axis** type. Instead of showing numerical values on the axis, the Symbol Axis type can show plain text on the axis.

The second subplot in this Easy Chart uses a Multi-State Button component to demonstrate the use of the **Symbol Axis** type. The Multi-State Button component is bound to an OPC Tag, and the value of the Tag is stored in the Tag History system. Instead of showing the numerical values '0,' '1,' and '2,' you can use plain text such as 'Hand,' 'Off,' and 'Auto.' This is helpful to an operator who immediately knows the state of the equipment instead of having to learn what the numeric values mean.



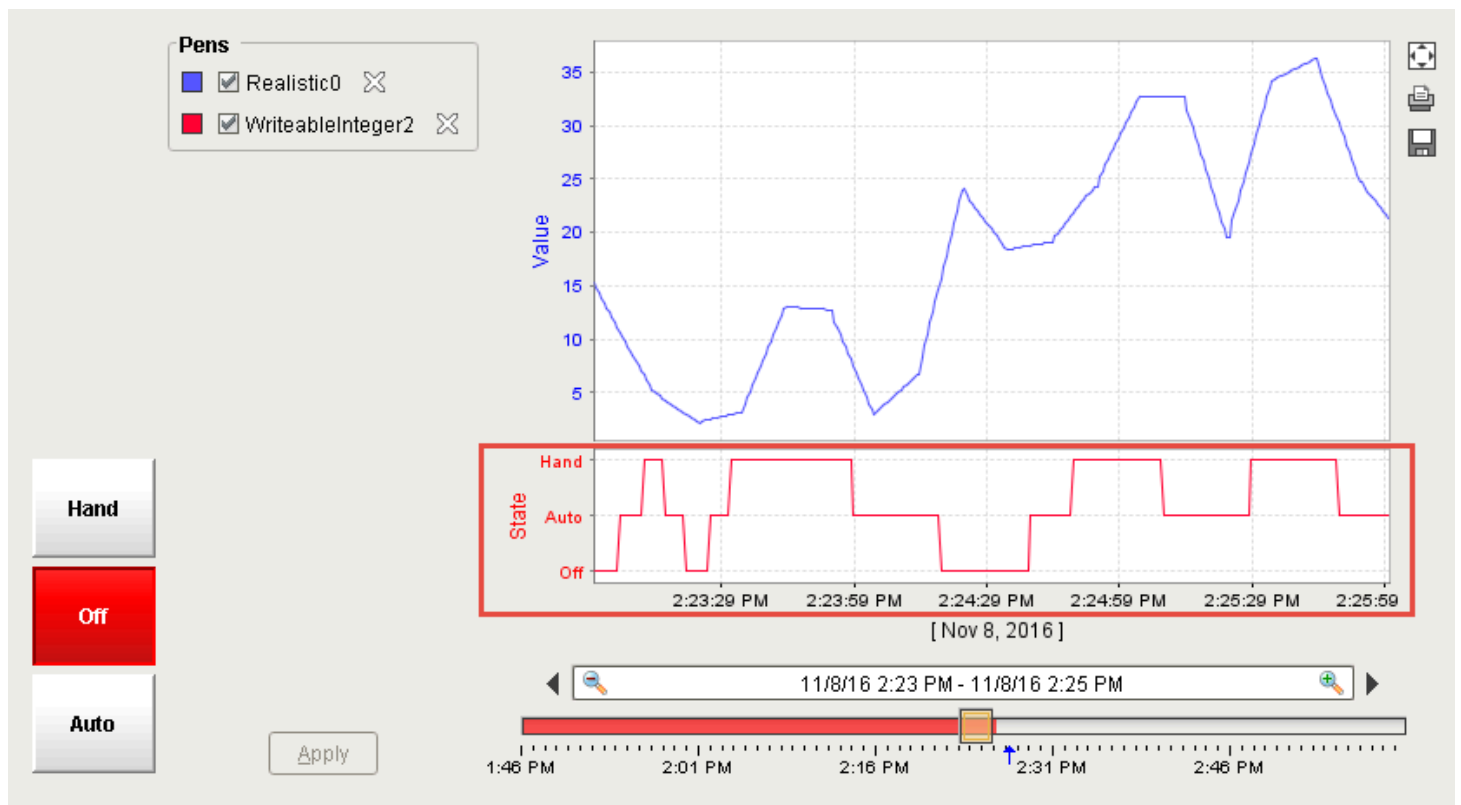
1. Click on your Easy Chart component, and select **Customizers > Easy Chart Customizer**.
2. To use the Symbol Axis type, click on the **Axes** tab, and the **Edit Axis** window will be displayed.
3. In the Type field, select '**Symbol**' from the dropdown, and enter your text in the **Symbols/Grid Bands** field for what you want the operator to see for the 'State'. 'Off,' 'Hand,' and 'Auto' were entered separated by commas, and no spaces.



NOTE

The order of the symbols when you type them in will be ascending order on the axis

- Click **OK** two times.
- In **Preview Mode**, begin toggling the Multi-State buttons to 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 above 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.

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

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

Easy Chart - Subplots



INDUCTIVE
UNIVERSITY

Easy Chart - Subplots

[Watch the video](#)

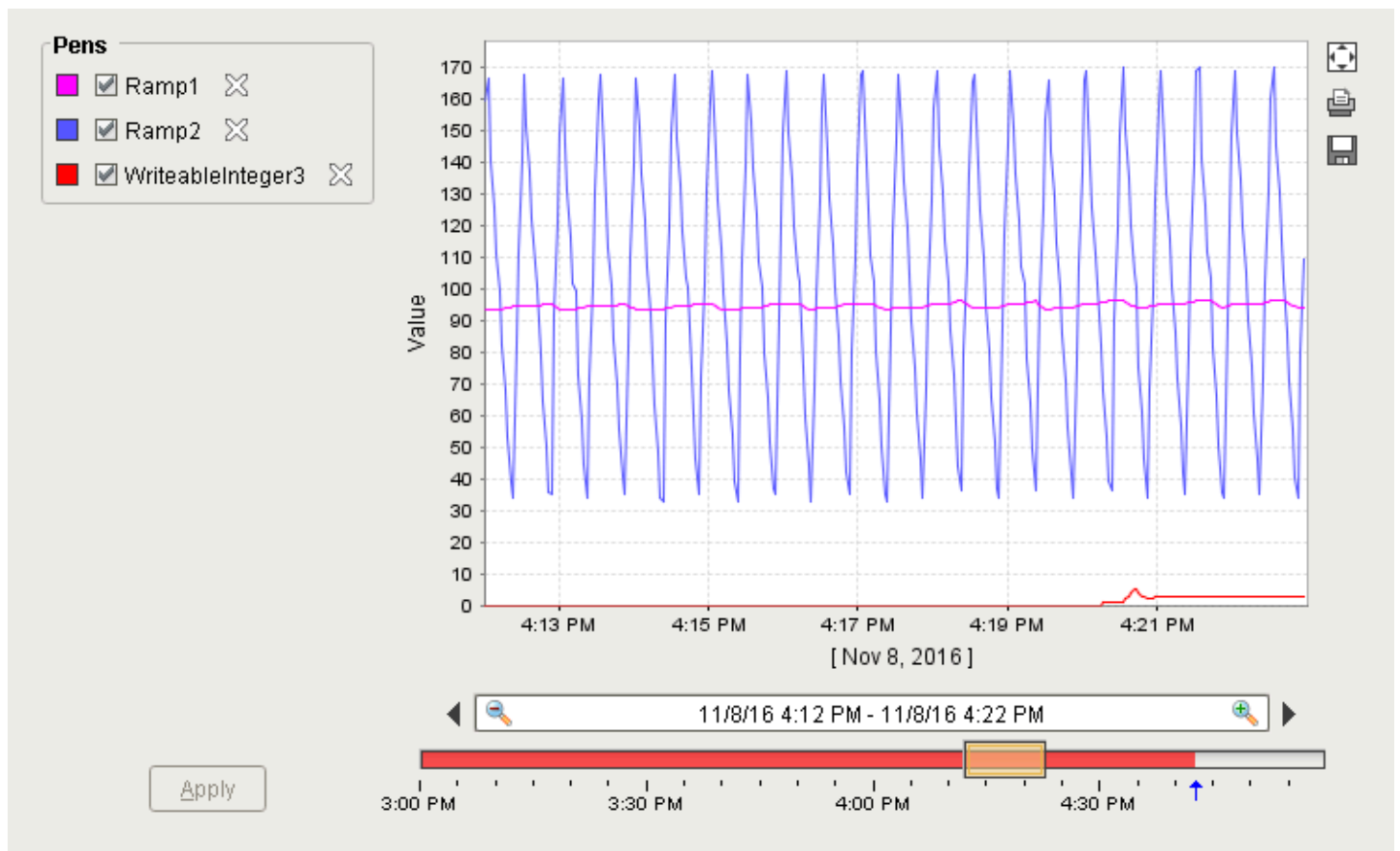
ⓘ THIS SECTION ASSUMES THAT TAGS AND TAG HISTORY HAVE BEEN CONFIGURED

To learn more, go to the [Tag](#) and [Set up Tags to Log](#) pages. The examples below use [OPC tags](#) from the [Generic Simulator](#) driver, but [Memory, Expression, and Query Tags](#) can be used instead.

Subplot Overview

The subplot feature of the Easy Chart component allows you to break up the chart plot area into multiple distinct subplots sharing the 'X' axis, but they each have their own 'Y' axis. It is a good way to display lots of data from different Tags in one Easy Chart.

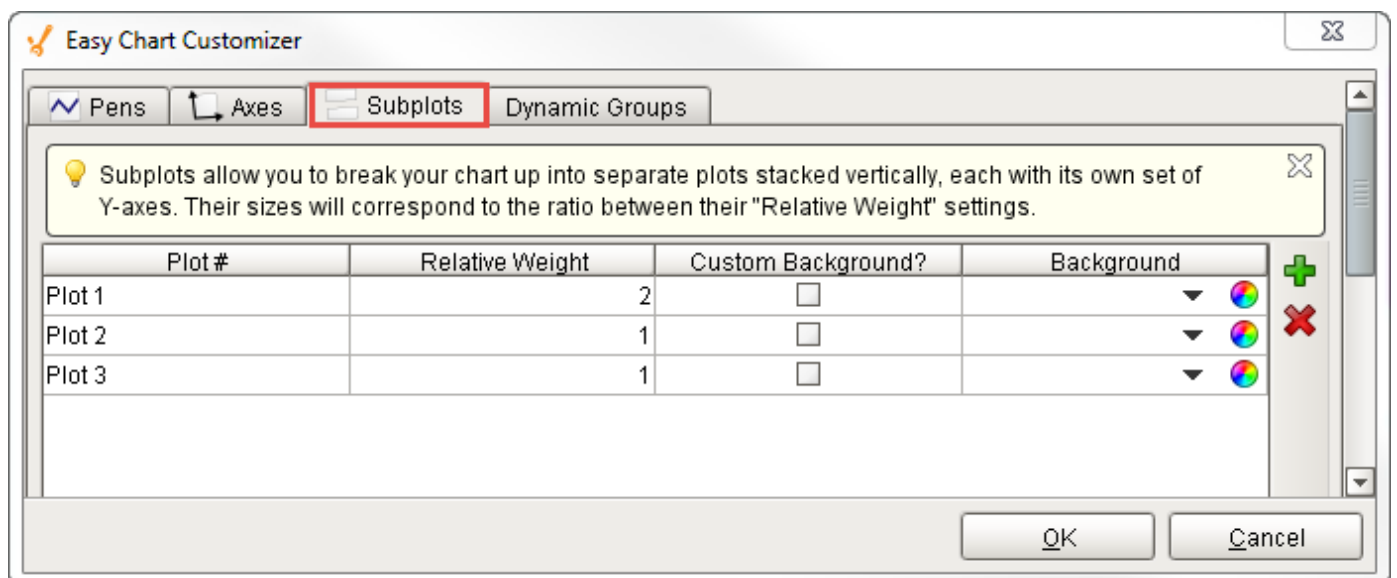
By default, the Easy Chart has one subplot which is the main white area. In this example, there are three Tags inside the chart, yet it's difficult to see the details of the data. It's possible to break up your Tags into multiple subplots which is often useful for discrete data.



Configuring Easy Chart Subplots

For each Tag in the Easy Chart example above, let's create its own subplot so the data is easier to view and analyze.

1. Right click on the Easy Chart component, and scroll down to **Customizers > Easy Chart Customizer**. The **Easy Chart Customizer** window will open displaying three Tags. Click on the **Subplots** tab.
2. The Subplots tab lets you add one or more subplots to the Easy Chart. Create two more subplots by clicking the **green plus icon**  two more times.

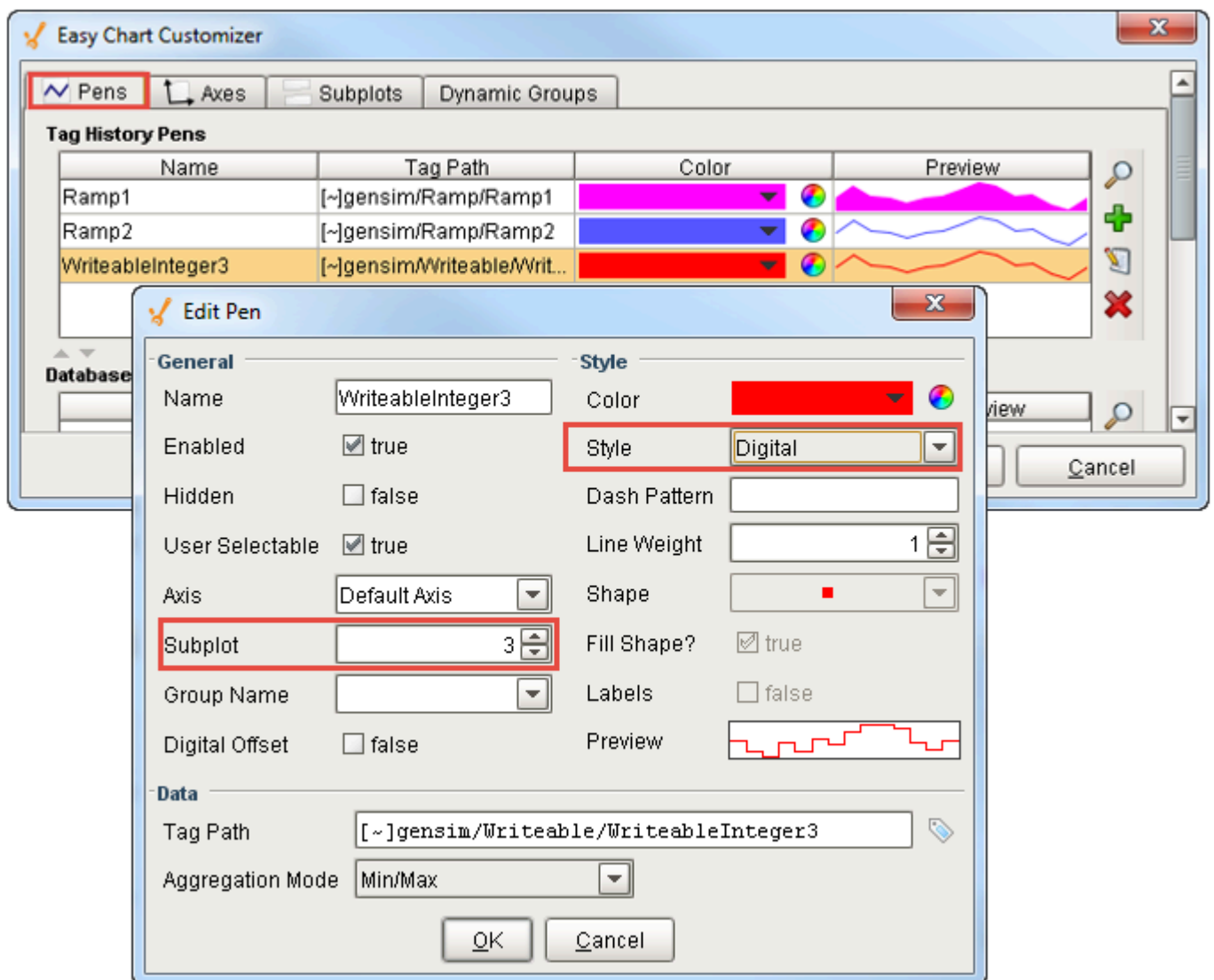


The size of each subplot corresponds to the ratio between their "Relative Weight" settings. By default, each subplot is assigned a Weight of 1, meaning each subplot will share an equal percentage of space on the chart. In this example, Subplot 1 has a weight of 2, and subplots 2 and 3 have a weight of 1. Subplot 1 is going to be 2 times larger than Subplots 2 and 3.

3. Once you add the subplots, put each of the different pens into a different subplot. Click on the **Pens** tab, select the **Pen row**, and click the **Edit Pen** icon . In the Subplot field, use the dropdown to select the subplot number.

If you would like to change the style of how the data is presented in the subplot, click the dropdown list next to **Style**, and choose from the several style options available. Click **OK**.

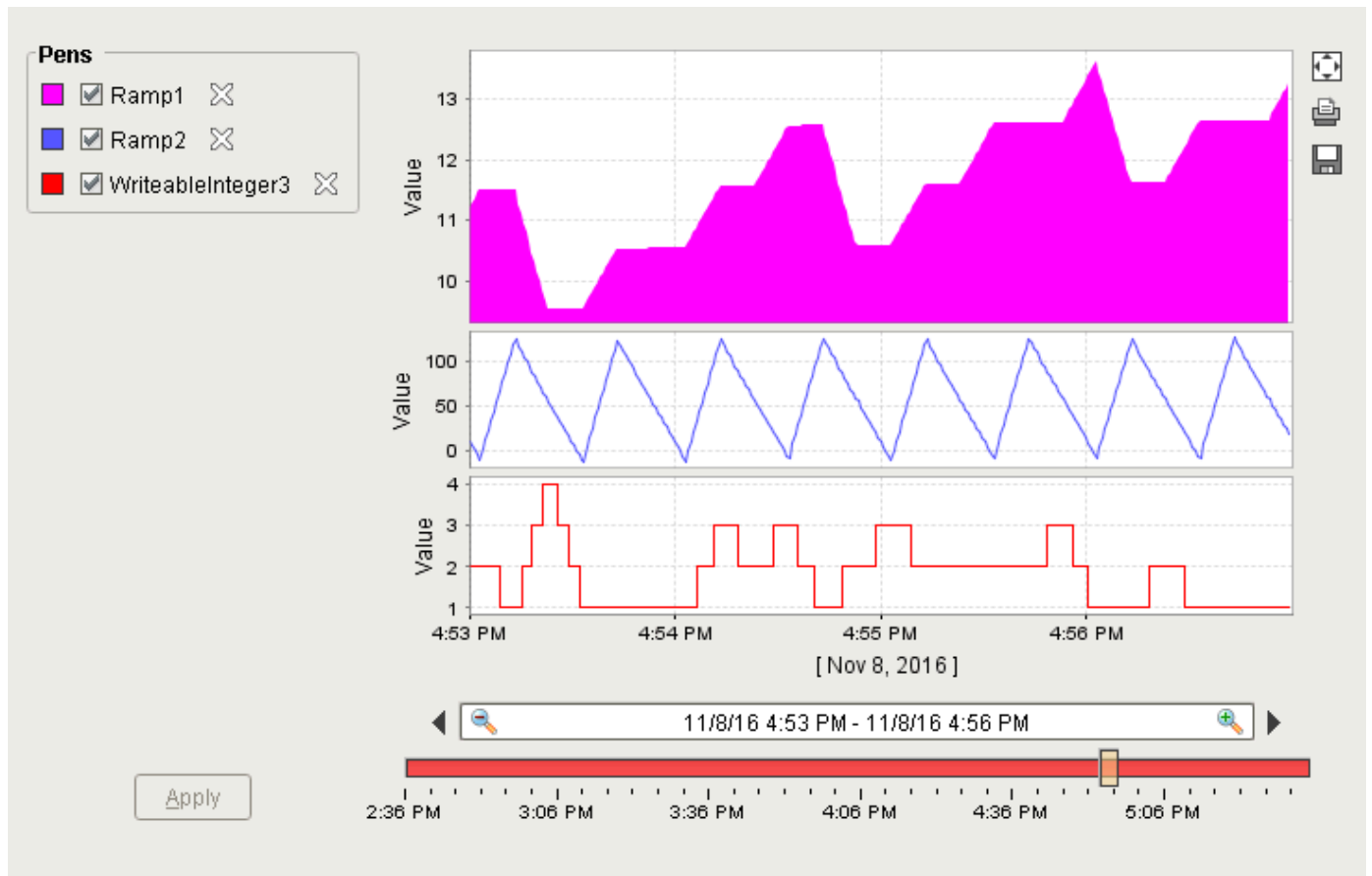
Repeat this step for each one of your pens, then press **OK**.



CUSTOMIZING A PEN

To learn more about customizing a pen to change its style, shape, weight, and color, refer to [Easy Chart - Pen Renderer](#).

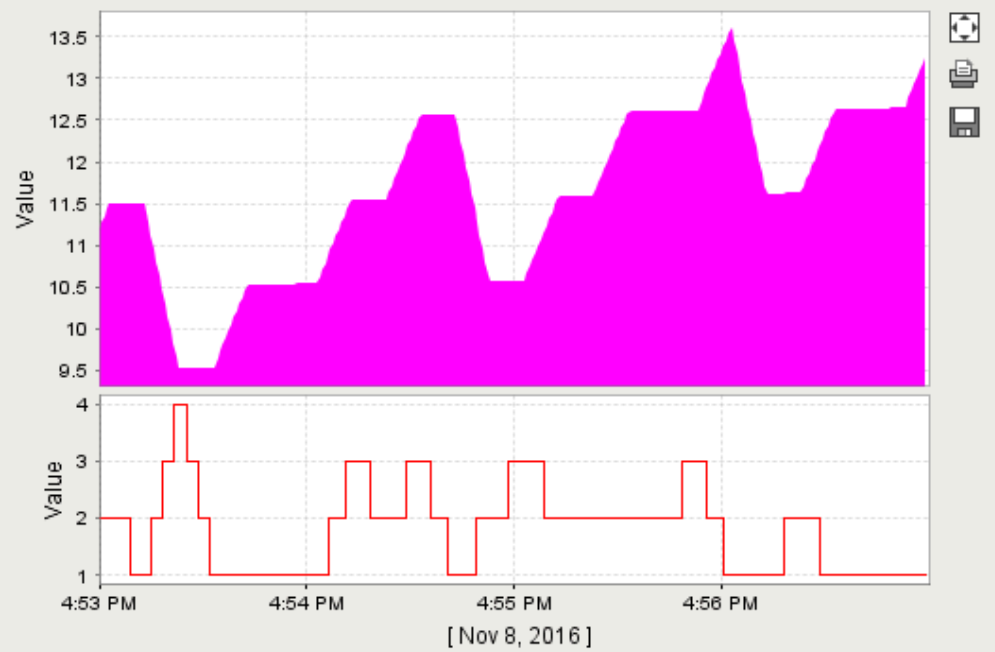
4. Now, you have three distinct subplots on one Easy Chart. There is no limit to the number of subplots on one Easy Chart.



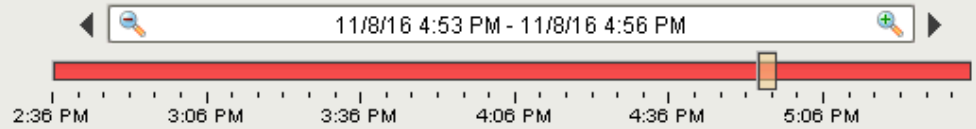
5. You can be selective about what subplots you want to view. Go to **Preview Mode**, uncheck the Pens you don't want to see, and click **Apply**. Notice how the Ramp 2 Pen is unchecked and is no longer displayed on the Easy Chart. To add the Ramp 2 pen back to the Easy Chart, check the Ramp 2 pen, and press **Apply**. The Easy Chart only shows subplots that have active Tags.

Pens

- ☒ Ramp1 ✕
- ☐ Ramp2 ✕
- ☒ WriteableInteger3 ✕



Apply



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

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

Easy Chart - Pen Names and Groups



INDUCTIVE
UNIVERSITY

Easy Chart - Pen Names and Groups

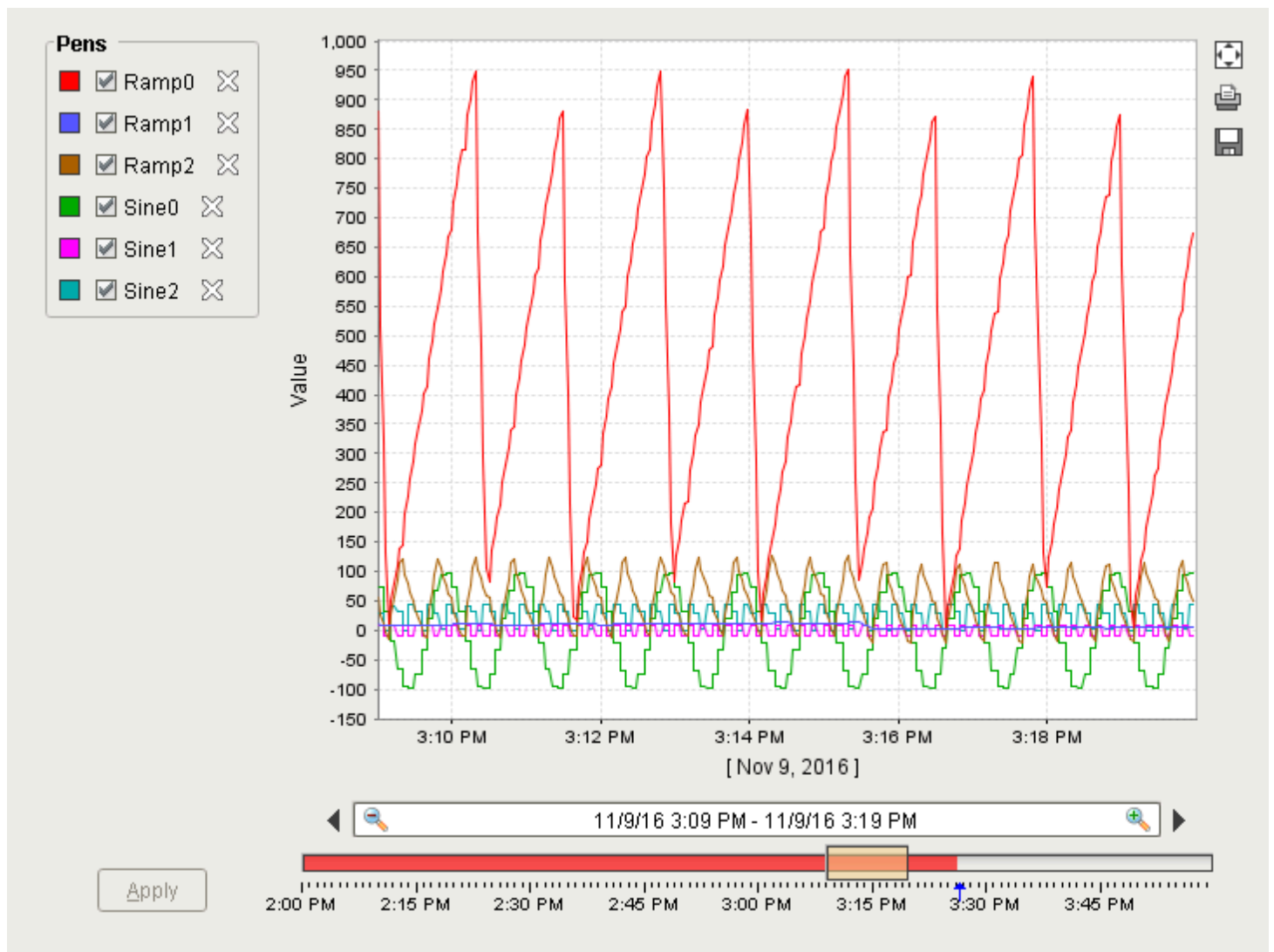
[Watch the video](#)

ⓘ THIS SECTION ASSUMES THAT TAGS AND TAG HISTORY HAVE BEEN CONFIGURED

To learn more, go to the [Tag](#) and [Set up Tags to Log](#) pages. The examples below use [OPC tags](#) from the [Generic Simulator](#) driver, but [Memory, Expression, and Query Tags](#) can be used instead.

Pen Names and Group Overview

You can organize pens on the Easy Chart by creating 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 the Tag name and organized into a single group called '**Pens.**' One of the great things about pens is you can change pen names and organize pens into different groups making it easier for the operator to quickly analyze the data.



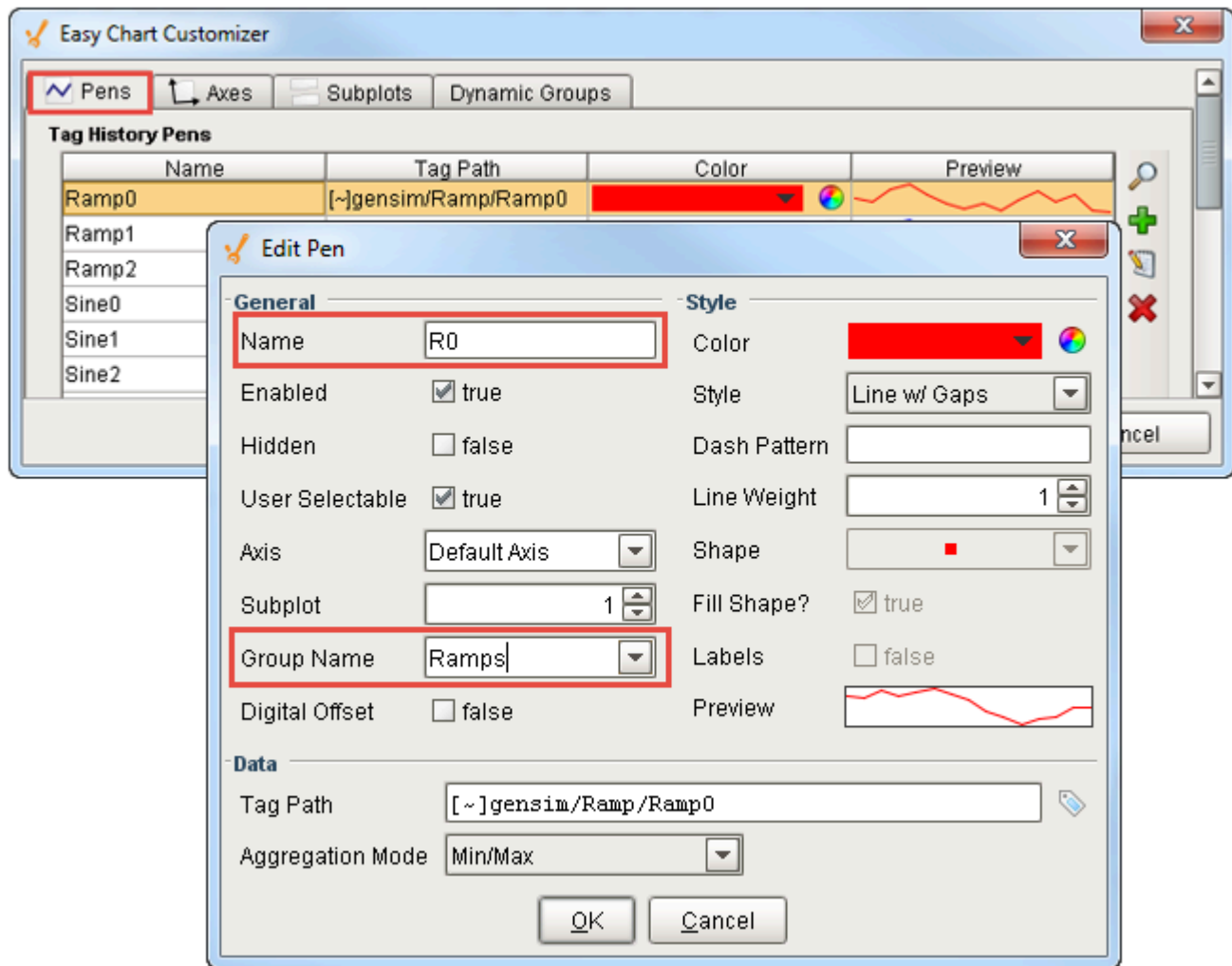
Configuring Pen Names and Groups

For each Tag on the Easy Chart, let's create unique pen names and organize each pen into a group using the [Easy Chart Customizer](#).

1. Right click on the Easy Chart component, and scroll down to **Customizers > Easy Chart Customizer**.
2. On the **Pens** tab, edit each of the individual pens and give them a different name as well as put them inside a different group.

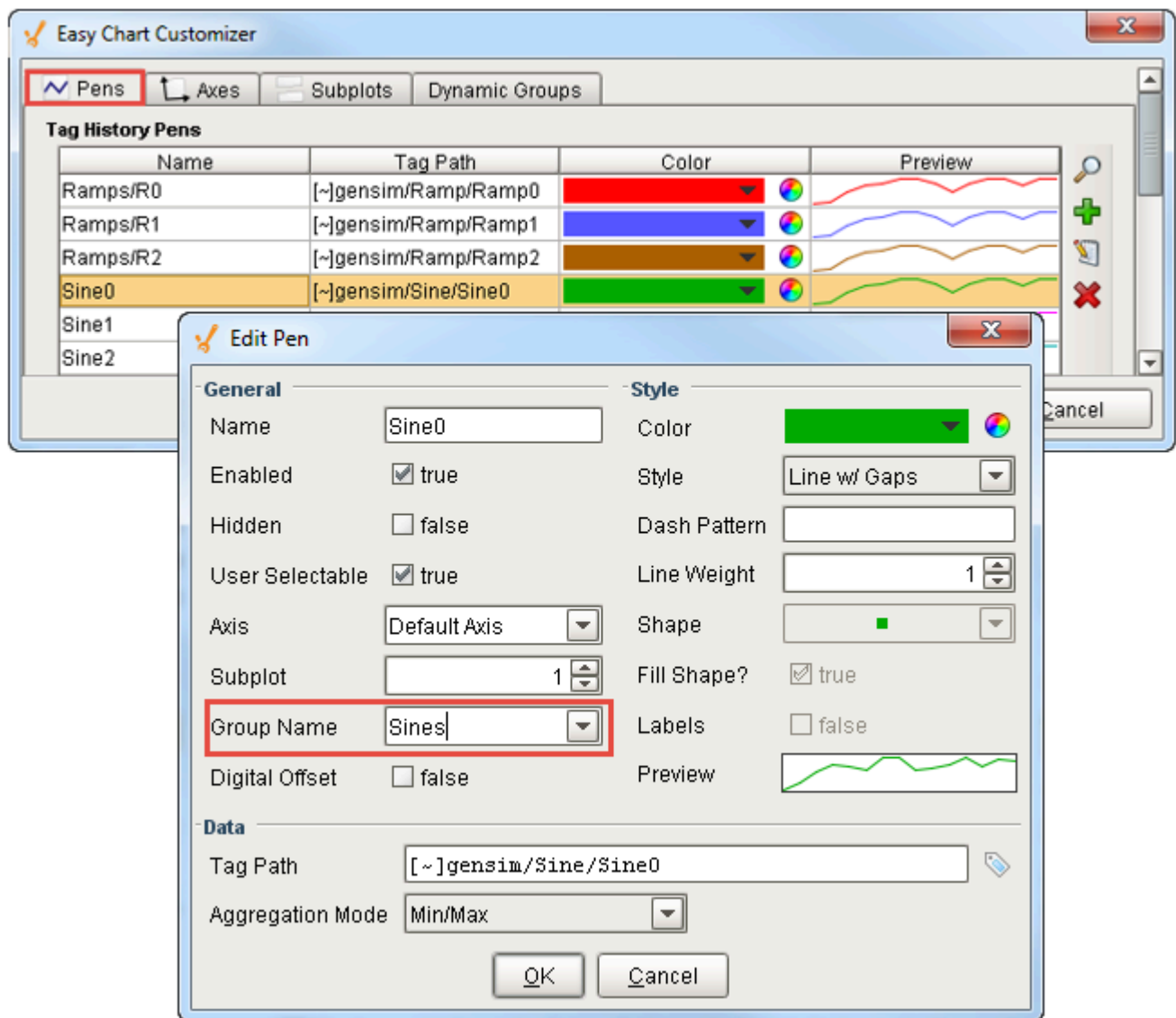
Select the first pen row, click the **Edit** icon, and rename the first Tag from '**Ramp0**' to '**R0**' as shown in the following example. At the bottom of the Edit Pen window, next to **Group Name**, create a new group called '**Ramps**.' If other groups exist, you can select one from the dropdown list or enter your own. In this example, this is the first group to be created, so simply type '**Ramps**,' and press **OK**.

Repeat this step for each Ramp pen assigning each pen a new name.



3. Let's keep the Sine pen names the same as the Tag name, but add them to a group. Select the **Sine0** pen row, and click the **Edit** icon. Create a new group name called '**Sines.**'

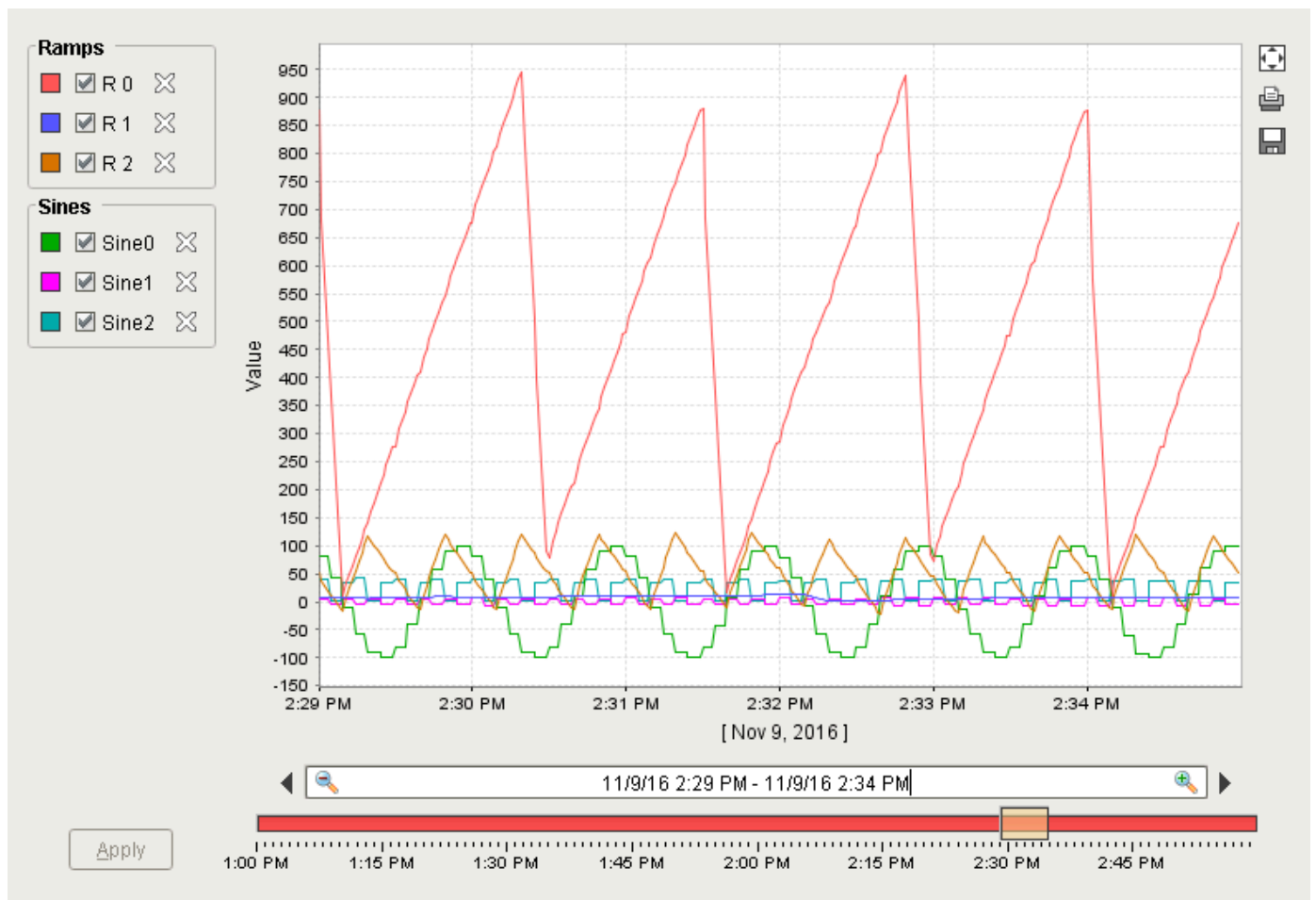
Repeat this step for each Sine pen assigning each pen to the Sines group. Click **OK**.



QUICK TIP ABOUT EDITING PENS

Once you customized the Pen and the Group name for each of the pens, you can see all your updates in one view under the Pens tab. You can update the pen and group names on this view by double clicking on any of the fields. It's a faster way to edit this information then having to go to the Edit icon for each pen.

- Once you have all your pens configured, click **OK**. Now the operators will see the Pen and Group names well organized on the Easy Chart.



TIP FOR VIEWING SPECIFIC PEN VALUES

To see specific Pen values on the Easy Chart, uncheck the Pen and click the Apply button.
To see all Pen values, make sure all Pens are checked, and hit the Apply button.

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

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

Easy Chart - Pen Renderer



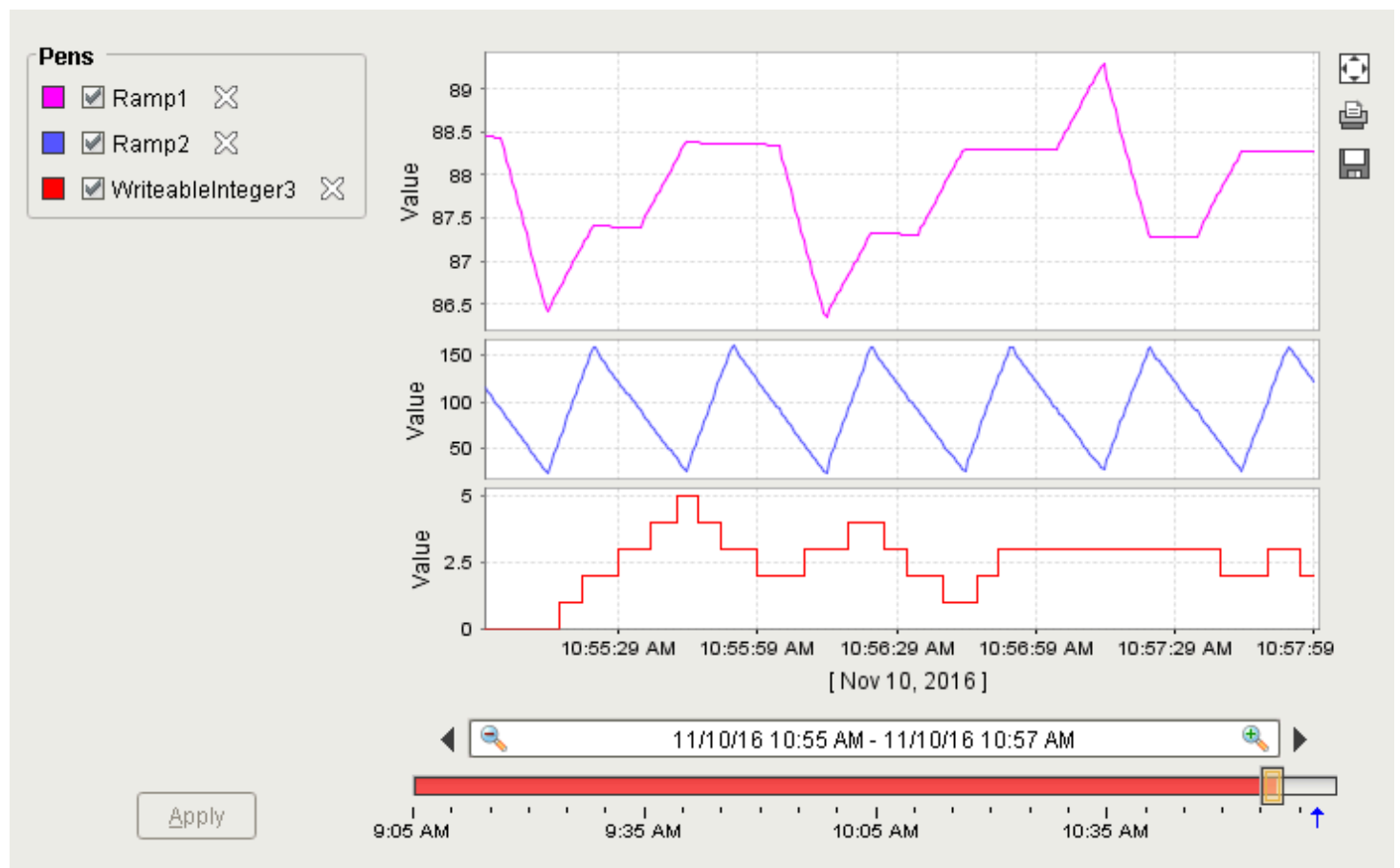
INDUCTIVE
UNIVERSITY

Easy Chart - Pen Renderer

[Watch the video](#)

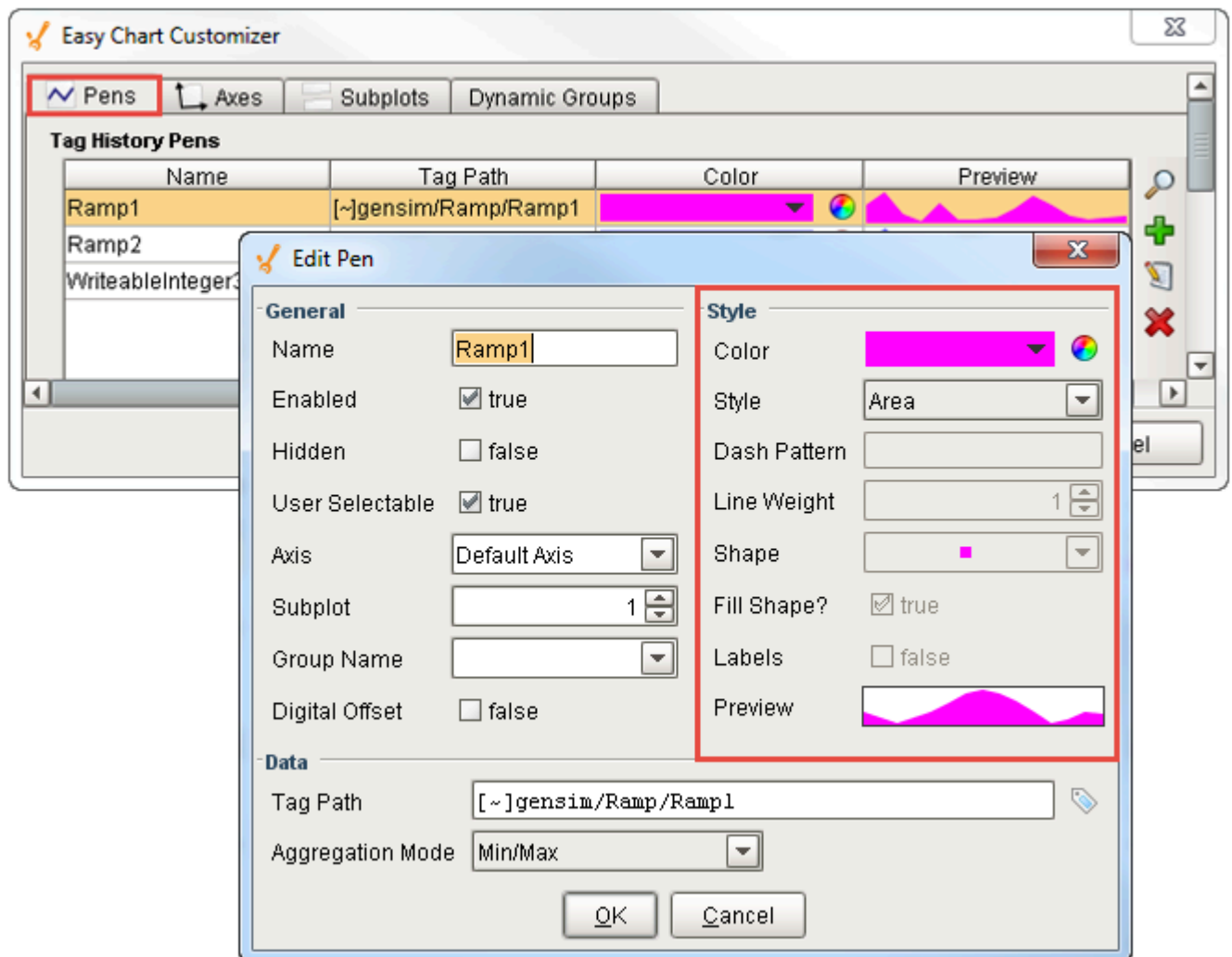
Customizing the Pen Renderer

You can customize the renderer of each pen on the Easy Chart to change its style, shape, weight and color. When adding Tags to an Easy Chart component, the default renderer or style of each pen is a simple line, but it can be customized for each pen.



1. To customize the pen renderer, right click on the Easy Chart component, and scroll down to **Customizers > Easy Chart Customizer**.

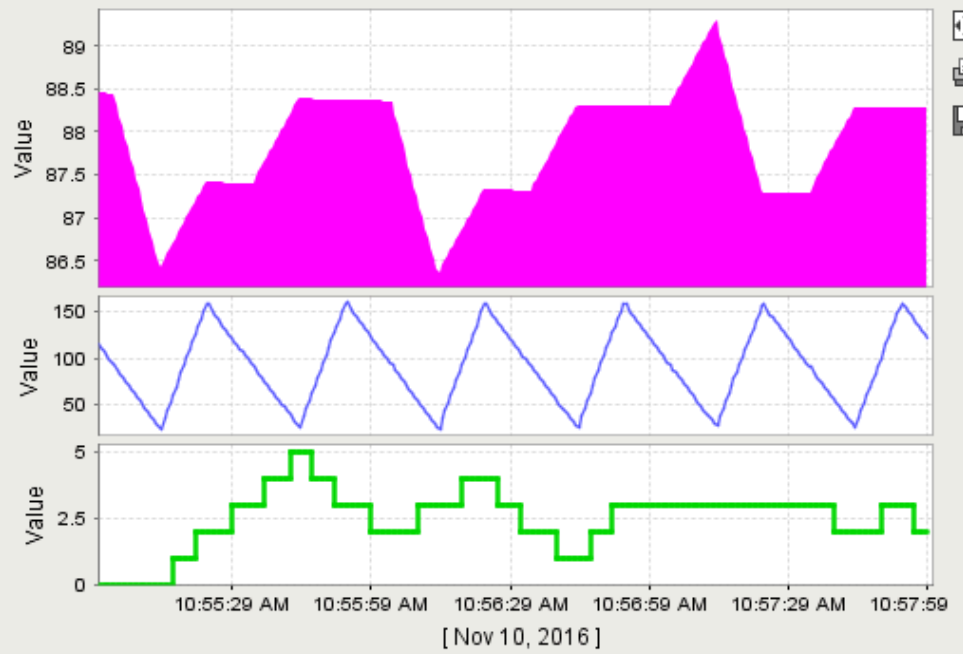
- From the **Pens** tab, select the pen row, and click the **Edit** icon. The Edit Pen window will open. On the right side of the Edit Pen window under Style, you can change the line color, style, weight, and shape. The line style, by default, is '**Line with Gaps**,' which means that if you lose communication 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.
- Edit each style property. If you don't like the result in the Preview window, select another style. You can edit it as many times as you like. When you're finished, press **OK**.



- Repeat steps 2 and 3 to customize the renderer for all the pens you want to change. When you're finished, press **OK**.
- Notice how all the pens on your Easy Chart now reflect your style changes, and how easy it is to customize the style for each of your pens.

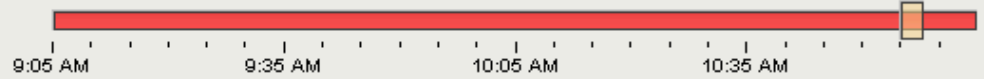
Pens

- ☒ Ramp1 ☐
- ☒ Ramp2 ☐
- ☒ WriteableInteger3 ☐



11/10/16 10:55 AM - 11/10/16 10:57 AM

Apply



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

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

Easy Chart - Digital Offset



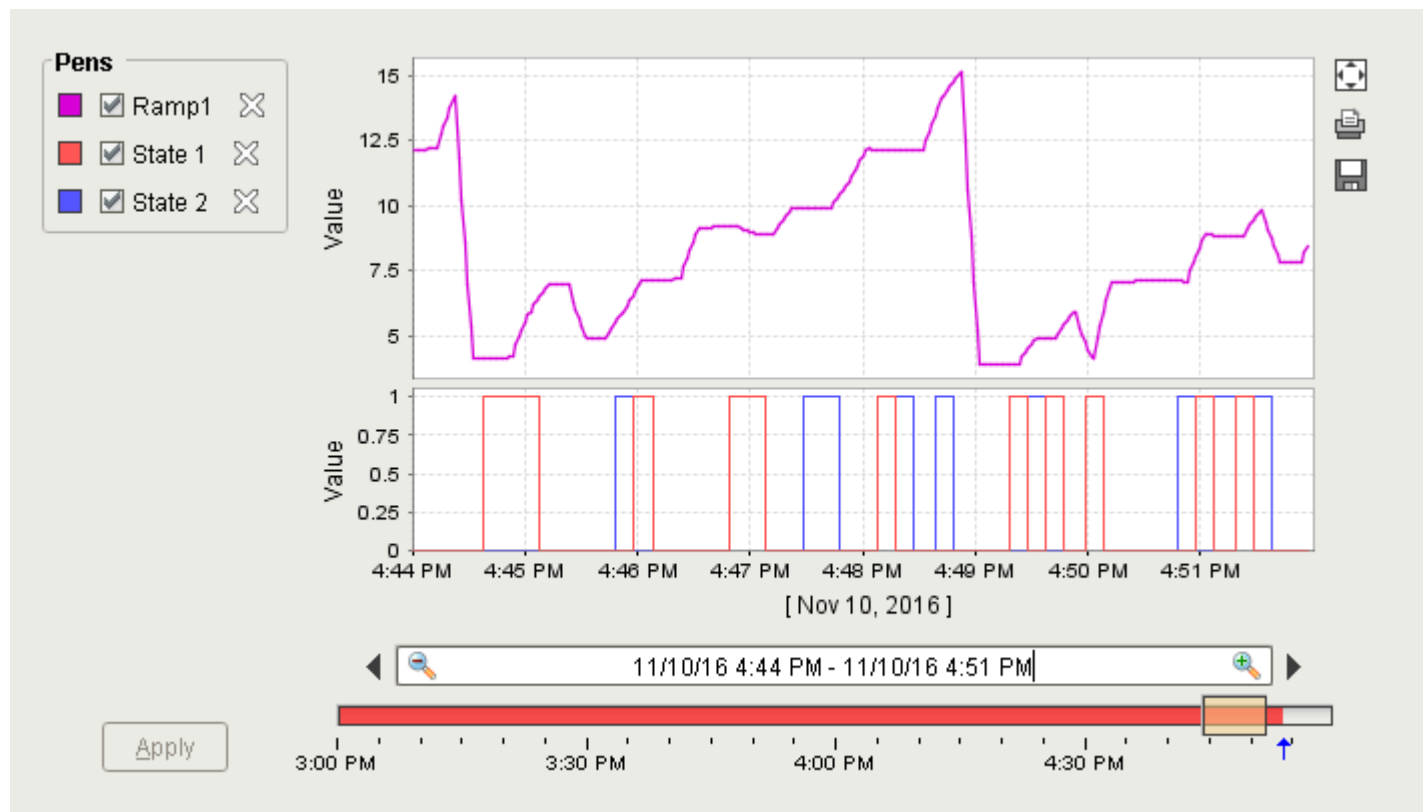
INDUCTIVE
UNIVERSITY

Easy Chart - Digital Offset

[Watch the video](#)

Digital pens often share the same subplot on the **Easy Chart** component. When you have multiple digital pens on the same Easy Chart subplot, it's hard to see what the values are of each pen because they may overlap each other. There is a digital offset pen setting that can be set which prevents the values from overlapping and enables them to be seen better in the subplot.

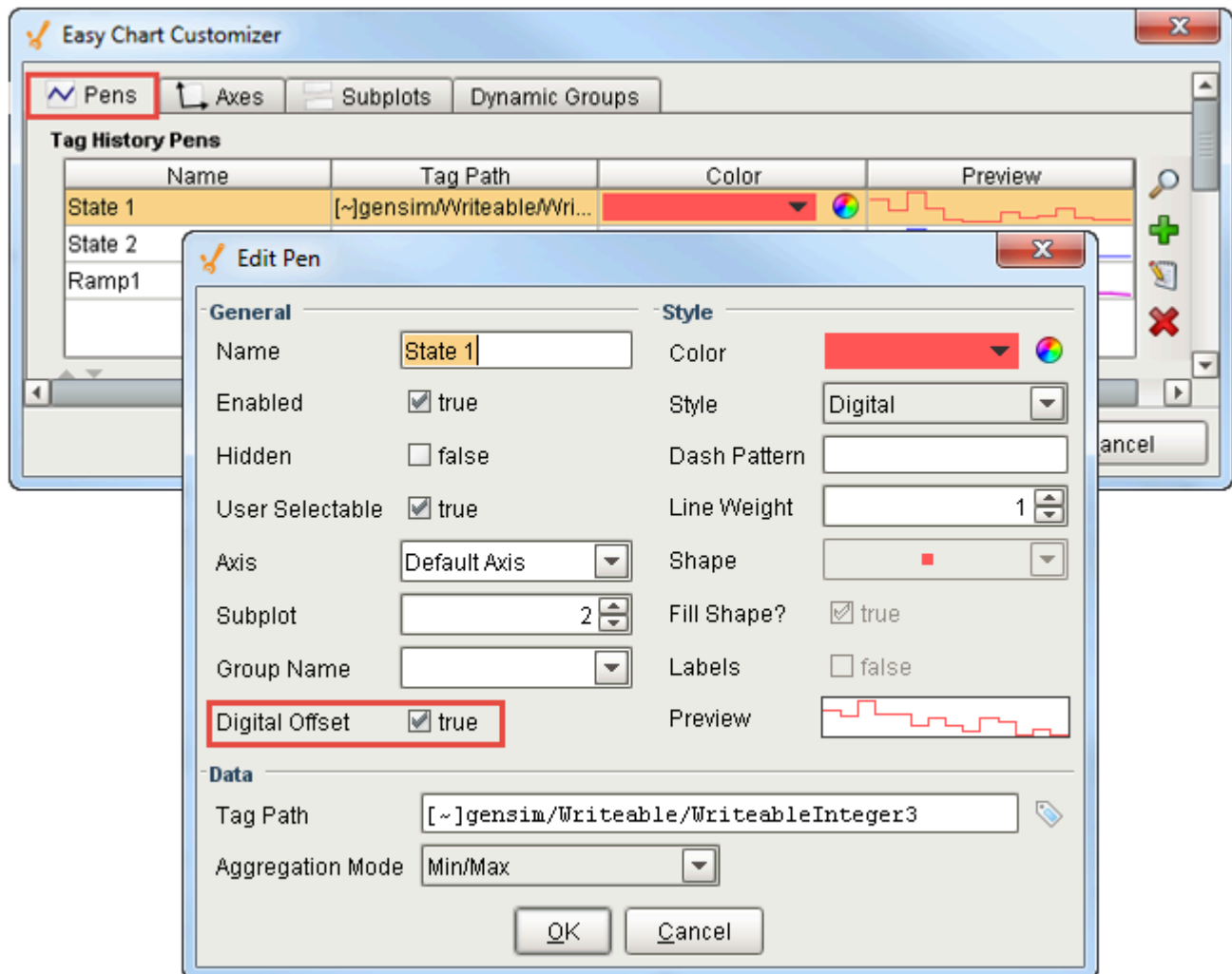
The Easy Chart below shows two digital pens on the same subplot: State 1 and State 2. The values are difficult to see because they are on top of each other.



To Add a Digital Offset

Let's apply a digital offset on the pen renderer so you can see the values better.

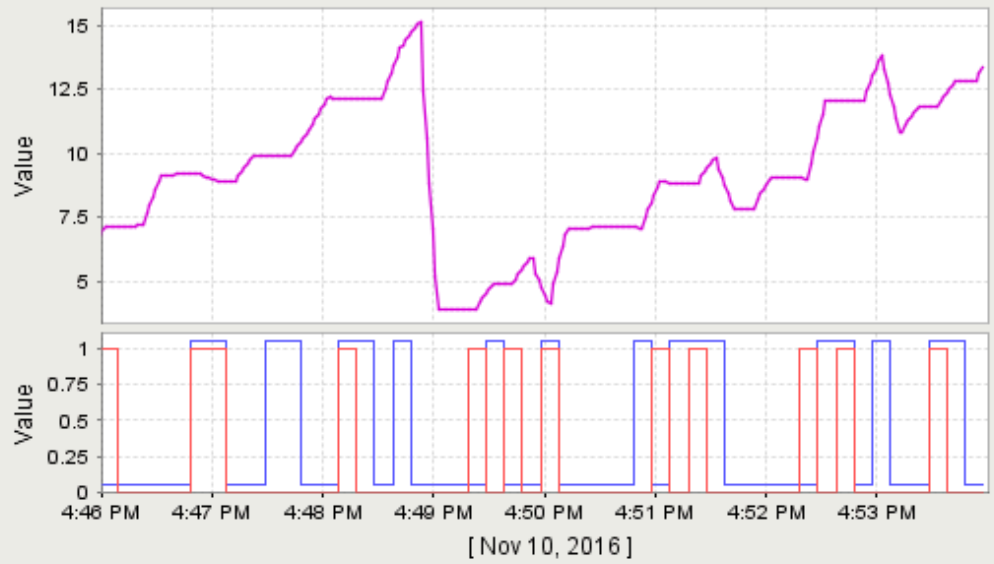
1. Right click on the Easy Chart component, and scroll down to **Customizers > Easy Chart Customizer**.
2. On the **Pens** tab, select the '**State 1**' row pen, and click the **Edit** icon. Check the **Digital Offset** box to '**true**,' and click **OK**.
3. Repeat this step for the '**State 2**' pen.



4. When you're finished editing your pens, click **OK** to return to the Easy Chart. You will see a little offset in the values between the State pens in the 2nd subplot so they don't overlap each other, making it a lot easier for the operator to read the digital values.

Pens

- ☒ Ramp1 ☐
- ☒ State 1 ☐
- ☒ State 2 ☐



11/10/16 4:46 PM - 11/10/16 4:53 PM

Apply

3:00 PM 3:30 PM 4:00 PM 4:30 PM

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

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

Easy Chart - Calculated Pens



INDUCTIVE
UNIVERSITY

Easy Chart - Calculated Pens

[Watch the video](#)

ⓘ THIS SECTION ASSUMES THAT TAGS AND TAG HISTORY HAVE BEEN CONFIGURED

To learn more, go to the [Tag](#) and [Set up Tags to Log](#) pages. The examples below use [OPC tags](#) from the [Generic Simulator](#) driver, but [Memory, Expression, and Query Tags](#) can be used instead.

Calculated pens display a value that is dynamically calculated based on another pen. This can be used to calculate certain values for a pen and graph them alongside the original pen values, allowing you to gain valuable insight into your data. There are many unique calculations that can be used, with some of them containing unique customization. Almost all of the Calculated pens require a driving pen, which is a tag or database pen that you have already set up.

ⓘ NOTE

You cannot bind the Calculated Pen values inside the Easy Chart Customizer. To bind the function values, use the [Cell Update Binding](#).

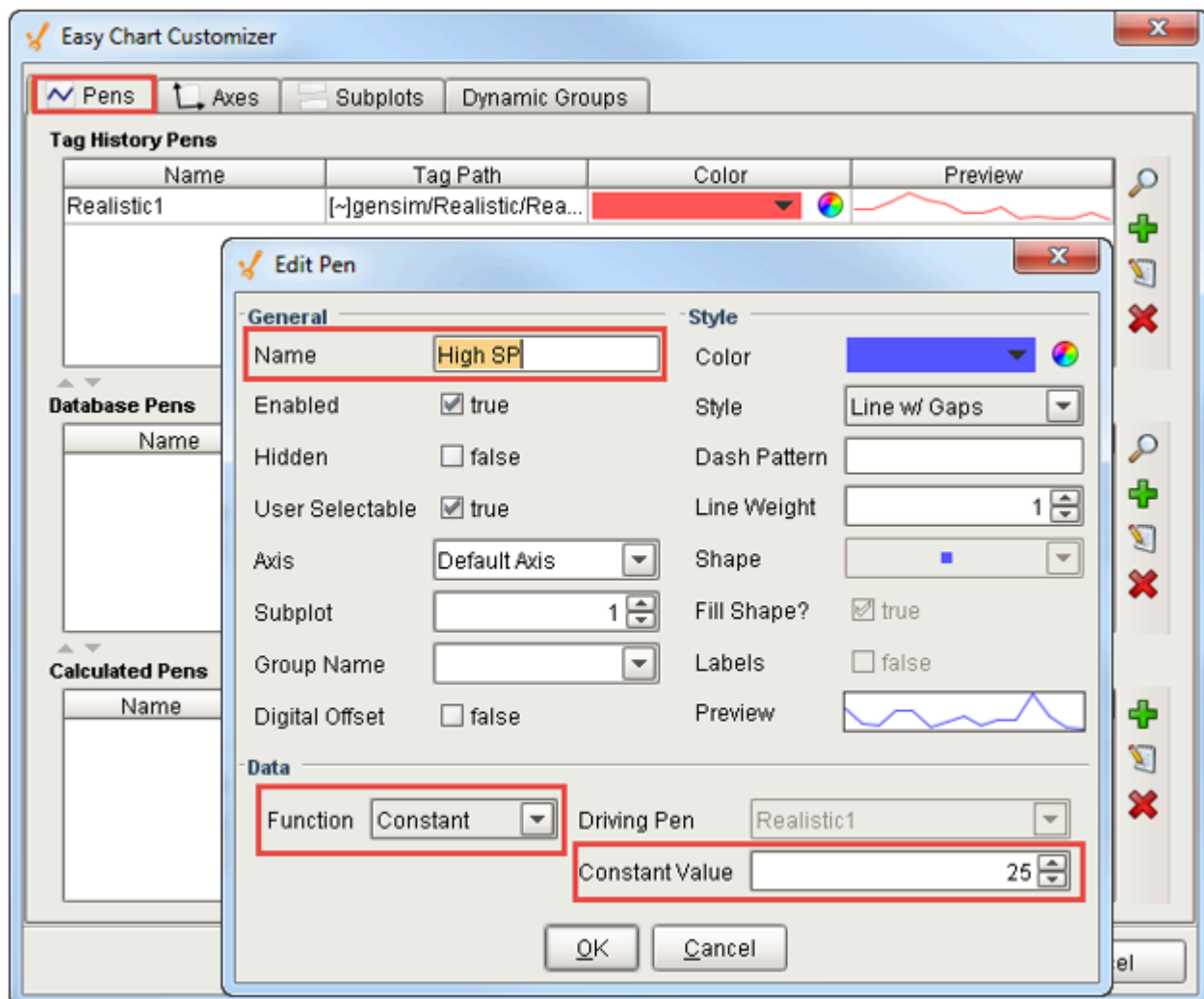
Function	Display Name	Description	Extra Properties
Constant	Constant	A constant value on the chart.	Constant Value - The constant value of the pen.
Upper Control Limit	UCL	The upper control limit of the driving pen, which is three standard deviations above the mean of the Driving Pen	
Upper Warning Limit	UWL	The upper warning limit of the driving pen, which is two standard deviations above the mean of the Driving Pen	
Average	Avg	The average of the driving pen.	
Lower Warning Limit	LWL	The lower warning limit of the driving pen, which is two standard deviations below the mean of the Driving Pen	
Lower Control Limit	LCL	The lower control limit of the driving pen, which is three standard deviations below the mean of the Driving Pen	
Moving Average	MovingAvg	A series of averages based on subsets of the driving pen. The subsets are determined by the window size.	Window Size - The size of the moving average window, specified as a multiplier of the chart's date range.

Function	Display Name	Description	Extra Properties
Multiply Pen	Multiply	Multiply each data point of the driving pen by a factor.	Factor - The factor that each data point of the driving pen is multiplied by.
Minimum Value	Min	The minimum value of the driving pen.	
Maximum Value	Max	The maximum value of the driving pen.	
Running Sum	RunningSum	A running sum or running total of the driving pen.	
Sum	Sum	The sum of two different driving pens.	Secondary Pen - The second driving pen.
Difference	Difference	The difference of two separate driving pens.	Secondary Pen - The second driving pen.
Linear Regression	LinearRegression	New in 7.9.4 Will create a linear regression line for the driving pen.	

Example - Creating Calculated Pens

1. Right click on the Easy Chart component, and scroll down to **Customizers > Easy Chart Customizer**.
2. The Pens tab will open and you'll notice a **Calculated Pens Table** at the bottom of the screen. You can add one or more calculated pens by clicking the **green plus icon**.

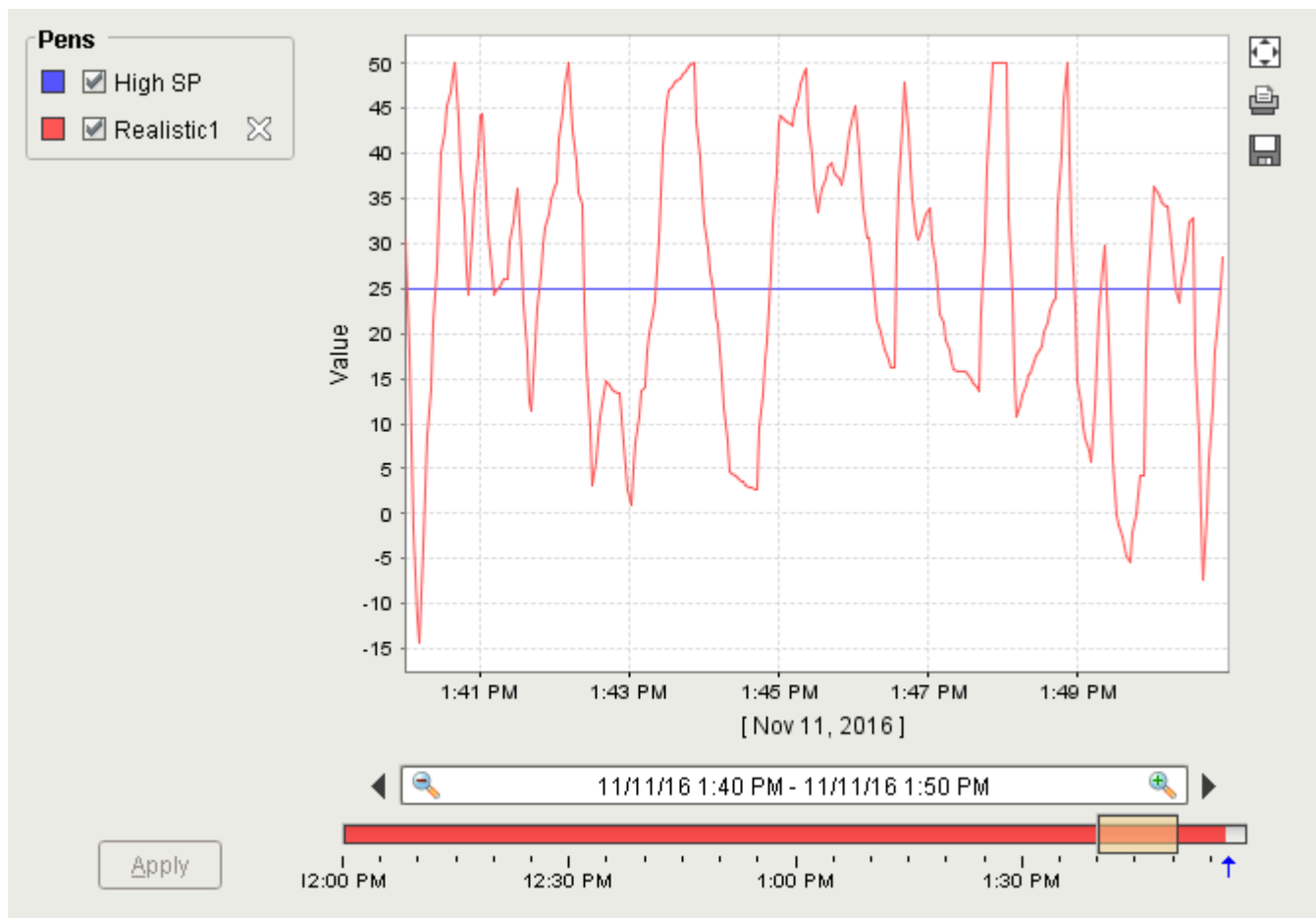
Add a calculated pen name called '**High SP**.' In the Function field, select '**Constant**' from the dropdown list, and set the 'Constant Value' at '**25**.' Click **OK**.



NOTE

Calculated pens are just like other pens, so you can specify the style, color, axis and subplot in the Edit Pen window.

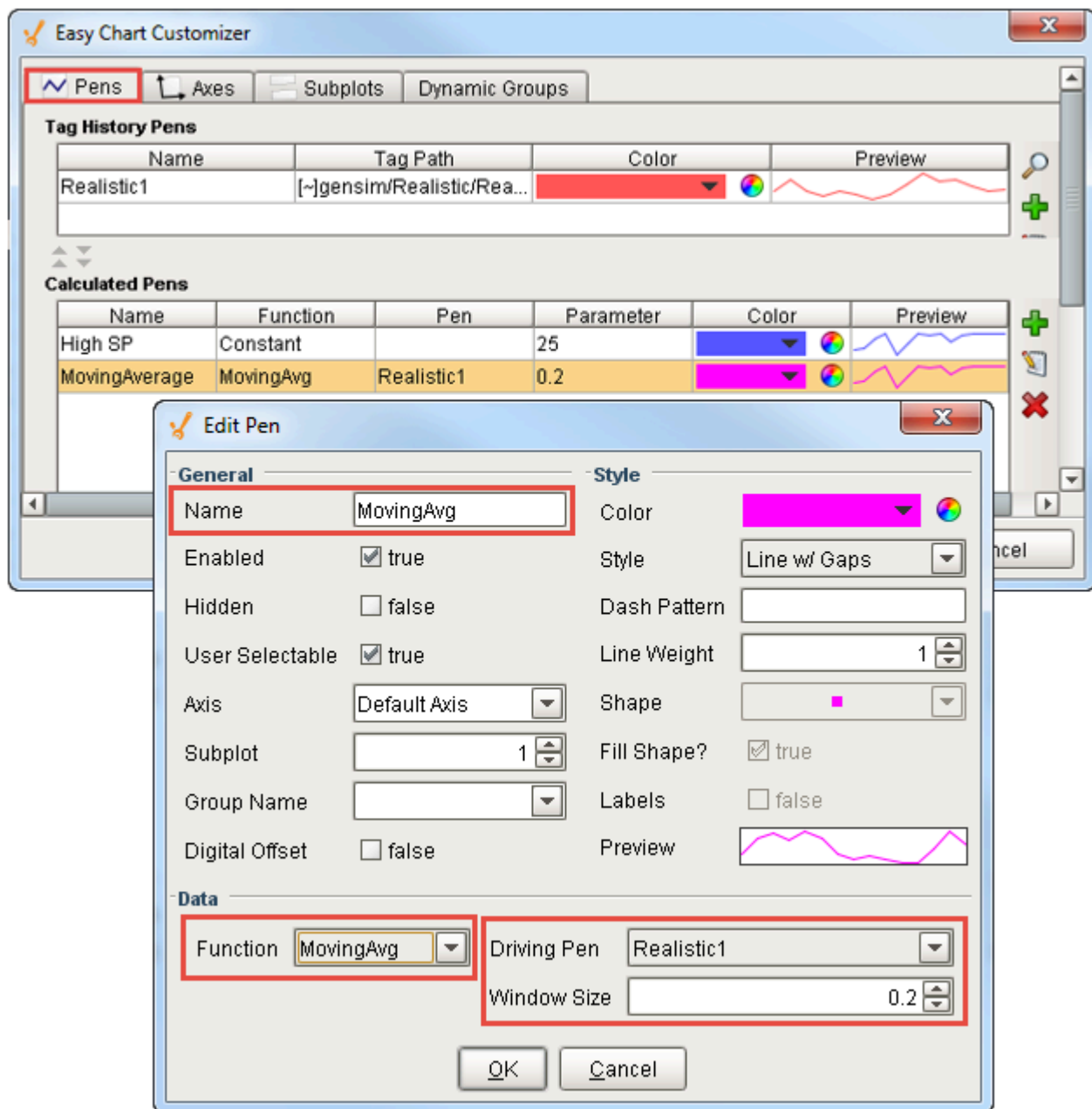
3. Press **OK** again to view your High SP pen on your Easy Chart. The High SP value of 25 is represented on your Easy Chart by a blue horizontal line.



4. Let's add a second calculated pen. On the **Pens** tab, click the **green plus icon** on the Calculated Pens Table.

On the **Edit Pen** window, enter the calculated pen name called '**Moving Average.**' In the Function field, select '**MovingAvg**' from the dropdown list. Set the Driving Pen to '**Realistic1**,' also from the dropdown list. (If you have more Tags that you dragged on to your Easy Chart from the Tag Browser, you'll have more pens to choose from in the Driving Pen dropdown list).

Specify the **Window Size** which is the percentage of the total time period that you are going to do a Moving Average on, for example **20%**. Press **OK**.

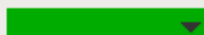
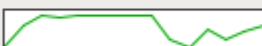


- Let's add two more calculated pens: one for Upper Control Limit (**UCL**) and another for Lower Control Limit (**LCL**), and set the Driving Property to '**Realistic1**.'

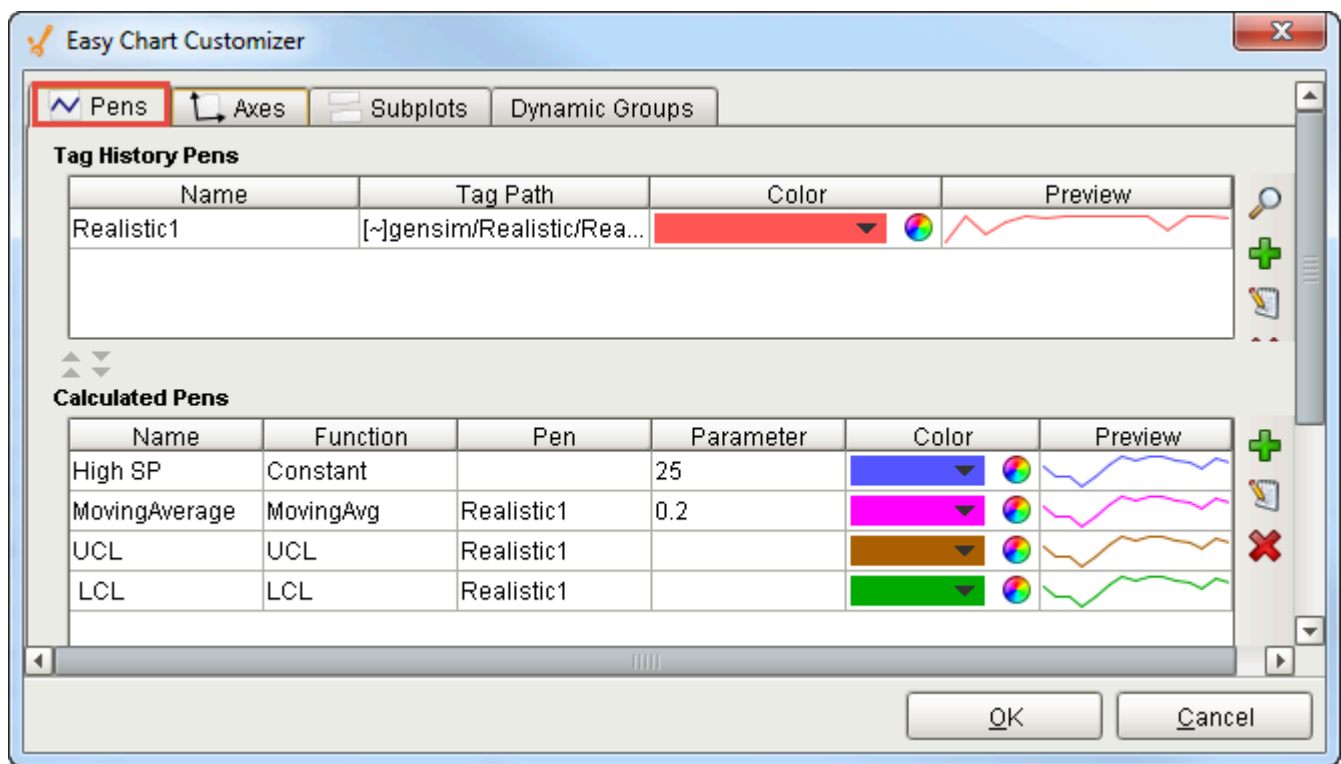
Edit Pen

General		Style	
Name	UCL	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	
Data Function: UCL Driving Pen: Realistic1			
OK Cancel			

Edit Pen

General		Style	
Name	LCL	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	
Data Function: LCL Driving Pen: Realistic1			
OK Cancel			

6. Once you added all your calculated pens, you'll see all your pens in the Calculated Pens Table at bottom of the window. Press **OK**.



7. Now, all your calculations are displayed on your Easy Chart. The blue pen is the Constant, the pink pen is the Moving Average, and UCL and LCL are brown and green respectively.



Once you have your Calculated Pens created, you'll notice that they disappear if you disable the pen driving them. If you want to remove the Driving Pen but leave the Calculated Pens, set the **Hidden** property of the Driving Pen.

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

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

Ad Hoc Charting



INDUCTIVE
UNIVERSITY

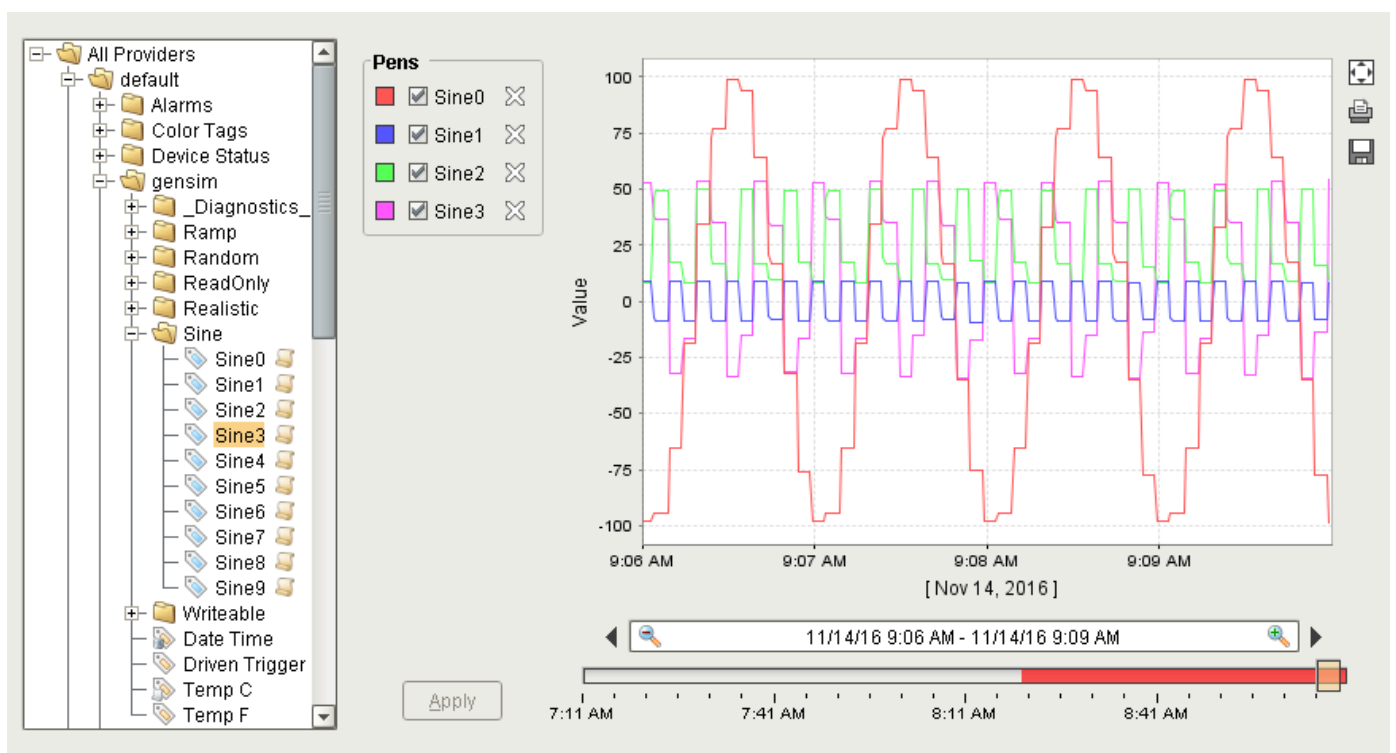
Ad Hoc Charting

[Watch the video](#)

Using Ad Hoc Charting

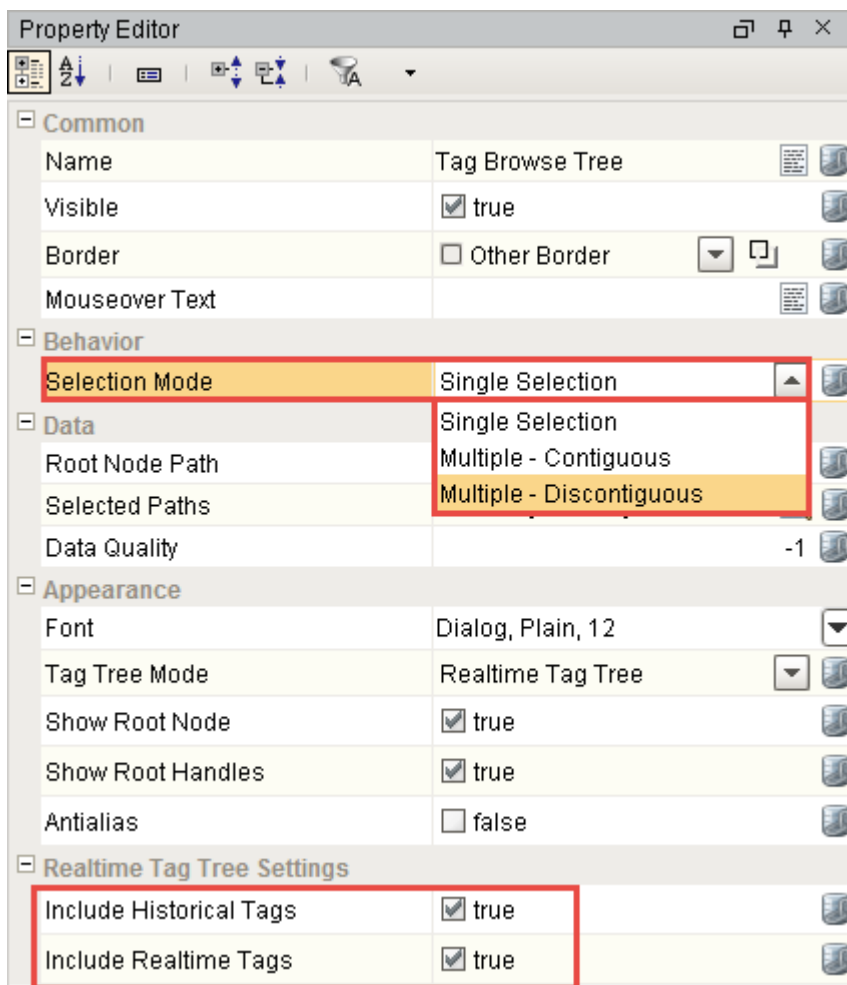
In the Designer, you can use a special built-in component called the Tag Browse Tree to create Ad Hoc Charts where you can pick and choose which pens you want to put on an Easy Chart. This same functionality is also available for an operator in the Runtime.

1. In the Designer, drag the **Tag Browse Tree** and **Easy Chart** components from the component palette to your workspace. In **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.



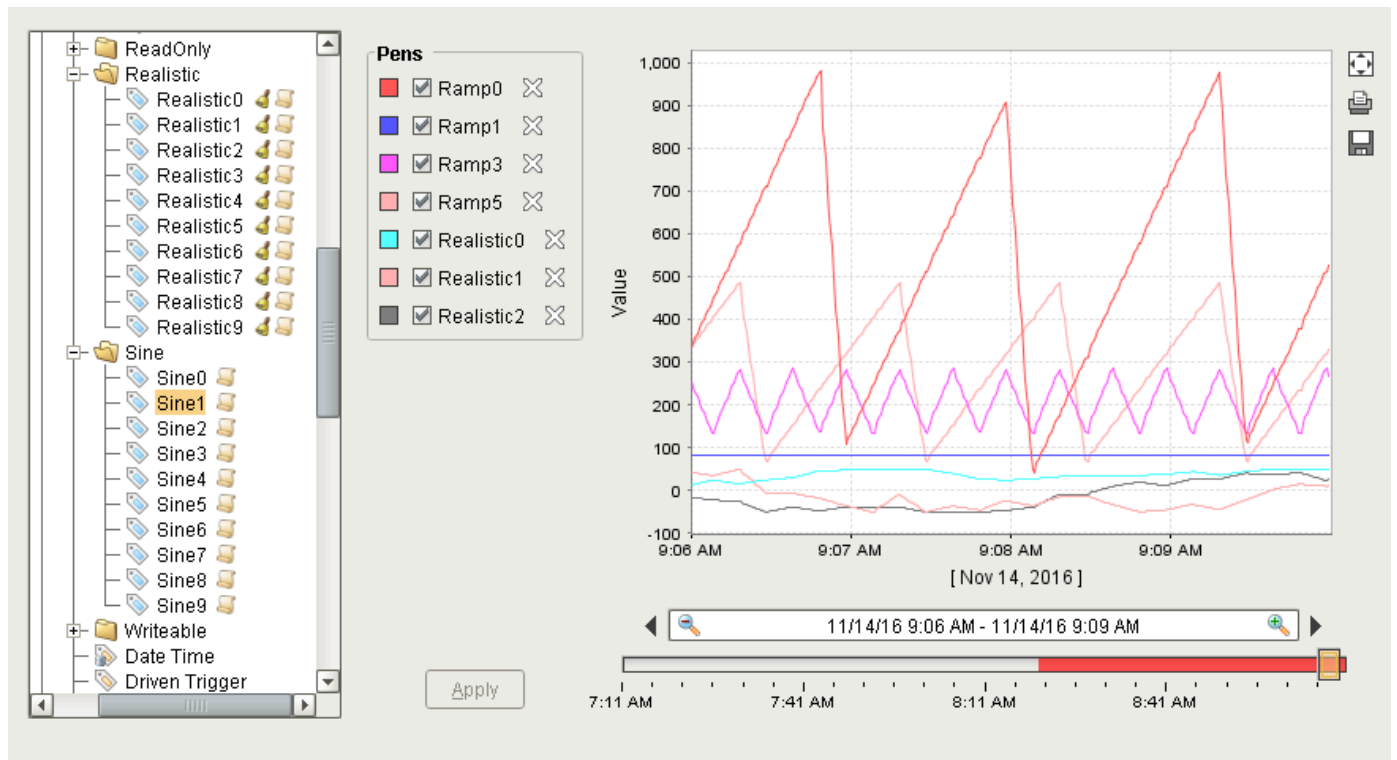
2. In the **Property Editor**, there are a couple of properties worth mentioning that will help you select Tags on the Tag Browse Tree component.

- **Include Historical Tags** - property that you can uncheck to not include Historical Tags in the event you only want to see Realtime Tags.
- **Include Realtime Tags** - property that you can uncheck to not include Realtime Tags in the event you only want to see Historical Tags.
- **Selection Mode** - property where you can make a single Tag selection or you can select multiple Tags to drag to your Easy Chart.

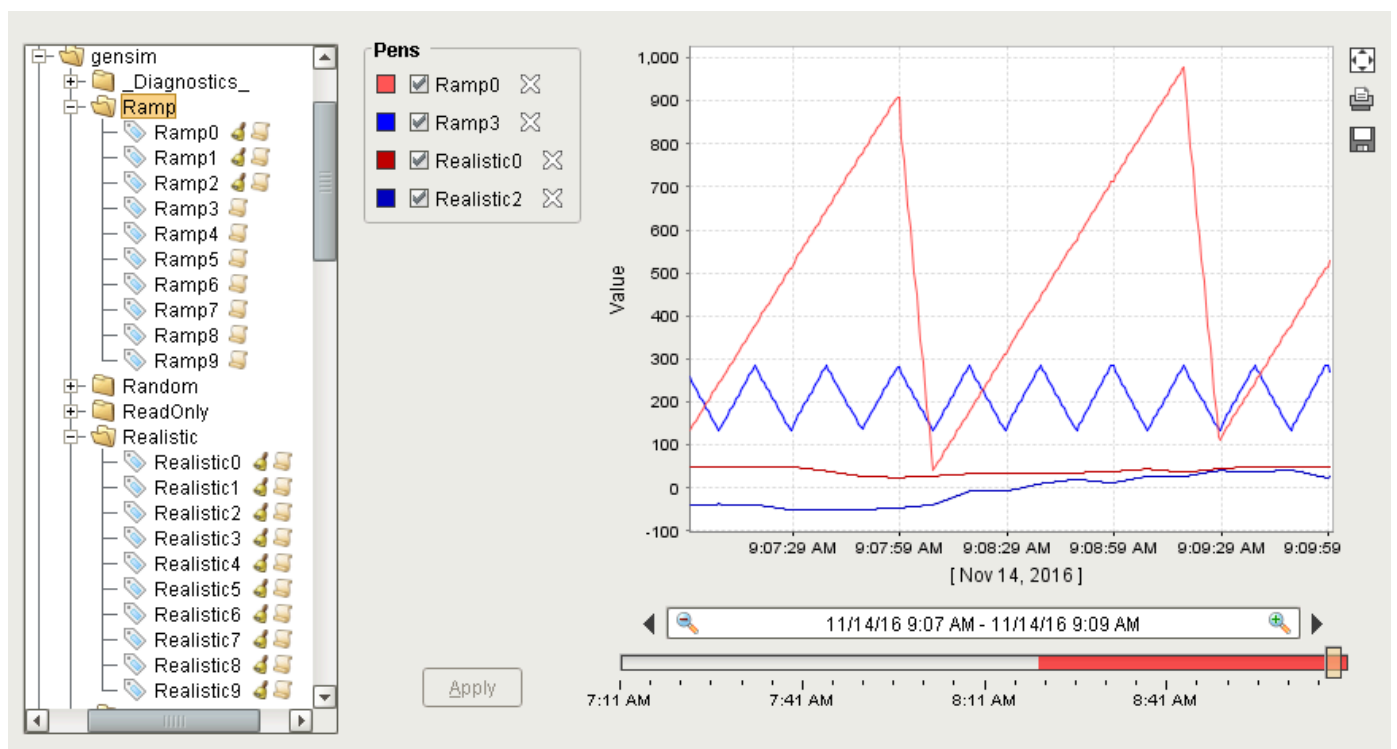


3. By adding the Tag Browse Tree component on the window, each of the components knows exactly how to work together. Format the Tag Browse Tree component, pens and Easy Chart so you can see all three nicely on the window. **Save and Publish** your project, and go to the **Runtime** or **Preview** Mode to try it out.

4. 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 **Realistic** Tags were randomly selected to create the following Ad Hoc Chart.



5. 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.



THINGS TO KEEP IN MIND WHEN WORKING WITH AD HOC CHARTING

- When working in the Designer, whatever pens you have on your Easy Chart when you saved your project, 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.

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

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

Indirect Easy Chart



INDUCTIVE
UNIVERSITY

Indirect Easy Chart

[Watch the video](#)

Configuring an Indirect Easy Chart

i THIS SECTION ASSUMES THAT TAGS AND TAG HISTORY HAVE BEEN CONFIGURED

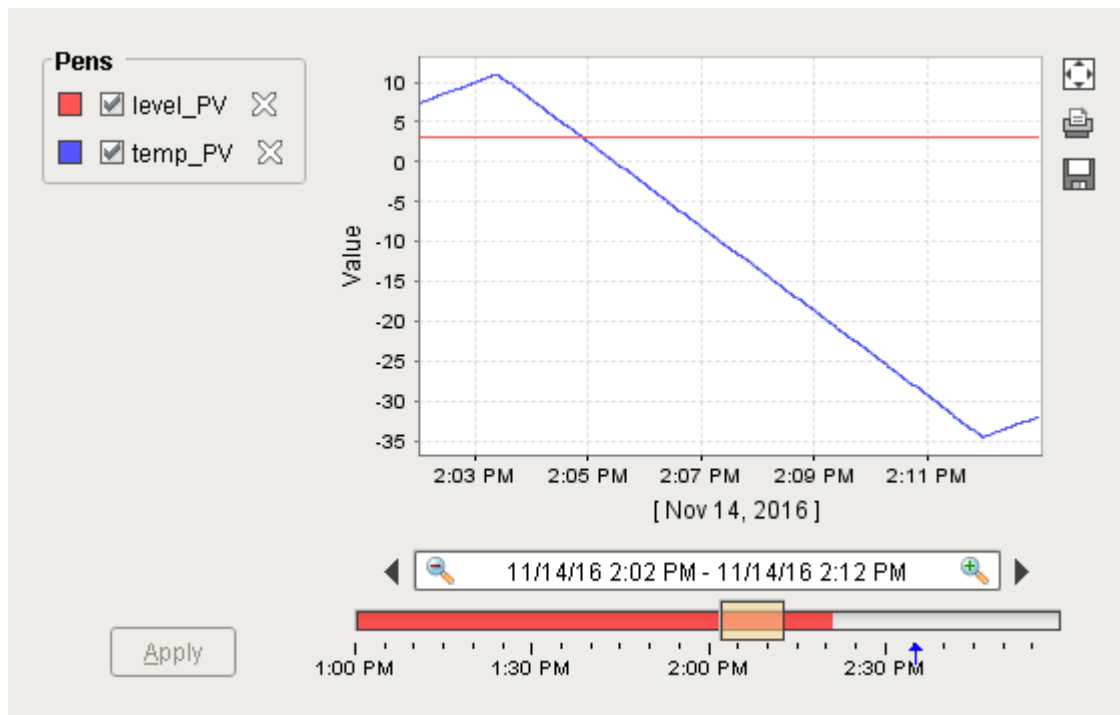
To learn more, go to the [Tag](#) and [Set up Tags to Log](#) pages. The examples below use [OPC tags](#) from the [Generic Simulator](#) driver, but [Memory, Expression, and Query Tags](#) can be used instead.

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 example, suppose you have a small tank farm consisting of Tank 100 through Tank 105. Every tank is identical and uses the same Tags. Each tank has a **level_PV** and **temp_PV** Tag which are set to log to the Historian.

Tag Browser			
Tanks			
Tank 100			Tanks
+ HOA	1	Int2	
+ level_PV	3	Int2	
+ tank_num	100	Int2	
+ tank_status	0	Int2	
+ temp_delta	44.5	Float4	
+ temp_hi_SP	90	Float4	
+ temp_PV	-26.28	Float4	
+ temp_SP	10	Float4	
+ valve_cntl	1	Int2	
+ valve_status	1	Int2	
+ Tank 101			Tanks
+ Tank 102			Tanks
+ Tank 103			Tanks
+ Tank 104			Tanks
+ Tank 105			Tanks

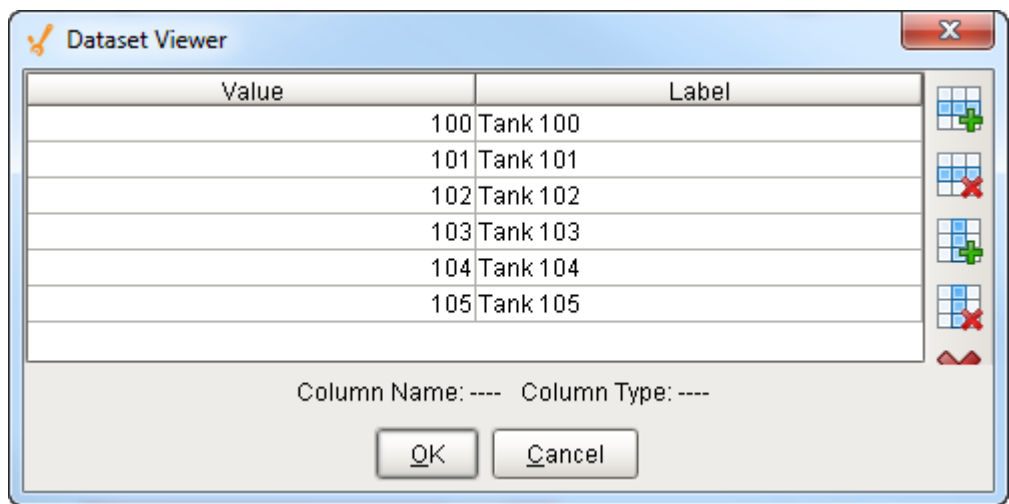
Using an Indirect Easy Chart is a good way to see the history of these Tags for each tank in the tank farm. An easy way to do this is, is by using a dropdown list where an operator can select various tanks to see the Tag history and have the data change dynamically as you select different tanks. Let's give it a try!

1. In the Designer, drag an Easy Chart component into your workspace. Drag the '**level_PV**' and '**temp_PV**' Tags from the Tag Browser to the Easy Chart to see the history of these two Tags for Tank 100.

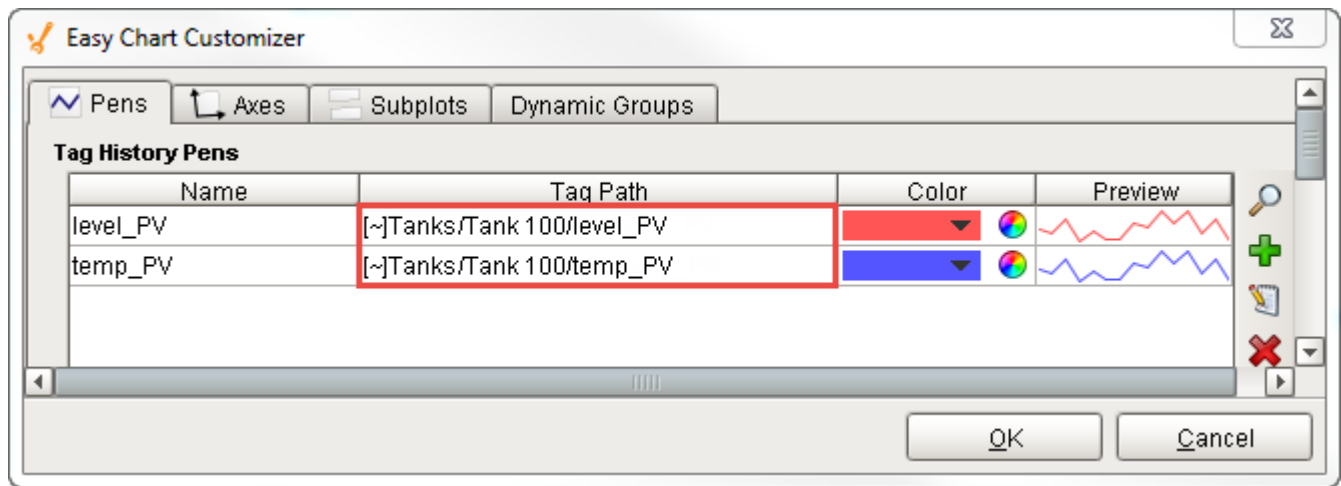


2. In Design Mode, drag a **Dropdown** component to your workspace. Using a Dropdown list will make it convenient for the operator to select a tank and see the history of the **level_PV** and

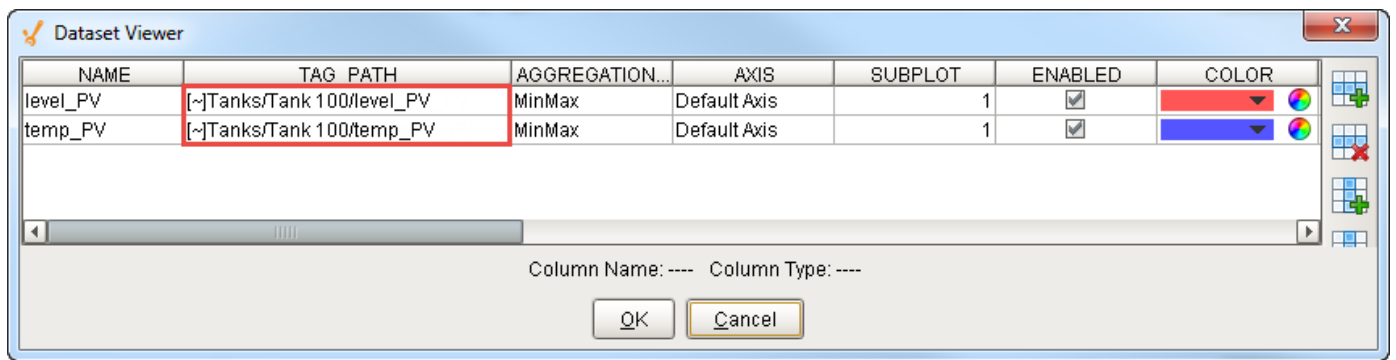
temp_PV Tags for that tank. Double click on the Dropdown component and enter the Values and Labels for all the tanks in your tank farm. Click **OK**.



3. In **Preview Mode**, click the dropdown list to see the complete list of tanks in your Tank Farm that you just entered.
4. Go back to **Design Mode**, right click on your Easy Chart, and scroll down to **Customizers > Easy Chart Customizer**. You'll notice that the Tag Paths are pointing directly to Tank 100. The only difference between Tank 100 and all the tanks in the tank farm is the tank number (i.e., 101, 102, 103, etc.). You can manually point to a different Tag Path by replacing '100' in the Tag Path with a different tank number such as '101,' but this defeats the purpose. You want the Tag Paths to change dynamically when you select various tanks. Press **OK**.



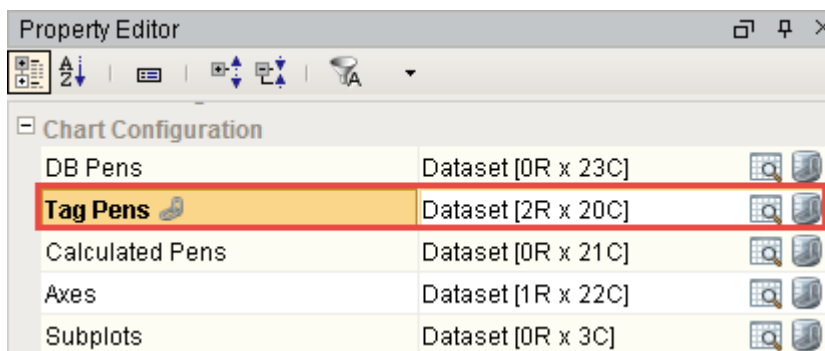
5. The Easy Chart component has a property called **Tag Pens** which stores all the configuration information that you have in the Easy Chart about your pens. In the **Property Editor**, click on the dataset icon to view the information for your **Tag Pens**. The dataset contains two rows, one for each Tag. The second column is the Tag Path that you want to change dynamically. Press **OK**.



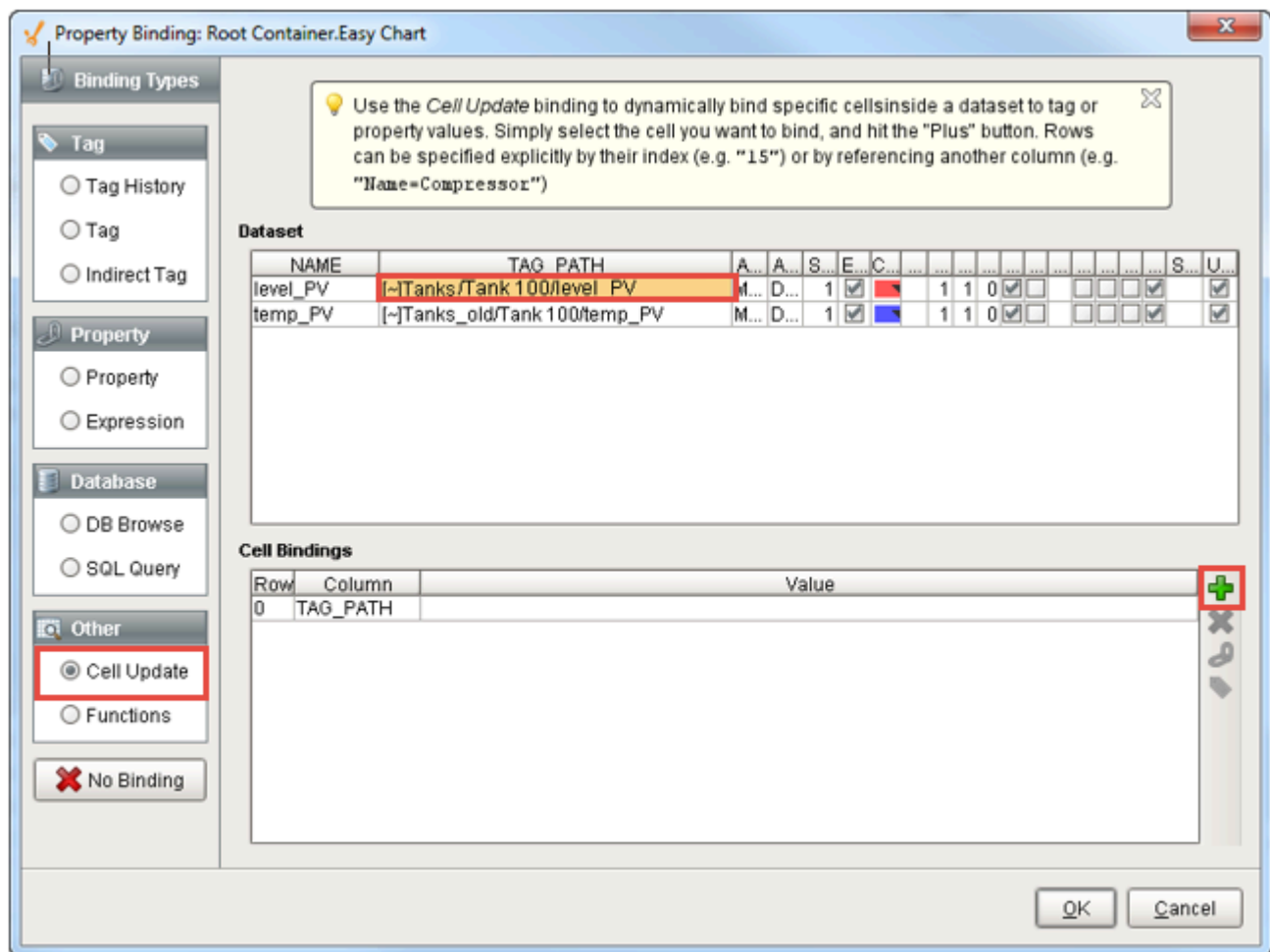
6. In order to change an individual cell of a dataset, you can bind a dataset to a Cell Update Binding type. In a Cell Update Binding, you can select one or more cells and dynamically bind them to a property or to a Tag that you have in your system.

To change both Tag Paths in Tank 100, **level_PV** and **temp_PV**, use the following instructions to make the Tag Paths dynamic:

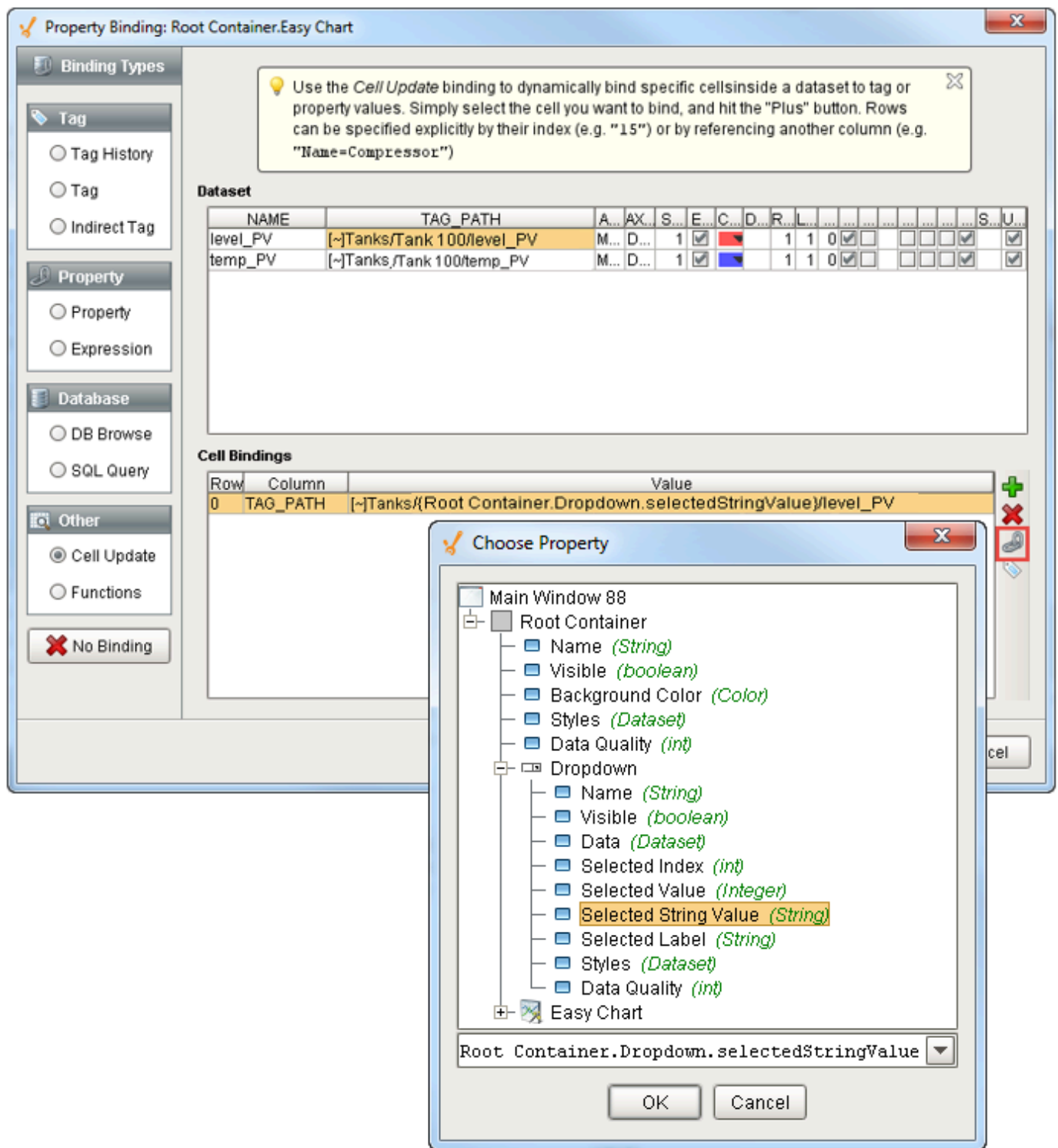
- i. Select the Easy Chart, and in the Property Editor click on the **Binding** icon next to the **Tag Pens** property. The Property Binding window will open.



- ii. Click the **Cell Update** binding type.
- iii. Select Tag Path cell for the **level_PV**, and click on the **green plus** icon under the **Cell Bindings Table**.



- iv. Select the first row you just added in the Cell Bindings Table, and click the **Link** icon.
- v. From the Property Window under the **Dropdown** folder, click the **Selected String Value**.
This will grab the Tank number the operator selected from the dropdown list. Click **OK**.



vi. Next, build out your Tag Path for the level_PV and temp_PV as shown below:

Cell Binding to make Tag Path Dynamic

```
[~]Tanks/{Root Container.Dropdown.selectedStringValue}/level_PV
[~]Tanks/{Root Container.Dropdown.selectedStringValue}/temp_PV
```

- vii. Now, let's repeat the **Steps i-vi** for temp_PV. Select the **temp_PV** row under the Dataset Table, and click the **green plus** icon in the Cell Bindings Table to add a second row.
- viii. You can cheat a little and copy the Tag_Path from the level_PV in the Cell Bindings Table and paste it in the second row for temp_PV. Delete **level_PV** and enter **temp_PV**.

ix. The Tag Paths for the level_PV and temp_PV are shown below. Click **OK**.

Property Binding: Root Container.Easy Chart

Binding Types

- ☒ Tag
 - ☒ Tag History
 - ☐ Tag
 - ☐ Indirect Tag
- ☐ Property
 - ☐ Property
 - ☐ Expression
- ☐ Database
 - ☐ DB Browse
 - ☐ SQL Query
- ☐ Other
 - ☒ Cell Update
 - ☐ Functions
- ☒ No Binding

Dataset

NAME	TAG_PATH	A...	A...	S...	...	...	...	...	...	...	...	...	...	...	...	...	...	S...	U...
level_PV	[~]Tanks/Tank 100/level_PV	...	...	1	✓	■	...	1	1	0	✓	...	...	...	...	...	...	✓	
temp_PV	[~]Tanks/Tank 100/temp_PV	...	...	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

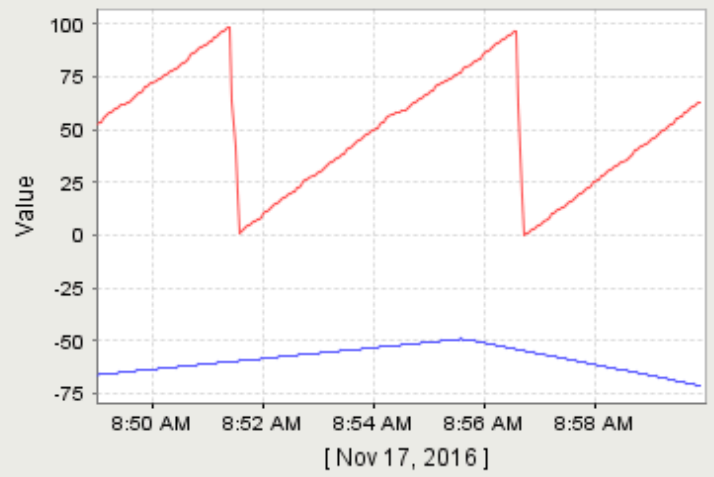
OK Cancel

7. Now you have an Indirect Easy Chart. In **Preview Mode**, when an operator selects a different tank, the level_PV and temp_PV Tags will change. Select different tanks from your dropdown list and see how the history changes for the selected tank on the Easy Chart.

Tank 100

Pens

- ☒ level_PV ☐
☒ temp_PV ☐



11/17/16 8:49 AM - 11/17/16 8:59 AM

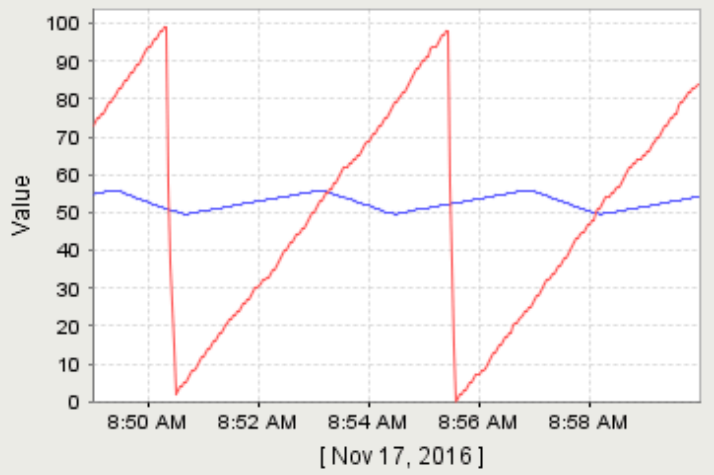
Apply

7:00 AM 7:30 AM 8:00 AM 8:30 AM

Tank 103

Pens

- ☒ level_PV ☐
☒ temp_PV ☐



11/17/16 8:49 AM - 11/17/16 8:59 AM

Apply

7:00 AM 7:30 AM 8:00 AM 8:30 AM

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

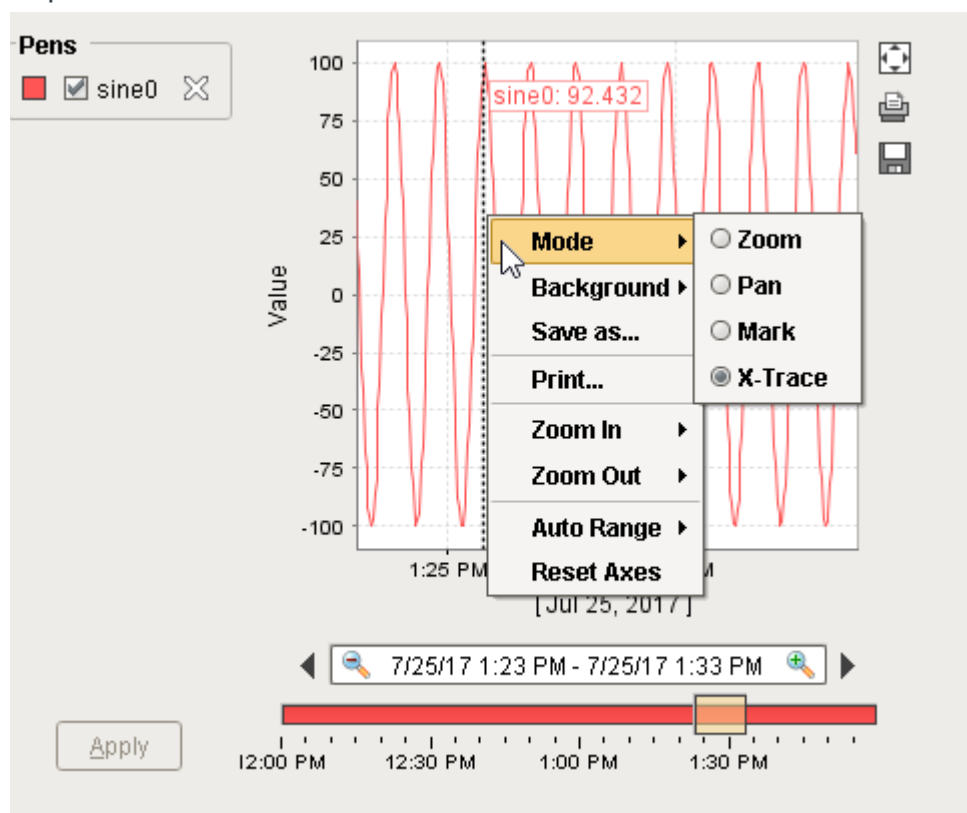
For documentation on the latest stable release, see the [latest version](#) (8.3).

Charting - Right Click Menu

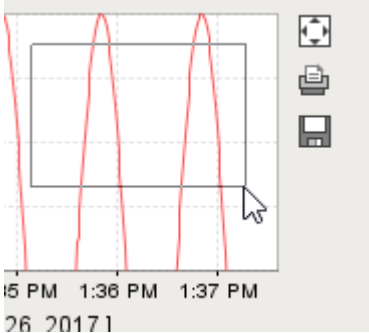
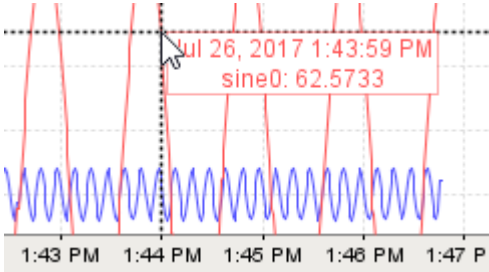
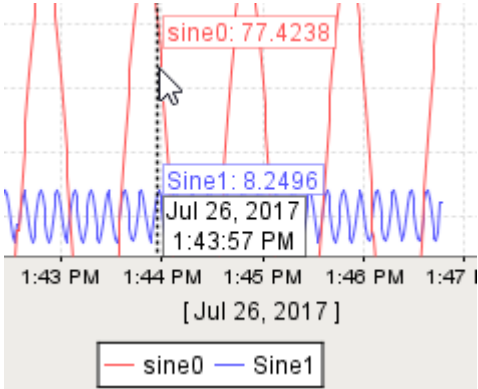
Right Click Menu Functions

When viewed from the client (or the Designer's Preview Mode) many chart components, such as the **Easy Chart** or **Bar Chart**, allows users to right-click and access a menu with additional features.

Explanations of these features are listed below.



Note that many of the behaviors listed below may be overridden by a property on the parent component. For example, the **Easy Chart** provides a way to define the plot background color with the [Easy Chart Customizer](#), so the Background options on the right-click menu would not modify the behavior on the component when the user selects them.

Item	Description
Mode	Specifies the functionality of a left-click on the chart. The following modes are available: • Zoom - Zoom in: Users can left-click and drag on the chart to zoom in or out. Dragging to the right creates a hollow rectangle. When the user releases the mouse button, the viewable space in the chart will zoom in to a rectangle.  Zoom out: Left-click and dragging to the left will not create a rectangle, but releasing the mouse button will zoom back out. • Pan - Allows users to Pan the chart horizontally or vertically by left-click and dragging. The Pan mode may be used after using the Zoom mode: doing so allows panning within the zoomed in view-space. • Mark - While in Mark mode, a left click will target the closest point on the nearest pen, and return the value and timestamp of that point.  • X-Trace - Similar to Mark, X-Trace will instead show the values of all pens at the selected point in time. 

Item	Description
Background	Allows the user to change the background color. Available options include Black and White .
Save as...	Provides a Save window that allows the user to save a PNG of the chart locally.
Print...	Provides a Print dialog that allows the user to print an image of the chart. This will send the chart to a physical printer. Note: a printer must be installed on the client computer.
Zoom In	Allows the user to Zoom in on the chart. There are three possible options: • All axes: Zooms in on both the Range and Domain. • Domain axis: Zooms in on just the Domain, or X- axis. Reduces the range of viewable values on the domain (typically a datetime), while leaving the Range axis untouched. • Range axis: Zooms in on just the Range, or Y-axis. Reduces the range of viewable values (typically numeric), while leaving the Domain axis unmodified. This is helpful when a trend shows very little change over a period of time, as this allows the user to view the trend with increased granularity.
Zoom Out	Allows the user to Zoom out on the chart. There are three possible options: • All axes: Zooms out on both the Range and Domain. • Domain axis: Zooms out on just the Domain, or X- axis. • Range axis: Zooms out on just the Range, or Y-axis
Auto Range	Automatically updates the range, or viewable space, on the chart to incorporate all data points. As mentioned above, many components have properties/customizations that restrict the capability of these (such as the date range on on the Easy Chart). • All axes: Adjusts the viewable space on both axes. • Domain axis: Adjusts the viewable space on just the Domain, or X- axis. • Range axis: Adjusts the viewable space on just the Range, or Y-axis.

Item	Description
Reset Axes	Adjusts the Axes back to their initial state.

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

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

Using the Classic Chart

Chart

The Chart, sometimes referred to as the **Classic Chart**, is capable of rendering time series, XY, and Bar charts.

Configuration

The basic idea behind configuring the classic chart is simple: add datasets, and fill them in with data in a format that the chart understands. You add datasets to the chart using the chart's customizer. You then use a bindings to populate each dataset. Commonly you'll use a **SQL Query Binding**. Since these datasets are just normal dynamic properties, you can also access them via scripting.

The Customizer also lets you add additional X and Y axes. There are various types of axes and each works slightly differently. Lastly, you can configure additional properties for each dataset, such as which axis it maps to, its visual style, subplot, etc.

Datasets

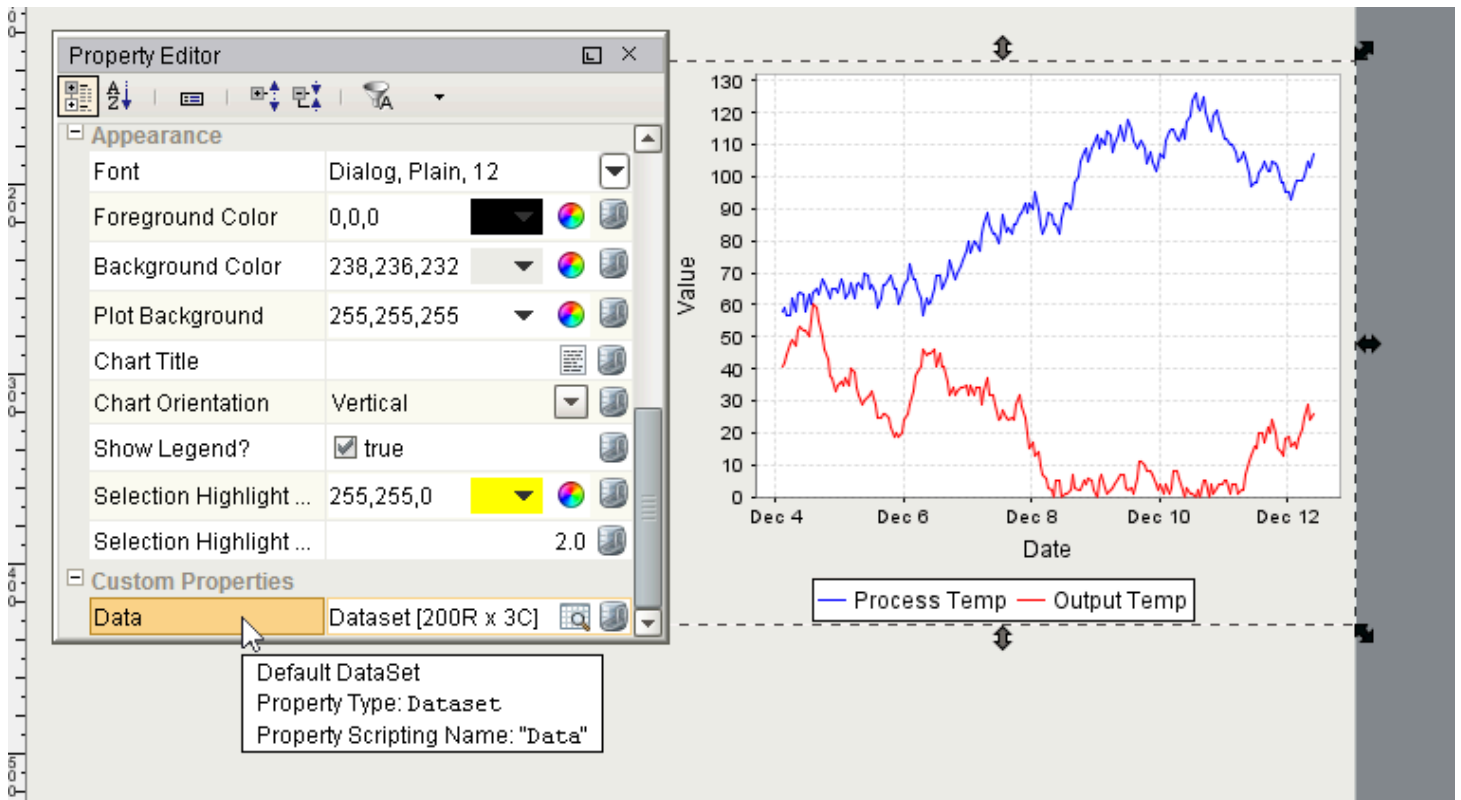
Each dataset should define one or more "series" (a.k.a "pens"). The expected anatomy of a dataset for the Classic Chart depends on several settings: namely the **Extract Order** property on the chart, as well as the type of renderer specified in the Chart Customizer's **Dataset Properties** tab.

Binding Techniques

The classic chart can be used to make almost any kind of chart, with some effort. Historical, realtime, dynamic pen selection, etc., is all possible. Your job is just to fill the datasets with the pertinent data, and the chart will display it. The most common idea is to make the chart dynamic by varying the date range that the dataset's SQL Query bindings run. This is easy to do by adding a **Date Range** component and using **Indirect Bindings**.

Chart Type: XY vs Category

The classic chart is typically in XY Plot mode. This means that the X-axis is either date or numeric, and the Y-axis is numeric. If your X-axis is categorical (names, not numbers), you can switch the Chart Type property to Category Chart in the Property Editor. Don't be surprised when you get a few errors - you'll need to go and switch your X-axis to be a Category Axis, and fill your dataset in with valid category data, that is, String-based X-values. This is most often used with the Bar Renderer (see the [Chart Customizer](#)).



Usage

Populating the Classic Chart involves using Custom Properties to collect data. New Custom Properties are added to the component via the [Chart Customizer](#). Multiple dataset properties may be configured on the component, and each additional dataset will show as a new subplot. Populating the Chart with historical data involves populating one of these Custom Properties.

The default configuration of the component expects a dataset where the first column is a timestamp (the first column *always* acts as the domain for the chart), and sequential columns are pens that should be drawn on the chart. The default dataset contains a **t_stamp** column with values for the domain, and two columns (**Process Temp** and **Output Temp**) with values that will be drawn against the range.

Dataset Viewer

t_stamp	Process Temp	Output Temp
12/04/2017 02:2...	58	41
12/04/2017 03:2...	59	42
12/04/2017 04:2...	57	44
12/04/2017 05:2...	57	48
12/04/2017 06:2...	62	49
12/04/2017 07:2...	58	47
12/04/2017 08:2...	63	52
12/04/2017 09:2...	64	53
12/04/2017 10:2...	63	52
12/04/2017 11:2...	58	52
12/04/2017 12:2...	63	50
12/04/2017 13:2...	60	55
12/04/2017 14:2...	64	60
12/04/2017 15:2...	65	59

Column Name: ---- Column Type: ----

OK Cancel

Example - Tag History

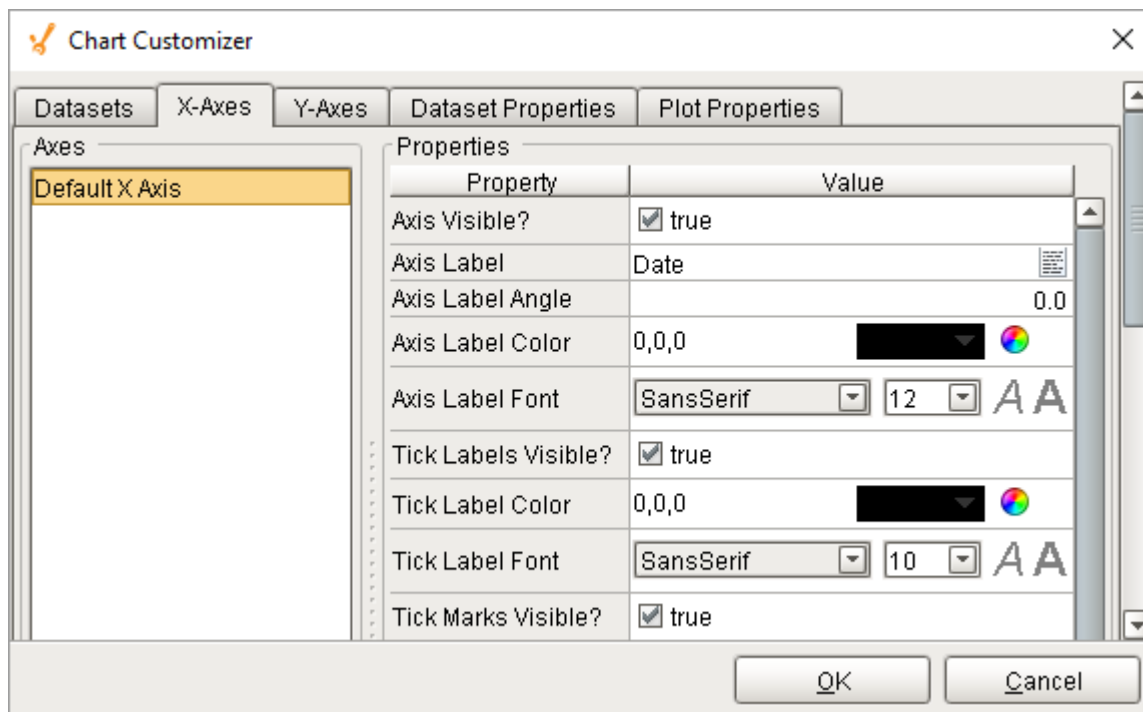
The Classic Chart is initially configured in a manner that easily displays Tag History with a **Tag History Binding**.

1. Create a new Chart component.
2. Locate the **Data** property. It should be located at the bottom of the Property Editor when sorted by section. Click the binding  button.
3. Select **Tag History** from the Tag section of the Binding window. Browse the **Available Historical Tags**, and drag them to the **Selected Historical Tags** table on the right of the window.
4. Change the **Date Range** to **Realtime** with a **Most Recent** set to 10 minutes. Click the **OK** button.

The Chart will now show Tag History from the Tags you selected. Note that the timestamp has automatically been included as the first column, so the domain has been automatically configured for us. If you wish to change the orientation of the axes, then the **Chart Orientation** property in the Property Editor will swap the position of the Domain and Range axes.

Additional Customization using the Chart Customizer

Once trends are present on the chart, additional customization can be achieved through the **Chart Customizer**. When you open the customizer, you'll notice five tabs at the top: Dataset, X-Axes, Y-Axes, Dataset Properties, and Plot Properties. Each tab has its own set of properties and defaults.



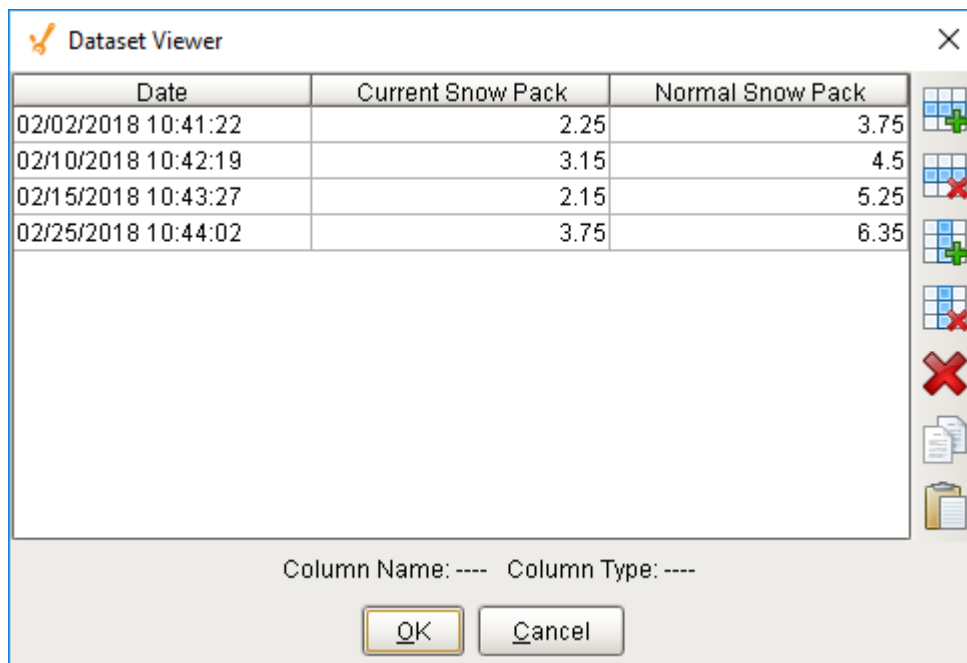
An overview of each section in the Chart Customizer is listed below:

- **Datasets:** Allows additional datasets to be added to the component. Each dataset will appear as a separate custom property on the component. Data from multiple charts can be drawn on the same chart.
- **X-Axes** and **Y-Axes:** Allows for the creation of new X and Y Axes on the chart. There are six types of axes to choose from when configuring a chart, each having its own list of properties: Number Axis, Date Axis, Category Axis, Logarithmic Axis, Elapsed Axis, and Symbols Axis. Most of the X and Y axes properties are used in the customizer, and some properties are specific to the axis type and have their own unique properties. See the [Chart Customizer](#) page for more information on types of axis, associated properties, and examples.
- **Dataset Properties:** Specify which axes should a be used with each dataset. Also allows you to specify which subplot each dataset should be shown on.
- **Plot Properties:** Configure the look of each plot.

Chart Customizer Example

This example walks you through adding a new dataset to a chart, and modifying the existing chart properties using the Chart Customizer. When you first drag a chart to your window, you'll notice that it will display some data, that's because it's using the default dataset provided. For this example, you can add a new dataset by either copying and pasting the one below, or adding your own.

1. Let's add a new dataset (i.e., SnowPackData) that measures the Snow Pack Level for the month of February. Double click on the chart to open the Chart Customizer. Go to the Datasets tab and add (+) a new dataset.



Dataset Viewer

Date	Current Snow Pack	Normal Snow Pack
02/02/2018 10:41:22	2.25	3.75
02/10/2018 10:42:19	3.15	4.5
02/15/2018 10:43:27	2.15	5.25
02/25/2018 10:44:02	3.75	6.35

Column Name: ---- Column Type: ----

OK Cancel

Classic Chart - SnowPack Dataset

```

"#NAMES"
"Date","Current Snow Pack","Normal Snow Pack"
"#TYPES"
"date","F","F"
"#ROWS","4"
"2018-02-02 10:41:22","2.25","3.75"
"2018-02-10 10:42:19","3.15","4.5"
"2018-02-15 10:43:27","2.15","5.25"
"2018-02-25 10:44:02","3.75","6.35"

```

- Once you add your dataset, the chart will show the data from the default Dataset and your new dataset, so let's disable the default dataset by going to the **Dataset Properties** tab, selecting the default dataset (i.e., Data) and unchecking the **Enabled?** box. You'll notice the chart is now only displaying the values for the new dataset (i.e., SnowPackData).

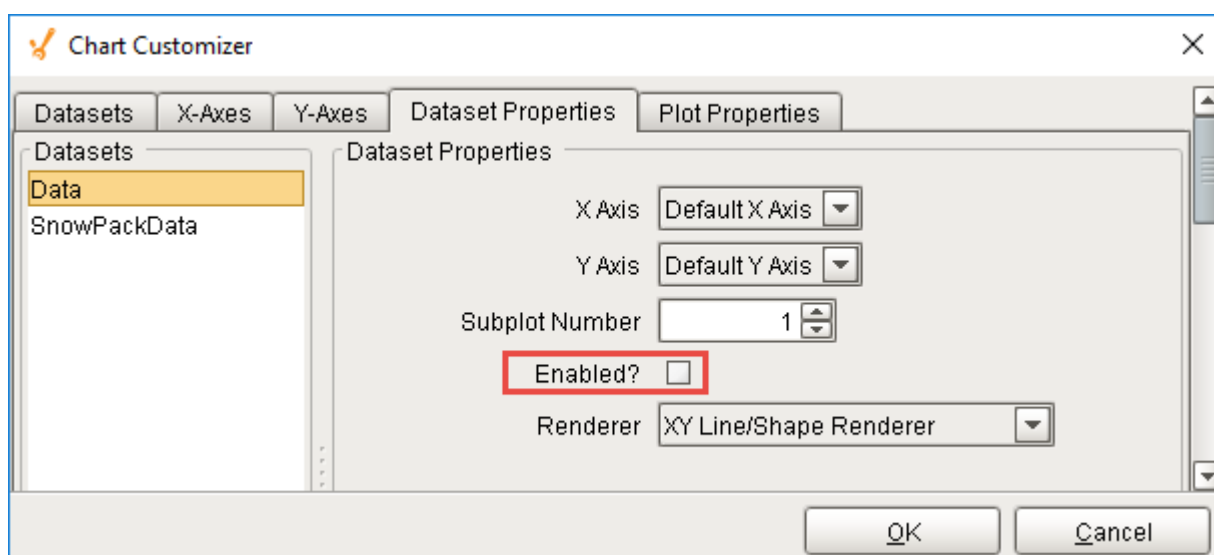


Chart Customizer

Datasets X-Axes Y-Axes Dataset Properties Plot Properties

Datasets

- Data
- SnowPackData

Dataset Properties

X Axis: Default X Axis

Y Axis: Default Y Axis

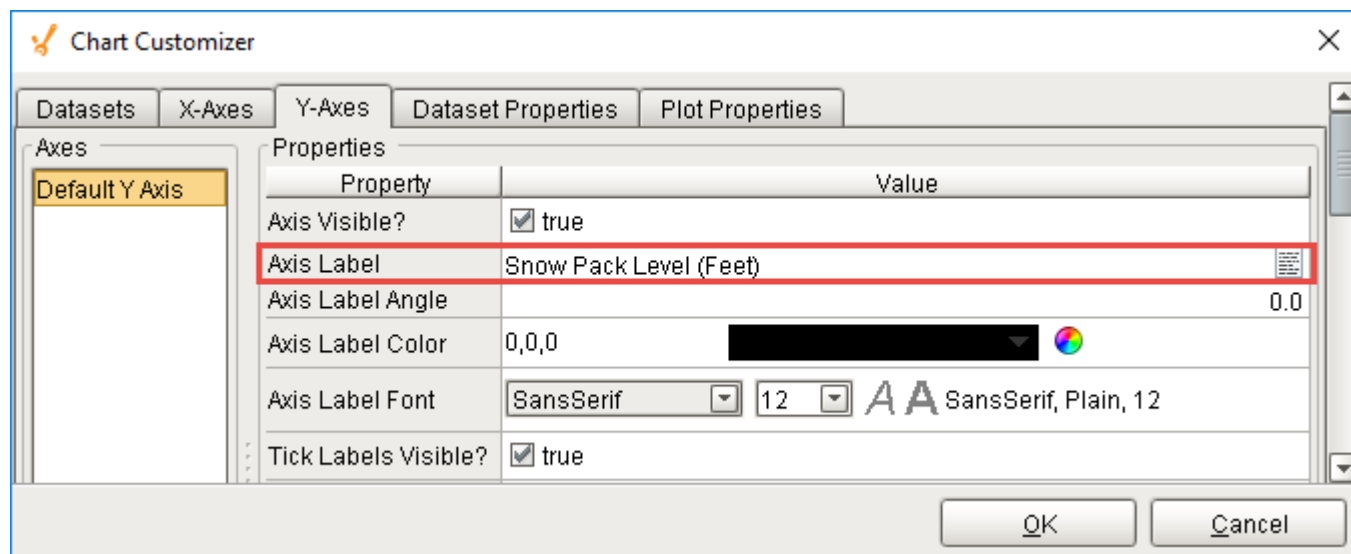
Subplot Number: 1

Enabled? ☐

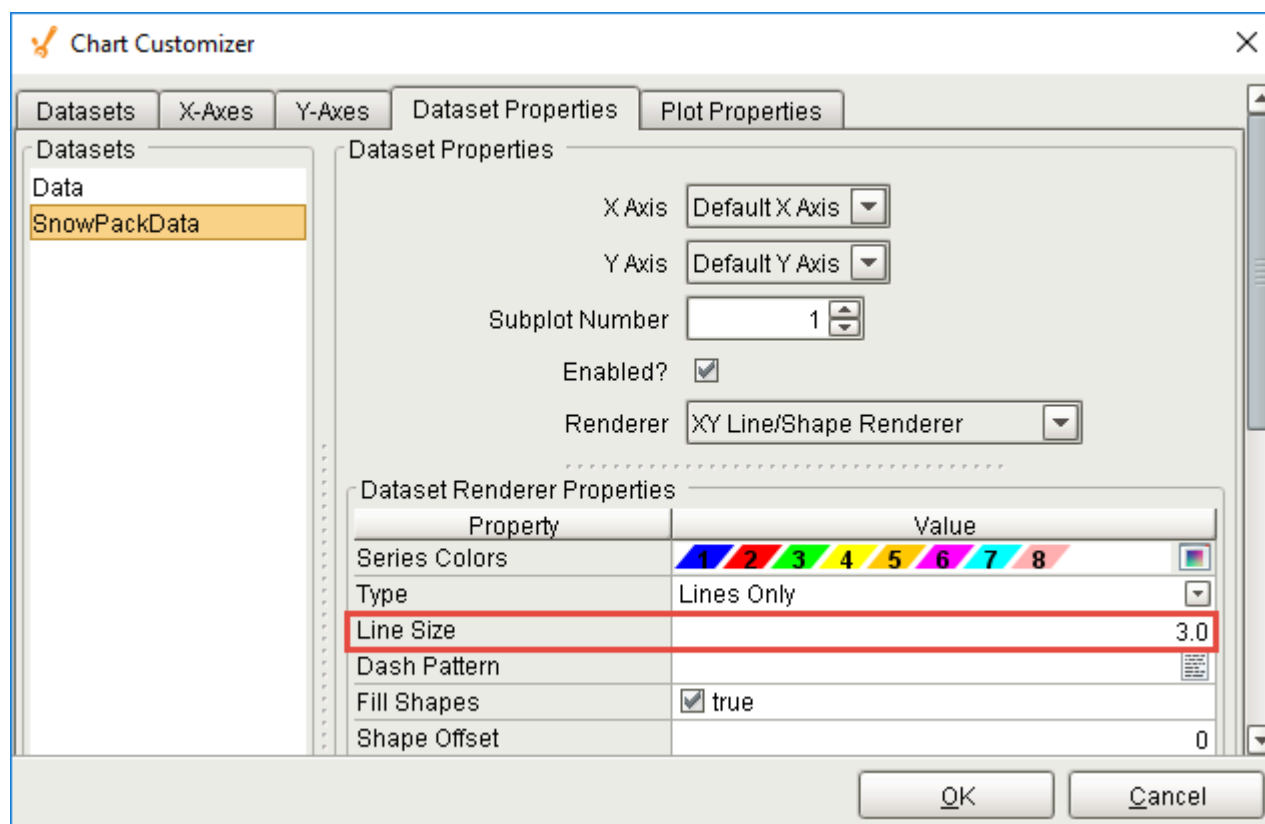
Renderer: XY Line/Shape Renderer

OK Cancel

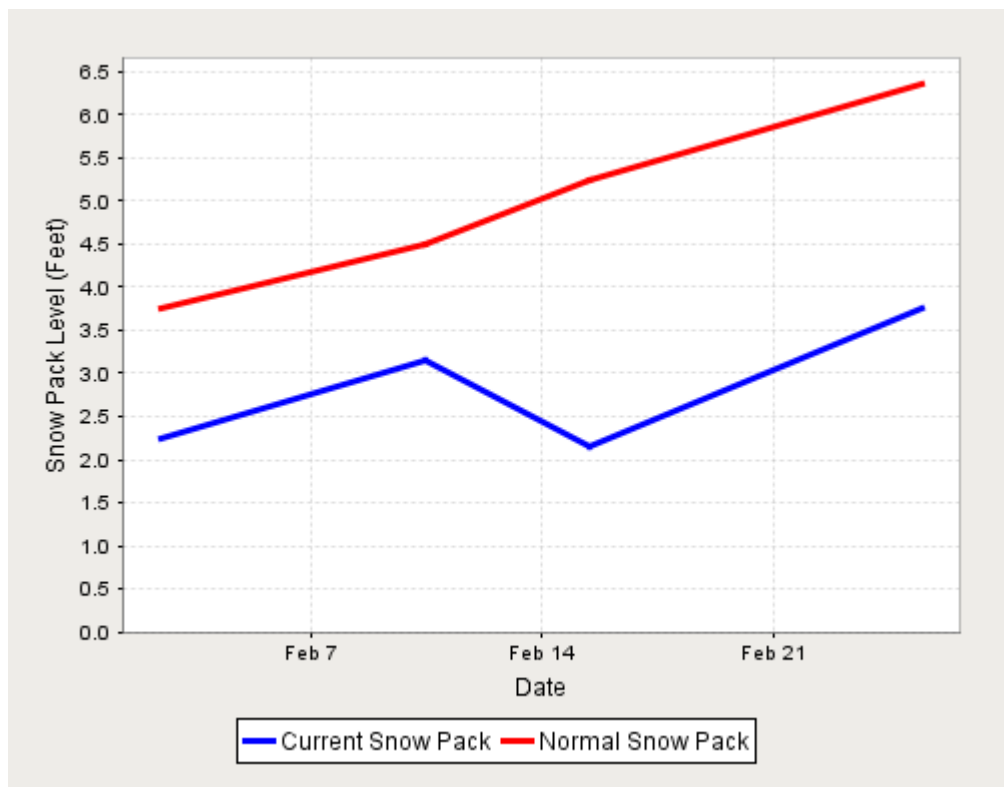
3. This example uses the default Number Axis type, but if you want to add a new Axis Type, go the **X and Y Axes tabs** and select a new Axis type.
4. Next, let's change a few visual properties on our chart. Go to the **Y-Axes tab** and select Default Y Axis, and change the **Axis Label** (i.e, Snow Pack Level (Feet)).



5. Lastly, let's make the plot lines thicker on the chart. Go to the Dataset Properties tab, and change the **Line Size** renderer property from 1.0 to **3.0**.



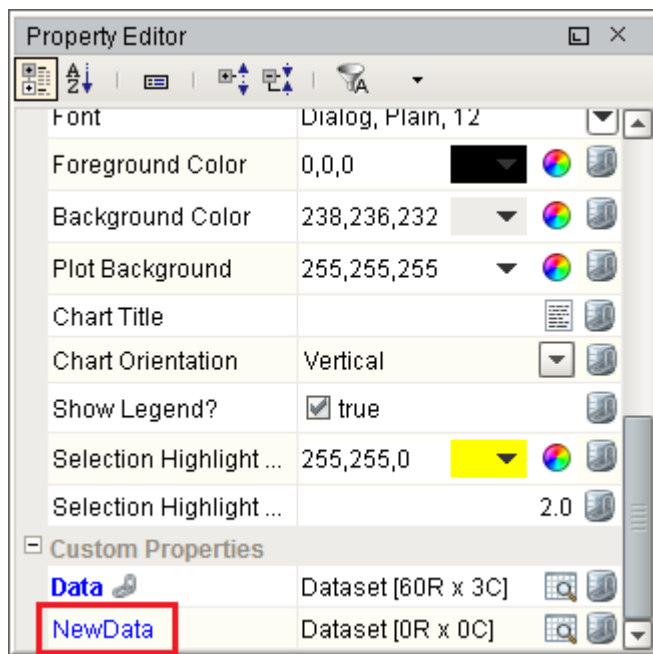
6. Here is the chart with the new dataset and updated properties. As you can see, you can easily configure additional properties for a dataset, as well as choose from a host of visual style properties to design your charts using the Chart Customizer.



Adding Another Dataset

In this example, we'll add a separate dataset to the chart.

1. Open the **Chart Customizer**: either double click on the chart or right-click and select **Customizers > Chart Customizer**.
2. Once the Chart Customizer is open, make sure the **Datasets** tab is selected, and click the add **+** icon to add a new dataset.
3. A new row will appear. Give the new dataset a name by typing into the cell under the **Name** column. We'll call it **NewData**. Click the **OK** button.
4. Check the bottom of the Property Editor. The newly created dataset will appear and may now be populated with data.



5. Now that we have a new dataset, you can add data however you like and it will show up in a second subplot.

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

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

Other Vision Trending Charts

Overview

Aside from the [Easy Chart](#) and [Classic Chart](#), there are several other types of charts contained in the Vision Module that can also visualize a trend and track a change in condition, output, or process through a single data point or multiple data points over time. This page describes the Sparkline and Status Charts and how they can be used.

Sparkline Chart



INDUCTIVE
UNIVERSITY

Sparkline Chart

[Watch the video](#)

The [Sparkline Chart](#) is a minimalistic chart that displays a line-chart history for a single data point that fits well with [High Performance HMI screens](#). It's a great way to show a lot of contextual information in a very small amount of space. It provides a quick way to display the recent history of a single data point so the viewer can quickly discern the most recent trend.

Note, the Sparkline Chart can not show multiple trends, however, multiple Sparkline Charts can be stacked. This chart is often laid over an image or level display to show a simple trend with the current value.

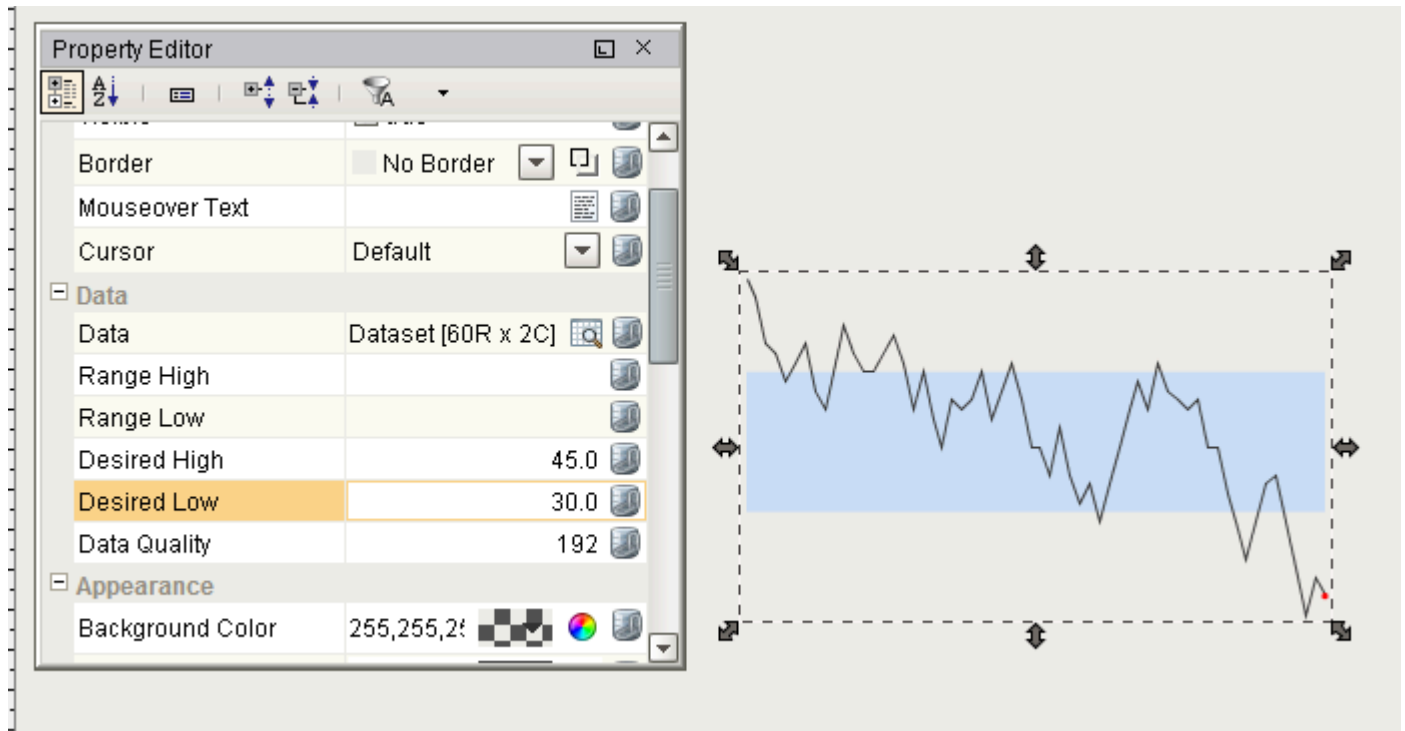
Usage

The trend on the Sparkline Chart is determined by the **Data** property located in the Property Editor. Bind the Data property to either a Tag Historian realtime query, or to a database query. The dataset

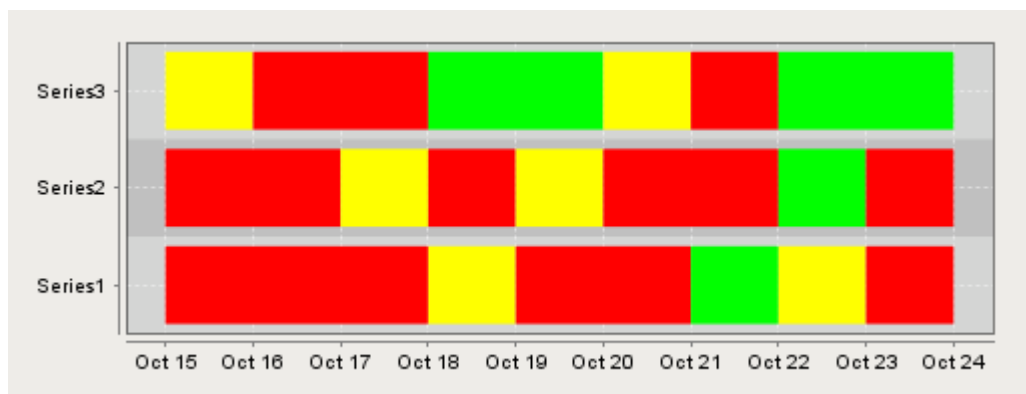
should contain two columns: the first being a **date**, and the second a **number/value**.

Any additional columns in the dataset are ignored. Thus, the Sparkline Chart will only show a single pen per instance of the component. The data should be sorted by date in ascending order.

The red dot on the Sparkline Chart represents the most recent value. The Sparkline Chart can display a band of color across the back of the chart which indicates the desired operating range of the datapoint. Setpoints can be displayed on the chart by setting a value for the **Desired High** and **Desired Low** properties. This allows an easy way to view when the value fell out of bounds. Both properties must have a value for the **Desired Range Color** to appear.



Status Chart



The **Status Chart** allows you to visualize the status of one or more discrete data points over a time range. The X-axis is always a time series axis, and the Y-axis is a category axis, with one entry per data series. The Status Chart is populated with the **Series Data** property in the **Property Editor**. This

chart is good for showing machine states or HOA values over time. It is recommended to include some sort of Legend when using this chart.

Series Data

The first column of the **Series Data** property must be a datetime column. Each additional column should be numeric (the default columns are doubles). The order of the columns (left-to-right) determines the order of the entries on the chart (bottom-to-top). Because of this, re-ordering the entries would involve changing the order of the columns as they appear in the Series Dataset by modifying the mechanism that is populating the dataset (i.e., changing the order of columns in a query).

Wide vs Tall Datasets

In **Wide** format, all of the columns but the first must be numeric. These "series" columns' headers will be used as the names on the y-axis. In **Tall** format, there should be exactly three columns. The first is the timestamp, the second is the series name, and the third is the value. For example:

Wide Format

t_stamp	Valve1	Valve2
2010-01-13 8:00:00	0	2
2010-01-13 8:02:00	0	2
2010-01-13 8:04:00	1	2
2010-01-13 8:06:00	1	1
2010-01-13 8:08:00	0	1

Tall Format

t_stamp	Name	Value
2010-01-13 8:00:00	Valve1	0
2010-01-13 8:00:00	Valve2	2
2010-01-13 8:02:00	Valve1	0
2010-01-13 8:02:00	Valve2	2
2010-01-13 8:04:00	Valve1	1
2010-01-13 8:04:00	Valve2	2
2010-01-13 8:06:00	Valve1	1
2010-01-13 8:06:00	Valve2	1
2010-01-13 8:08:00	Valve1	0
2010-01-13 8:08:00	Valve2	1

Color Mapping

Apart from getting the data into the series chart, the only other commonly configured option is the mapping of discrete values to colors. This is done in the Status Chart Customizer. Each named series can have its own mapping of colors, if desired. These mappings are stored in the expert-level dataset property **Series Properties Data** so they can be altered at runtime.