

Ignition 8.1 User Manual — Reference Pages and Tutorials

Generated 2026-10-12 from
<https://www.docs.inductiveautomation.com/docs/8.1/>

Reference Pages

This section is for various reference pages that may fall outside of a specific section in the User Manual.

Gateway Configuration File Reference

Charting - Right Click Menu

[Learn more about the right-click menu on Vision's Easy Chart Component](#)

Gateway Port Reference

The following table describes the default ports an Ignition server may use for communication.

Color Selector Reference

[Learn more about the color selection interface here](#)

Ignition Database Table Reference

Ignition has a lot of systems built-in that will query the database automatically without requiring you to build a qu...

Perspective Event Types Reference

[Learn more about events available under the Configure Events menu on Perspective components.](#)

Style Reference

[Learn more about the style user interface used in Perspective](#)

Symbol Reference

Data Type Formatting Reference

This page formatting patterns for dates, numbers, and strings.

Scripting Object Reference

This page describes several common scripting objects

Crontab Formatting Reference

Logback XML File Reference

This page describes the logback.xml file located in the installation directory

Gateway XML File Reference

This page describes the gateway.xml file located in the installation directory

resource.json File

This page describes the contents on resource.json files

System Name Usage Reference

Learn more about how the system name property on a gateway is referenced by other systems within Ignition

Gateway Configuration File Reference

This section details the various configuration changes that can be made to the Gateway's configuration file: `ignition.conf`. Before the Gateway starts, a wrapper service checks the configuration file for parameter values, and then attempts to start the Gateway using parameters. This means changes made to the configuration file will only occur on startup, and making changes to the configuration file while the Gateway is already running will not have any impact until a restart.

Configuration File Location

The Gateway's configuration file can be found at:

```
%IgnitionInstallationDirectory%/data/ignition.conf.
```

See the [Installing and Upgrading Ignition](#) page for default installation directories.

Comments in the File

Many lines in the configuration file are commented out, either to provide some documentation in the file, or to omit certain parameters, such as some options under the Java Additional Parameters header. The "#" character is used to comment lines in the configuration file.

```
# this line is commented
this line is not commented
```

Gateway Memory Allocation



INDUCTIVE
UNIVERSITY

**Changing Gateway Memory
Allocation**

[Watch the video](#)

On a standard installation, the Gateway is initially allocated a fixed amount of heap (memory). Increasing this allows the Gateway to utilize more of the server's memory. To determine if this is necessary, navigate to the **Performance** page under the Status section of the Gateway webpage. Generally speaking, if the Memory Trend shows a saw tooth pattern that peaks towards the maximum amount of memory, that's expected behavior. However, if memory usage usually floats around maximum allocated, and slow response events occur more than occasionally, then it may be time to consider increasing the maximum allocated memory, assuming the server has available memory.

To increase the maximum amount of memory allocated to the Gateway, open the Gateway configuration file, and search for "`wrapper.java.maxmemory`". It should look something like following:

```
# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=1024
```

The number at the end of the line represents the maximum amount of memory allocated to the Gateway (in megabytes). To change the amount of memory allocated to the Gateway, simply change this number, save the configuration file, and restart the Gateway. Once the Gateway starts up again, it will attempt to use the amount of memory specified.

The new heap size shouldn't exceed the amount of memory available, nor should you allocate all of the server's memory to the Gateway because the server likely has other applications (including the host operating system) that also need to make use of the system's resources.

Changing Java Additional Parameters

CAUTION

Adding or modifying parameters in the configuration file is considered an advanced configuration change. Most installations don't require any additional parameters, nor do they require modification to existing parameters. We generally discourage most users from making changes to parameters in the configuration file, as doing so could result some unintended behavior or security vulnerabilities. We list the parameters on this page for the sake of transparency.

If you choose to add or modify parameters in the the configuration file, you do so at your own risk.

A section of the configuration file contains a header stating "Java Additional Parameters". This section allows for a large number of configuration changes, and merits having some discussion on how to add new parameters. On install, the Java Additional Parameters section may look like the following:

```
# Java Additional Parameters
wrapper.java.additional.1=-Ddata.dir=data
#wrapper.java.additional.2=-Xdebug
#wrapper.java.additional.3=-
Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=*:8000
```

When adding a new parameter, the "wrapper.java.additional.#" prefix must be added to a new line. Each parameter contains a **prefix** ("wrapper.java.additional.#"), a **key**, and a **value** that the key will be set to. Generally speaking, each parameter in the file should follow the pattern below:

```
wrapper.java.additional.1=-key=value
```

Uncommented parameters should be listed in ascending numerical order, based on the number at the end of the prefix, as shown below.

```
# Add parameters like this:
wrapper.java.additional.1=-Ddata.dir=data
#wrapper.java.additional.2=-Xdebug
#wrapper.java.additional.3=-
Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=*:8000
wrapper.java.additional.2=-some.thing=foo
wrapper.java.additional.3=-some.other.thing=bar

# Avoid skipping commented numbers:
wrapper.java.additional.1=-Ddata.dir=data
#wrapper.java.additional.2=-Xdebug
#wrapper.java.additional.3=-
Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=*:8000
wrapper.java.additional.4=-some.thing=foo
```

Java Service Wrapper Parameters

Java Service Wrapper, developed by Tanuki Software, is used to control various Wrapper settings (i.e. wrapper.logfile.maxsize and wrapper.logfile.maxfiles). For a list of parameters and their function, visit [Tanuki Software's documentation](#).

Common Localization Parameters

You may sometimes need to change Ignition's locale information in the Gateway Configuration File. The following parameters will allow you to change the most commonly used localization information.

```
# Locale - Language (United States English)
wrapper.java.additional.1=-Duser.language=en_US

# Locale - Country (United States)
wrapper.java.additional.2=-Duser.country=US

# Locale - Variant (English (United States, Computer))
wrapper.java.additional.3=-Duser.variant=POSIX

# Timezone (Pacific Time)
wrapper.java.additional.4=-Duser.timezone=America/Los_Angeles
```

File Encoding Techniques

You might also need to change how your files are encoded. The following parameter will allow you to change the encoding technique to what you require.

```
# File Encoding (Using UTF-8)
wrapper.java.additional.1=-Dfile.encoding=UTF-8
```

Gateway and Gateway Network Parameters

See the [Gateway and Gateway Network Parameters](#) page for more Gateway and Gateway Network parameter information.

Vision Parameters

See the [Vision Client Parameters](#) page for more Vision parameter information.

Perspective Parameters

See the [Perspective Parameters](#) page for more Perspective parameter information.

Data Logging Parameters

Store and Forward

New in 8.1.2

It is possible to adjust the refresh rate of the System Tag Provider's System tags. To do so, add the following system flag in the `ignition.conf` file:

Example

```
wrapper.java.additional.1=-DStoreAndForwardStatusTagRefreshRate=####
```

Where "####" is the rate in milliseconds that the tags should refresh at. Default refresh rate is 5000 milliseconds.

Tag Historian Query Syntax

New in 8.1.2

The Ignition Tag Historian stores and queries data in a particular way. When a tag has Tag History enabled on it, an entry is generated for this tag inside the `sqlth_te` table in your database. This tag's tag path, along with other tag attributes, are stored in this table. Every tag entry in the `sqlth_te` table has a unique tag id which is used by the Tag Historian to identify each tag.

Whenever we make any type of change to this tag's historical configurations, such as its Tag Group for instance, the current `sqlth_te` table entry for this tag becomes retired and a new, unretired entry for this tag gets created with a new tag id. Due to this dynamic, it is common for a tag path in the `sqlth_te` table to be associated with more than one tag id. When a Tag History query executes for a specific tag, a check is made against the `sqlth_te` table for all the tag id's that match this tag's tag path. These tag id's are then used to build a dynamic SQL query like the one shown below:

```
SELECT "tagid", "intvalue", "floatvalue", "stringvalue", "datevalue", "t_stamp",  
       "dataintegrity"  
FROM sqlth_1_data  
WHERE "t_stamp" >= 1580680800000 AND "t_stamp" <= 1581026400000  
      AND ( "tagid" = 14568 OR "tagid" = 14571 OR "tagid" = 14572 )  
ORDER BY "t_stamp" ASC, "tagid" ASC
```

When we query tag history for a tag with three tag id's associated with its tag path, the system uses repetitive OR clauses to account for all the tag id's for this tag. Additional OR clauses in this fashion can be hard for databases to optimize. For this reason, we changed the query generating mechanism to use the IN operator like so:

```
SELECT "tagid", "intvalue", "floatvalue", "stringvalue", "datevalue", "t_stamp",  
"dataintegrity"  
FROM sqlth_1_data  
WHERE "t_stamp" >= 1580680800000 AND "t_stamp" <= 1581026400000  
      AND "tagid" IN (14568, 14571, 14572)  
ORDER BY "t_stamp" ASC, "tagid" ASC
```

The use of the `IN` operator allows for better database side query optimization. Users who want to change their historian to use the old query constructor with `OR` operators can do so by placing the following system flag in the `ignition.conf` file:

The use of the `IN` operator allows for better database side query optimization. Users who want to change their historian to use the old query constructor with `OR` operators can do so by placing the following system flag in the `ignition.conf` file:

Example

```
wrapper.java.additional.1=-Dignition.taghistorian.useLegacyQuerySyntax=true
```

Leased Licensing Parameters

New in 8.1.38

Two Java parameters exist for **leased (eight-digit) licenses** and are described below:

Session Termination HTTP Timeout Value

Prior to version 8.1.38, the default value could not be changed from 4 seconds. For version 8.1.38+, the default value is now configurable and has been changed to 7.5 seconds, instead of 4 seconds.

```
# Defines the HTTP timeout for leased activation session termination requests.  
wrapper.java.additional.1=-Dignition.license.leased-activation-session-  
termination-timeout-ms=xxxx
```

Leased Activation Session Termination

Defines whether or not all configured (and active) leased activation license sessions will be terminated when Gateway shutdown is initiated. The default value is false.

```
wrapper.java.additional.1=-Dignition.license.leased-activation-terminate-sessions-  
on-shutdown=false
```


Gateway and Gateway Network Parameters

Gateway Parameters

Gateway Security HTTP Headers and Configuration

Gateway security headers system properties and default settings are listed below. These parameters impact Perspective sessions, as well as any pages hosted on Ignition's web server.

Strict Transport Security

The following system properties interact with the **Strict-Transport-Security** header. Enabling the header involves setting the `Dignition.https.sts.maxAge` system property to a non-zero value. By default, the header is disabled.

System Property	Description
<code>-Dignition.https.sts.maxAge</code>	Sets the maximum age of the Strict Transport Security header. The value should be set to an integer, representing a number of seconds the browser should remember that a site is only to be accessed using HTTPS. Setting this to a number less than zero will disable the header altogether.
<code>-Dignition.https.sts.includeSubDomains</code>	Applies the <code>includeSubDomains</code> parameter. Expects a boolean value. If omitted, defaults to false.
<code>-Dignition.https.sts.preload</code>	New in 8.1.5 This parameter allows websites to tell the browser to only connect using HTTPS. It can be set to <code>true</code> or <code>false</code> to enable or disable the <code>preload</code> directive of the <code>Strict-Transport-Security</code> header. If omitted, defaults to false.

Example

```
wrapper.java.additional.1=-Dignition.https.sts.maxAge=31536000
wrapper.java.additional.2=-Dignition.https.sts.includeSubDomains=false
wrapper.java.additional.3=-Dignition.https.sts.preload=true
```

Referrer Policy

The following system properties interact with the **Referrer-Policy** header. By default, the header is enabled with a value of `strict-origin-when-cross-origin`.

System Property	Description
<code>-Dignition.http.header.referrer_policy.enabled</code>	Allows you to enable or disable the Referrer-Policy header. Expects a <code>true</code> or <code>false</code> value.
<code>-Dignition.http.header.referrer_policy.value</code>	Sets the value of the Referrer-Policy.

Example

```
wrapper.java.additional.1=-Dignition.http.header.referrer_policy.enabled=true
wrapper.java.additional.2=-Dignition.http.header.referrer_policy.value=strict-origin-when-cross-origin
```

X Content Type Options

The following system properties interact with the **X-Content-Type-Options** header. By default, the header is enabled with a value of `nosniff`.

System Property	Description
<code>-Dignition.http.header.x_content_type_options.enabled</code>	Allows you to enable or disable the X-Content-Type-Options header. Expects a true or false value.
<code>-Dignition.http.header.x_content_type_options.value</code>	Determines the value of the X-Content-Type-Options.

Example

```
wrapper.java.additional.1=-
Dignition.http.header.x_content_type_options.enabled=true
wrapper.java.additional.2=-
Dignition.http.header.x_content_type_options.value=nosniff
```

X Frame Options

The following system properties interact with the **X-Frame-Options** header. By default, the header is enabled with a value of `SAMEORIGIN`.

NOTE

If you're embedding a Perspective session inside an iframe from another origin and it stops working after upgrade, then you may need to modify the X-Frame-Options header.

System Property	Description
<code>-Dignition.http.header.x_frame_options.enabled</code>	Enables or disables the X-Frame-Options header.
<code>-Dignition.http.header.x_frame_options.value</code>	Determines the value of the X-Frame-Options header.

Example

```
wrapper.java.additional.1=-Dignition.http.header.x_frame_options.enabled=true
wrapper.java.additional.2=-Dignition.http.header.x_frame_options.value=SAMEORIGIN
```

X XSS Protection

The following system properties interact with the X-XSS-Protection header. By default, the header is enabled with a value of `1; mode=block`.

System Property	Description
<code>-Dignition.http.header.x_xss_protection.enabled</code>	Enables or disables the X-XSS-Protection header.
<code>-Dignition.http.header.x_xss_protection.value</code>	Determines the value of the X-XSS-Protection header.

Example

```
wrapper.java.additional.1=-Dignition.http.header.x_xss_protection.enabled=true
wrapper.java.additional.2=-Dignition.http.header.x_xss_protection.value=1;
mode=block
```

Thread Pool Counts

In some testing and troubleshooting scenarios, it may be required to increase thread pool counts from their default settings.

CAUTION

While there is no limit on how high thread pool counts can be increased, be aware that resource usage such as CPU and RAM will also increase.

Tag Value Change Scripts

The following system properties allow you to increase the tag value change script thread pool count and queue size. By default, the thread pool count is set to 3 and the queue size is set to 5.

System Property	Description
<code>-Dignition.tags.scriptthreads</code>	Sets the tag event script thread count. The value should be an integer, representing the thread count.
<code>-Dignition.tags.scriptqueuemaxsize</code>	Sets the tag event script queue size. The value should be an integer, representing the queue size.

Example

```
wrapper.java.additional.1=-Dignition.tags.scriptthreads=5
wrapper.java.additional.2=-Dignition.tags.scriptqueuemaxsize=10
```

Query Tags

The following system property allows you to increase the thread pool count of `ignition.tags.db_tag_threads`. By default, the thread pool count is set to 3.

System Property	Description
<code>-Dignition.tags.db_tag_threads</code>	Sets the Query tag thread pool count. The value should be an integer, representing the thread pool count.

Example

```
wrapper.java.additional.1=-Dignition.tags.db_tag_threads=10
```

Expression Tags

The following system property allows you to increase the thread pool count of `ignition.tags.expressionthreads`. By default, the thread pool count is set to 3.

System Property	Description
<code>-Dignition.tags.expressionthreads</code>	Sets the Expression tag thread pool count. The value should be an integer, representing the thread pool count.

Example

```
wrapper.java.additional.1=-Dignition.tags.expressionthreads=10
```

Transaction Groups

The following system property allows you to increase the thread pool count of `ignition.sqlbridge.maxthreads`. By default, the thread pool count is set to 5.

System Property	Description
<code>-Dignition.sqlbridge.maxthreads</code>	Sets the thread pool count for transaction groups. The value should be an integer, representing the thread pool count.

Example

```
wrapper.java.additional.1=-Dignition.sqlbridge.maxthreads=10
```

HTTP Client Manager

New in 8.1.12

The following system properties interact with the platform's HTTP Client. These settings allow you to configure how the IdP system and Perspective's HTTP Bindings parse cookies, handle idle HTTP connections, and respect proxy settings.

System Property	Description
<code>-Dignition.http.client.manager.cookieSpec</code>	Controls the cookie parsing behavior of the HTTP Client. This system property may take one of the following values: <code>default</code>: Default cookie specification that picks up the best matching cookie policy based on the format of cookies sent with the HTTP response. <code>netscape</code>: This CookieSpec implementation conforms to the original draft specification published by Netscape Communications. It should be avoided unless absolutely necessary for compatibility with legacy applications. <code>standard</code>: Standard CookieSpec implementation that enforces a more relaxed interpretation of the HTTP state management specification (RFC 6265, section 5) for interoperability with existing servers that

System Property	Description
	do not conform to the well-behaved profile (RFC 6265, section 4). <code>standard-strict</code>: Standard CookieSpec implementation that enforces syntax and semantics of the well-behaved profile of the HTTP state management specification (RFC 6265, section 4).
<code>-Dignition.http.client.manager.socket.keepalive</code>	When set to <code>true</code>, HTTP connections issued from the HTTP Client will set the <code>SO_KEEPALIVE</code> flag on the underlying socket, enabling keepalive. Enabling keepalive maintains a connection between a client and the server, reducing the number of TCP and SSL/TLS connection requests. You must also enable and configure keepalive settings on the OS running the Gateway. See KeepAlive Configuration.
<code>-Dignition.http.client.manager.pool.idleConnectionCheckInterval</code>	An integer representing the duration in seconds between each check for

System Property	Description
	idle connections. This value is ignored when <code>maxIdleConnectionTime</code> is less than or equal to zero. If this value is undefined or less than or equal to zero, then the interval is set to match <code>maxIdleConnectionTime</code> .
<code>-Dignition.http.client.manager.pool.maxIdleConnectionTime</code>	An integer representing the number of seconds that connections in the HTTP connection pool may be idle (i.e. no data is sent or received on the socket) before it is evicted and closed. If set to any number less than or equal to zero or undefined, idle connection eviction will be disabled.
<code>-Dignition.http.client.manager.proxy.enabled</code>	If set to true, the HTTP Client will respect the JVM's system default proxy settings. Default value if undefined is false.

Example

```
wrapper.java.additional.1=-Dignition.http.client.manager.cookieSpec=standard-strict
wrapper.java.additional.2=-Dignition.http.client.manager.socket.keepalive=true
wrapper.java.additional.3=-Dignition.http.client.manager.pool.idleConnectionCheckInterval=60
wrapper.java.additional.4=-Dignition.http.client.manager.pool.maxIdleConnectionTime=300
wrapper.java.additional.5=-Dignition.http.client.manager.proxy.enabled=true
```

KeepAlive Configuration Examples

KeepAlive Configuration for Windows

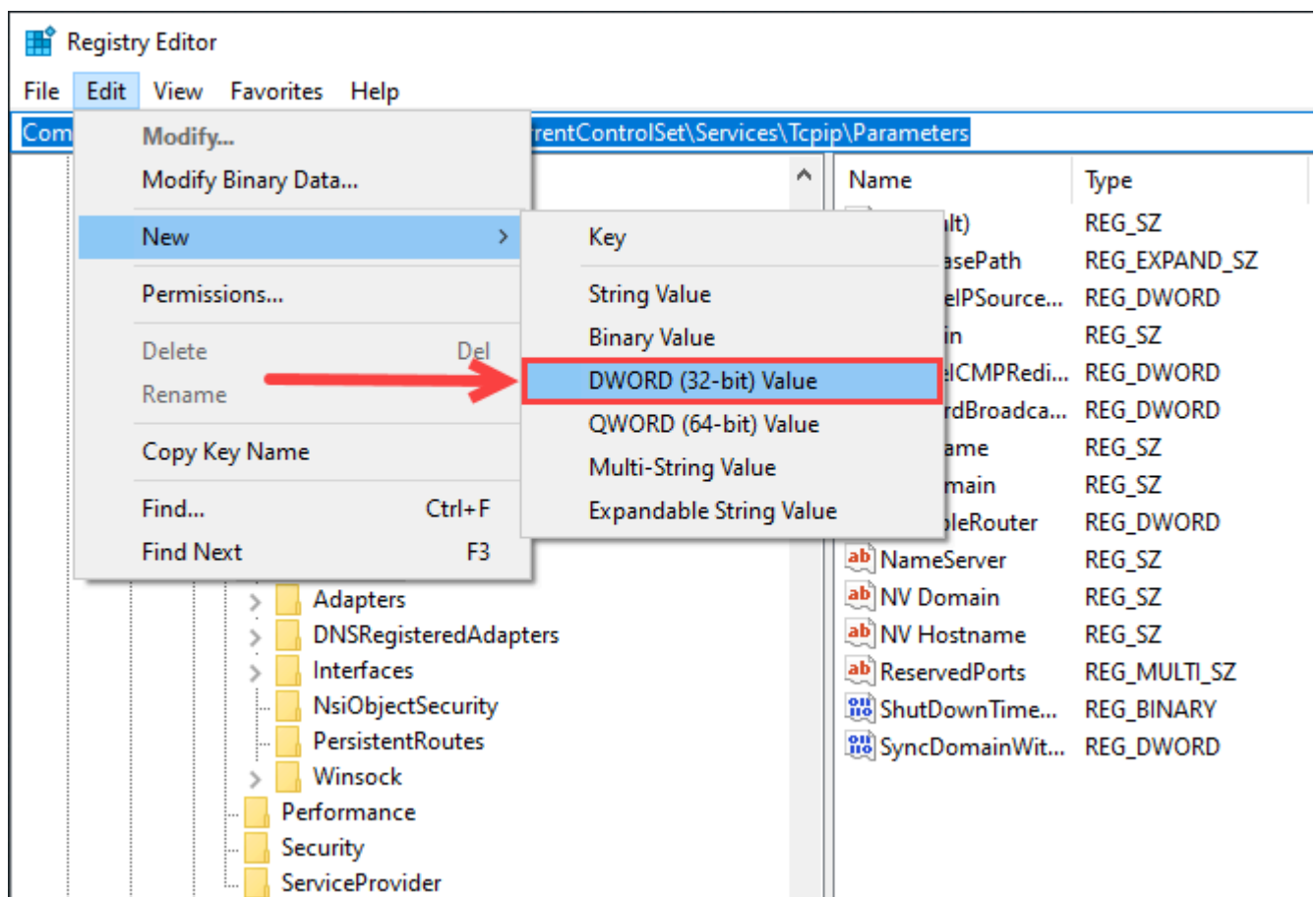
 CAUTION

The following example involves modifying a registry value on Windows. incorrect registry configurations can cause serious problems and impact your server's performance. For information on how to back up and restore the registry in Windows, [click here](#).

In this example, we'll add the following parameters to our registry:

Name	Data Type	Range	Default Value
KeepAliveTime	REG_DWORD	0x1–0xFFFFFFFF (milliseconds)	0x6DDD00 (7,200,000 milliseconds = 2 hours)
KeepAliveInterval	REG_DWORD	0x1–0xFFFFFFFF (milliseconds)	0x3E8 (1,000 milliseconds = 1 second)

1. Navigate to \HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters.
2. If parameters for \KeepAliveTime and \KeepAliveInterval do not exist, create them.
3. Click **Edit>New>DWORD (32-bit) Value**.
4. Open the Registry Editor.



5. Enter the values listed on the table above this example (KeepAliveTime and KeepAliveInterval)
6. Exit the Registry Editor.
7. Restart the computer for changes to take effect.

KeepAlive Configuration for Linux

For Linux systems, there are three parameters to change:

Name	Default Value
tcp_keepalive_time	7200 (seconds)
tcp_keepalive_intvl	75 (seconds)
tcp_keepalive_probes	9 (number of probes)

1. Edit /etc/sysctl.conf:


```
# vi /etc/sysctl.conf
```
2. Add the following settings:
 - `net.ipv4.tcp_keepalive_time = #newvalue`
 - `net.ipv4.tcp_keepalive_intvl = #newvalue`

- `net.ipv4.tcp_keepalive_probes = #newvalue`

i NOTE

The following system property will configure Ignition to use IPv4 sockets only.

System Property	Description
<code>-Djava.net.preferIPv4Stack</code>	Configures Ignition to use only IPv4 sockets when this property is set to true.

Maximum Form Size

i NOTE

By default, the maximum size is kept low on the default installation to prevent DoS attacks. We do not recommend changing this, especially so if the Gateway isn't in a closed network.

This parameter allows you limit the amount of data that can be posted back from a browser or other client, to the Gateway. Generally, this is useful to increase the maximum size for **Web Dev** POST requests. More information about this parameter can be found in [Jetty's documentation](#).

Example

```
wrapper.java.additional.1=-  
Dorg.eclipse.jetty.server.Request.maxFormContentSize=2000000
```

Internal Database

Connection Timeout

New in 8.1.16

The default maximum time that the Gateway will wait for a connection to an internal database is 30 seconds. This parameter allows you to override this setting and increase the maximum wait time.

Example

```
wrapper.java.additional.1=-Dignition.internalDbMaxWait=60000
```

Internal Database VACUUM Operation

New in 8.1.23

A VACUUM operation is executed against the internal database upon Gateway startup by default. This operation reclaims unused disk space, trimming down on both the time it takes for auto-backups to complete, as well as the overall size of the internal database itself. This action can be disabled by using the following parameter.

Example

```
wrapper.java.additional.1=-Dignition.skipConfigDbVacuum=true
```

Internal Database Legacy Tag Cleanup

New in 8.1.23

Upon Gateway startup, you can conduct a cleanup task on old 7.X.X SQLTAG tables. This helps to reduce bloated internal databases, save disk space, and potentially improve Gateway performance. Using the following parameter will opt the Gateway in for performing this cleanup task.

Example

```
wrapper.java.additional.1=-Dignition.tags.cleanupLegacyTagTables=true
```

Internal Database Project Table Cleanup

New in 8.1.23

Upon Gateway startup, a cleanup task is conducted on old 7.X.X project tables. This helps to trim down internal databases, providing more space for other operations, and may improve Gateway performance. The following parameter will allow you to skip this cleanup operation.

Example

```
wrapper.java.additional.1=-Dignition.projects.skipProjectRecordTableCleanup=true
```

Hosted Launcher Installers

Normally, each Ignition Gateway includes files for the various launchers. When you download a launcher from a Gateway, it simply streams its local launcher files. However, you can override this behavior, causing the Gateway to ignore its local launcher files and instead download launchers from

the internet. This is an uncommon configuration for most cases, as it was devised primarily to aid with systems where memory limitations are strict (such as physical devices that include preinstalled Ignition Gateways).

Enabling Hosted Launchers

The following parameter (and value) enables the use of hosted launchers. While enabled, the Gateway will only ever attempt to retrieve the hosted launchers, meaning it will ignore the local launcher files.

Example

```
wrapper.java.additional.1=-Dignition.hostedLaunchers=true
```

Hosted Launcher Version

By default, enabling Hosted Launchers will cause the Gateway to retrieve launchers version appropriate launchers: 8.0.0 launchers for 8.0.0 Gateways, 8.1.0 launchers for 8.1.0 Gateways, etc.

The parameter below can be used to explicitly state which launcher version to retrieve. Generally, this is not recommended, but can potentially be useful if you're looking for a specific launcher version. It requires that `-Dignition.hostedLaunchers` is set to "true".

Example

```
wrapper.java.additional.1=-Dignition.hostedLauncherVersion=8.1.0
```

Automatic Thread Dump Sample Rate

New in 8.1.13

The following parameter can be used to override the default sample rate of 1 second for Automatic Thread Dumps. The time is set in milliseconds, and the minimum time setting is 500ms.

Example

```
wrapper.java.additional.1=-Dignition.automaticThreadDump.sampleRate=5000
```

Ignition Edition

You can specify which edition of Ignition a Gateway should be set to with the parameter demonstrated below.

Example

```
wrapper.java.additional.1=-Dedition=standard
```

CAUTION

Ignition Gateway Licenses are matched to a specific edition. Changing the edition of an Ignition Gateway that is already licensed can result in the license becoming invalidated. It's recommended that you unactivate a license on a gateway before changing its edition.

Values

- Standard Ignition - `standard`
- Ignition Edge - `edge`
- Ignition Maker - `maker`

Loading Unsigned Modules - Developer Mode

Normally, an Ignition Gateway will not allow unsigned module to be installed. This restriction can be disabled with the flag below. This is normally done in cases where a user is developing a custom module, and wants to install it without having the module signed.

Example

```
wrapper.java.additional.1=-Dignition.allowunsignedmodules=true
```

Project Directory Scan Rate

This setting dictates how often the Ignition Gateway will scan its project directory for changes. As of Ignition 8.1.0 this was changed to a 300 second rate (from a 10 seconds rate), but it can be configured. This is useful in cases where project files are being modified from external sources and the gateway needs to be aware of these changes in a more timely manner. The argument below expects an integer, which represents a number of seconds.

Example

```
wrapper.java.additional.1=-Dignition.projects.scanFrequency=100
```

Ignored Project Directory Files

New in 8.1.15

This setting allows users to specify a list of folders in the project directory that the Ignition Gateway will ignore when scanning for changes. By default, the Gateway will ignore changes to any files located within folders with the name ".git", ".svn", or ".hg". The argument below expects a list of additional folders to ignore, using the ':' character as a file separator.

Example

```
wrapper.java.additional.1=-Dignition.projects.ignoredFiles=Folder1:Folder2:Folder3
```

CAUTION

By specifying a folder to ignore, note that any folder within the data/projects folder with the same name will be excluded, including any projects with that name. For example, if you exclude a folder called "test" and your project name is also "test" the project will be excluded as well.

Redundant Alarm Runtime Limit

In redundant systems, both alarm event and alarm shelf items wait in separate queues on the active node before they're transferred to the inactive node. Normally the active node sends recently changed alarm events and shelf events to the inactive node as they occur. However if the queue becomes full the active node will pause this behavior and send a full transfer to the inactive node, ensuring that the state of events on the inactive node is accurate.

System Property	Description
<code>-Dalarm.redundancy.runtimeupdates.max</code>	Determines the maximum size of the alarm events queue. Defaults to 2,000,000 if omitted.
<code>-Dalarmshelf.redundancy.runtimeupdates.max</code>	Determines the maximum size of the alarm shelf queue. Defaults to 2,000,000 if omitted.

Example

```
wrapper.java.additional.1=-Dalarm.redundancy.runtimeupdates.max=1000000  
wrapper.java.additional.2=-Dalarmshelf.redundancy.runtimeupdates.max=1000000
```

Web Server

The following parameters are used to configure Ignition's default settings for web requests that are not configured in some other, more specific manner, such as [Perspective Route Handling](#). These settings usually do not need to be modified, but a higher value may help mitigate issues on very busy systems.

System Property	Description
<code>-Dignition.routes.defaultConcurrency</code>	Determines the number of concurrent sessions allowed to acquire a lock per webserver route. The lock must be acquired within two seconds of an incoming request, or the request will return an error. Default is 5.
<code>-Dignition.routes.defaultRequestTimeout=xxxx</code>	New in 8.1.38 The route group concurrency timeout value. Increasing this value may help mitigate timeout and error 503 issues. Default is 2000 milliseconds (2 seconds)

Example

```
wrapper.java.additional.1=-Dignition.routes.defaultConcurrency=5  
wrapper.java.additional.2=-Dignition.routes.defaultRequestTimeout=2000
```

JX Browser User Directories

New in 8.1.27

These parameters allow users to specify a direct user data directory other than the default directory. This change will apply to both Designers, Perspective sessions, and the Vision Web Browser Component, regardless of the user's operating system.

System Property	Description
<code>-Dignition.jxBrowser.userDataDir.browser</code>	Specifies the user data directory to use for the Vision Web Browser Component.
<code>-Dignition.jxBrowser.userDataDir.perspective</code>	Specifies the user data directory to use for a Perspective session.

Example

```
wrapper.java.additional.1=-
Dignition.jxBrowser.userDataDir.browser=C:\Users\ignitionUser\AppData\Local\Temp\2\Us
wrapper.java.additional.2=-
Dignition.jxBrowser.userDataDir.perspective=C:\Users\ignitionUser\AppData\Local\Temp\
```

Gateway Diagnostic Bundle Export Memory Dump

The following parameter enables a memory dump as part of the [diagnostic bundle export feature](#).

System Property	Description
<code>-Dignition.diagnostic-bundle.enable-memory-dump</code>	Enables or disables a memory dump as part of the diagnostic bundle. Expects either "true" or "false". Defaults to "false".

Example

```
wrapper.java.additional.1=-Dignition.diagnostic-bundle.enable-memory-dump=true
```

SMTP Email Notifications

New in 8.1.28

The following parameter allows users to define send behavior when an invalid notification email is present for the [system.net.sendEmail\(\)](#) function.

System Property	Description
 <pre>Dignition.smtp.sendpartial</pre>	If the value is set to true, email notifications are sent to the valid emails provided for that user and an exception is logged for the failed emails. If set to false, no emails are sent if any of the provided emails are invalid. The default value is set to true.

Example

```
wrapper.java.additional.1=Dignition.smtp.sendpartial=true
```

Restore File TTL

New in 8.1.43

The following parameter configures the time-to-live (TTL) for restore files used during backup and restore processes, including redundancy. This setting determines how long restore files are retained before being automatically deleted.

System Property	Description
 <pre>Dignition.redundancy.restoreFileTTL</pre>	Sets the TTL for restore files used in both redundancy and general backup/restore operations. The value should be an integer, representing the number of minutes the file will be retained. The default value is 20 minutes.

Example

```
wrapper.java.additional.1=-Dignition.redundancy.restoreFileTTL=20
```

Gateway Network Parameters

Proxy Depth

By default, Gateway remote providers can only "hop" two Gateways away. For example, assume the numbers below represent Gateways, which are all connected over a Gateway Network.

Normally, Gateway 1 only has access to Gateway 2 and 3. Gateway 4 is too far away. However this can be changed by adding the following to Gateway 1's configuration file:

Example

```
wrapper.java.additional.1=-Dignition.gan.maxproxydepth=3
```

NOTE

The [Allow Proxying setting](#) must be enabled for requests to be forwarded in this manner.

Ping Thread Count

New in 8.1.34

The following system property allows you to increase the ping thread pool count to reduce the connection reconnect time if multiple outgoing connections are faulted. By default, the count is set to 3 and with a 10 second timeout. For example, if no adjustments are made and there are 15 faulted outgoing connections, there would be about a minute delay for new incoming connections.

System Property	Description
 Dignition.metro.pingexecutor.threads	Sets the ping thread count. The value must be an integer.

Example

```
wrapper.java.additional.1=-Dignition.metro.pingexecutor.threads=10
```

Gateway Network Queue Timeout

New in 8.1.22

The Long Wait Queue timeout and Proxy Queue timeout can be adjusted if necessary for slower connections. These settings will mostly apply to EAM, but can also apply for more general GAN instances such as a GWBK transferring from a redundant master to a backup.

System Property	Description
<code>-Dmetro.queues.proxyQueue.timeoutSecs</code>	By default, the Proxy Queue will timeout after 60 minutes. This can be adjusted by setting the system property shown.
<code>-Dmetro.queues.longWaitQueue.timeoutSecs</code>	This parameter will adjust the Long Wait Queue timeout setting. The Long Wait Queue defaults to handle messages that take up to an hour to deliver if left unchanged. Note: The longWaitQueue system property was implemented without the "e" in timeout. Be sure to spell it as shown in the System Property field when using.

EAM Timeout

The following system properties interact with EAM. These settings allow you to configure the timeout settings for various possible actions in EAM.

System Property	Description
- Dignition.eam.task.collectBackup.transferTimeout	New in 8.1.17 By default, the EAM Collect Backup task will timeout if the Agent takes longer than 60 minutes to transfer the backup to the Controller over the Gateway Network. This setting can be adjusted by adding the aforementioned parameter, which expects an integer value in milliseconds.
-Dignition.eam.freeSpace.timeout.millis	New in 8.1.18 This parameter will allow for adjusting the timeout setting that checks if there is sufficient free space to perform the EAM Controller's save operations. Expects an integer value in milliseconds. Default value is 10,000 milliseconds (10 seconds).
-Dws.handoff.timeout	New in 8.1.29 By default, EAM tasks that need to send a large file through a proxy Gateway timeout after 60 minutes. This value can be adjusted by setting the handoff timeout system property integer value, in seconds, on the proxy Gateway.

HTTPS Settings

The following system properties will either include or exclude specified cipher suites for the Gateway Network port 8060. For including or excluding cipher suites for port 8043, refer the the [Web Server Settings](#) page.

System Property	Description
<code>-Dciphers</code>	Lists included cipher suites for connections using SSL/TLS for port 8060.
<code>-DexcludedCiphers</code>	Lists excluded cipher suites for connections using SSL/TLS for port 8060.

Example

```
# Make sure to separate listed cipher suites with a comma.  
  
wrapper.java.additional.X=-Dciphers=TLS_DHE_DSS_WITH_AES_128_CBC_SHA256,TLS_DHE_DSS_W  
wrapper.java.additional.X=-  
DexcludedCiphers=TLS_AES_128_GCM_SHA256,TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_
```

Perspective Parameters

Gateway Configuration

Session Caching

Perspective Sessions cache assets (our CSS, JS, fonts, and images) to reduce load times after the first time opening the project. Sessions will cache assets, but some browsers will attempt to re-download assets on refresh. This is mostly controlled by the browser instead of Ignition so there are some important differences between browsers.

Chrome and Safari handle browser cache differently than Edge and Firefox when it comes to hitting the 'refresh' or 'reload' button. Edge and Firefox both seem to send a request for new copies of the resources with header of max-age=0 instead of pulling from the cache. Navigating via a 'back' button or entering in the URL bar and pressing 'enter' will load from cache. This seems to simply be a design difference by these two browsers, in which both are working as intended.

Caching can be disabled via the following flag:

```
wrapper.java.additional.1=-Dperspective.gateway.disableClientCaching=true
```

Alternatively the cacheControl flag can be used to modify aspects of the cache. You can learn more about cache control from the [MDN documentation](#). The example below lists some default values:

```
wrapper.java.additional.1=-Dperspective.gateway.cacheControl=public,max-age=2419400,no-transform
```

Route Handling

New in 8.1.8

The following parameters are used to modify route handling for Perspective Sessions. Most systems will not need to modify these parameters. However increasing the values of these parameters may be helpful in cases where large project updates are sent out to active Perspective Sessions over a spotty network. A low timeout and concurrency in that scenario may result in browser timeouts. If these values are not set, Perspective will use the global routing values instead. See the [Web Server section](#) for more details on global routing.

System Property	Description
<code>-Dperspective.fetch-concurrency</code>	Determines the number of concurrency sessions. Defaults to 25.
<code>-Dperspective.fetch-timeout</code>	Determines the timeout period when receiving project updates. Defaults to 5000 milliseconds.

Example

```
wrapper.java.additional.1=-Dperspective.fetch-concurrency=25
wrapper.java.additional.2=-Dperspective.fetch-timeout=5000
```

Web Socket Compression

You can remove the compression header for perspective web socket connections.

This parameter is generally only used in cases where a Microsoft IIS based firewall is acting like a reverse proxy, and needs to be able to rewrite secure web socket messages.

Example

```
wrapper.java.additional.1=-Dperspective.ws.disablePermessageDeflateExtension=true
```

Web Socket Idle Time

New in 8.1.33

The following parameters allow customization of Perspective session failover redirect timing. Tuning these timeout values makes it possible to detect when a connection is down in a few seconds rather than minutes, and a failover to the backup Gateway can be triggered more quickly. If making changes, be sure `idleCheckIntervalMs` is less than `maxIdleTimeMs`.

CAUTION

Do not set these values too low on busy Gateways as they may take time to send messages over the websocket channel because of the amount of tasks, not connection issues.

System Property	Description
<code>-Dperspective.client.websocketChannel.maxIdleTimeMs</code>	Specifies the maximum idle time to detect a connection to a Gateway in milliseconds. Default value is 120000.
<code>-Dperspective.client.websocketChannel.idleCheckIntervalMs</code>	Specifies the interval value to trigger a failover to a backup Gateway in milliseconds. Default value is 30000.

Example

```
wrapper.java.additional.1=-Dperspective.client.websocketChannel.maxIdleTimeMs=30000
wrapper.java.additional.1=-Dperspective.client.websocketChannel.idleCheckIntervalMs=10000
```

Web Socket Message Size

New in 8.1.12

The following parameter can be used to specify the maximum web socket message size. Most systems will not need to modify this parameter. Increasing the maximum message size may be helpful if your project includes very large Perspective views.

System Property	Description
<code>-Dperspective.websocket.max-message-size</code>	Specifies the maximum message size in KB. Default value is 2048.

Example

```
wrapper.java.additional.1=-Dperspective.websocket.max-message-size=2048
```

Thread Pool Restrictions

New in 8.1.12

Perspective has two primary thread pools:

- A worker thread pool (used for long-running, "blocking" work)
- A queue thread pool (used for short-duration internal tasks)

By default, each of these pools can have an unlimited number of threads with a 60 second keep-alive. The following parameters can be used to fix the number of threads or change the keep-alive period. This can be useful in troubleshooting scenarios when attempting to limit the number of threads.

System Property (Worker)	Description
<code>-Dperspective.threadpool.perspective-worker.keep-alive</code>	Specifies the keep-alive period. Expects an integer, which represents an amount of seconds. When omitted, defaults to 60 seconds. Note that this setting is ignore if pool-size (listed below) is set.
<code>-Dperspective.threadpool.perspective-worker.pool-size</code>	Specifies the maximum number of workers as an integer. When omitted, defaults to unlimited.

Example

```
wrapper.java.additional.1=-Dperspective.threadpool.perspective-worker.keep-alive=40

# These settings are mutually exclusive, so setting both will result in the keep-alive being ignored
# wrapper.java.additional.1=-Dperspective.threadpool.perspective-worker.pool-size=100
```

System Property (Queue)	Description
<code>-Dperspective.threadpool.perspective-queue.keep-alive</code>	Specifies the keep-alive period. Expects an integer, which represents an amount of seconds. When omitted, defaults to 60 seconds. Note that this setting is ignore if pool-size (listed below) is set.
<code>-Dperspective.threadpool.perspective-queue.pool-size</code>	Specifies the maximum number of workers as an integer. When omitted, defaults to unlimited.

Example

```
wrapper.java.additional.1=-Dperspective.threadpool.perspective-queue.keep-alive=40

# These settings are mutually exclusive, so setting both will result in the keep-
# alive being ignored
# wrapper.java.additional.1=-Dperspective.threadpool.perspective-queue.pool-
# size=100
```

Cache and Share

New in 8.1.12

Certain bindings and queries in Perspective will often pull data to see if any values have changed. You can influence whether Perspective pulls in cached data or issues a new query by setting the timeout threshold before Perspective executes a new query. In other words, if Perspective fetches data that is within that time frame, the cached value will be used instead of a new, queried value. This timeout value can be modified with the following system property:

System Property	Description
<code>-Dperspective.value-cache.max-age</code>	Controls how long Cache and Share will keep cached values alive for before Perspective polls for new data. Default is 1 minute.

Example

```
# "####" is the timeout period for using cached data.
wrapper.java.additional.1=-Dperspective.value-cache.max-age=####
```

Designer Launcher Configuration

Designer Save Timeout

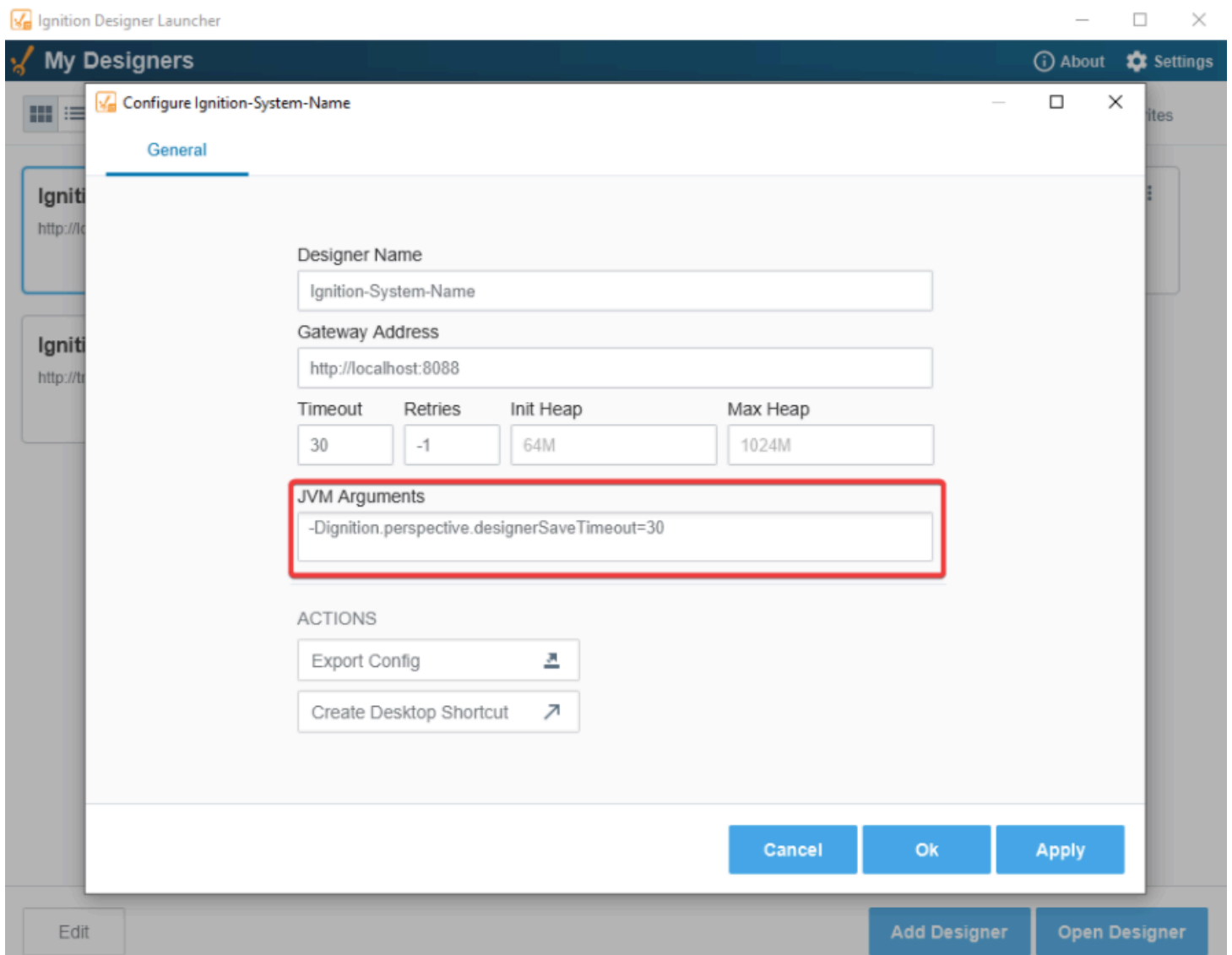
New in 8.1.45

The following parameter configures the Designer save timeout when working with a Perspective Session. This parameter must be entered in the **Designer Launcher JVM Arguments** field. To access this field, click the three dots menu for the project in the Designer Launcher and select Manage.

System Property	Description
- Dignition.perspective.designerSaveTimeout	Specifies the time (in seconds) the Designer will wait for an embedded browser frame to return results. The default timeout is 60 seconds.

Example

```
-Dignition.perspective.designerSaveTimeout=30
```



Synthetica Screen Share Warnings

New in 8.1.48

When running a Designer in Linux Wayland, you may receive screen sharing warnings that can cause the Designer to stop working. These warnings can be prevented by adding a custom Synthetica XML file to your Designer Launcher JVM arguments. Your XML file should look like the following:

test

```
<synth>
  <defaultsProperty key="Synthetica.popupRobot.enabled" type="boolean"
value="false"/>
</synth>
```

System Property	Description
<code>-Dignition.synthetic.customXml</code>	Applies the referenced XML file to disable the <code>Synthetic.popupRobot.enabled</code> UIManager key, which causes the warnings to appear.

Example

```
-Dignition.synthetic.customXml=C:\Users\username\Desktop\test.xml
```

Vision Client Parameters

Gateway Configuration File Parameters

The following parameters should be placed in the Gateway's configuration file

Suppressing Error Message Details

New in 8.1.10

By default, error messages in the Designer and Client report a stack trace, which provides granular information about the source of the error. However in some cases, these stack traces may include sensitive information from the Gateway. You can restrict the amount of information presented in these error messages with the following parameter:

Example

```
wrapper.java.additional.#=-Dignition.xmlrpc.errorDetailSuppression=%Value%
```

Where %Value% is one of the following values:

Value	Description
NONE	Errors are not suppressed at all. This is functionally the same as not specifying the parameter at all. (default behavior)
CLIENT	Error messages for client sessions will be suppressed, but not designer sessions.
ALL	Error messages for both clients and the designer will be suppressed.

Route Handling

New in 8.1.24

The following parameters are used to modify route handling for Vision Clients. Changing the values for these parameters is not necessary, but increasing the values may help improve Vision Client launching in extremely busy systems, resulting in fewer browser timeouts.

System Property	Description
<code>-Dvision.fetch-concurrency</code>	Determines the number of concurrency Clients. Defaults to 25,
<code>-Dvision.fetch-timeout</code>	Determines the timeout period when receiving project updates. Defaults to 5000 milliseconds.

Vision Launcher File Parameters

The following parameters should be placed in the [Vision Client Launcher's configuration file](#)

Timeout Thresholds

New in 8.1.27

The following parameters will modify Vision connection default wait times. These timeout settings are available to change under [Vision Project Property > Timing](#), but changes inside the Designer will only take effect after the project is downloaded. Using a timeout threshold parameter will override the default settings to increase maximum wait times immediately.

System Property	Description
<code>-Dignition.gateway.connectTimeout</code>	The maximum amount of time to wait for connections to the Gateway to be established. Specified in milliseconds. Default is 10,000.
<code>-Dignition.gateway.readTimeout</code>	The maximum amount of time for socket connection to the Gateway to remain open. Specified in milliseconds. Default is 60,000.
<code>-Dignition.gateway.opcBrowseTimeout</code>	Maximum amount of time, in milliseconds, to wait for the response to a request. Default is 120,000.

Long Press Delay

New in 8.1.28

The following system property allows the long press delay to be configured per client for the [Vision Touchscreen Keyboard](#).

System Property	Description
 <code>Dignition.touchscreen.longPressDelay</code>	The amount of time in milliseconds needed for a user to press and hold before a long press action is enabled. Default is 1000. A value less than or equal to 0 will disable long press behavior entirely.

Keyboard Font Scaling

New in 8.1.34

The following parameters change the scaling and size of the **Vision Touchscreen Keyboard**.

System Property	Description
 <code>-Dignition.touchscreen.keyboardWidth</code>	The percentage of the client window the touch screen keyboard displays. You can also set the "Touch Screen Keyboard Width" in Vision's Project Properties for all machines running the Vision project. Default is 75%.
 <code>Dignition.touchscreen.keyboardFontSize</code>	Customize touchscreen keyboard font size. You can also set the "Touch Screen Keyboard Font Size" in Vision's Project Properties for all machines running the Vision project. Default is 30.

Synthetica Screen Share Warnings

New in 8.1.48

When running a Designer in Linux Wayland, you may receive screen sharing warnings that can cause the Designer to stop working. These warnings can be prevented by adding a custom Synthetica XML file to your Designer Launcher JVM arguments. Your XML file should look like the following:

```
test

<synth>
  <defaultsProperty key="Synthetica.popupRobot.enabled" type="boolean"
value="false"/>
</synth>
```

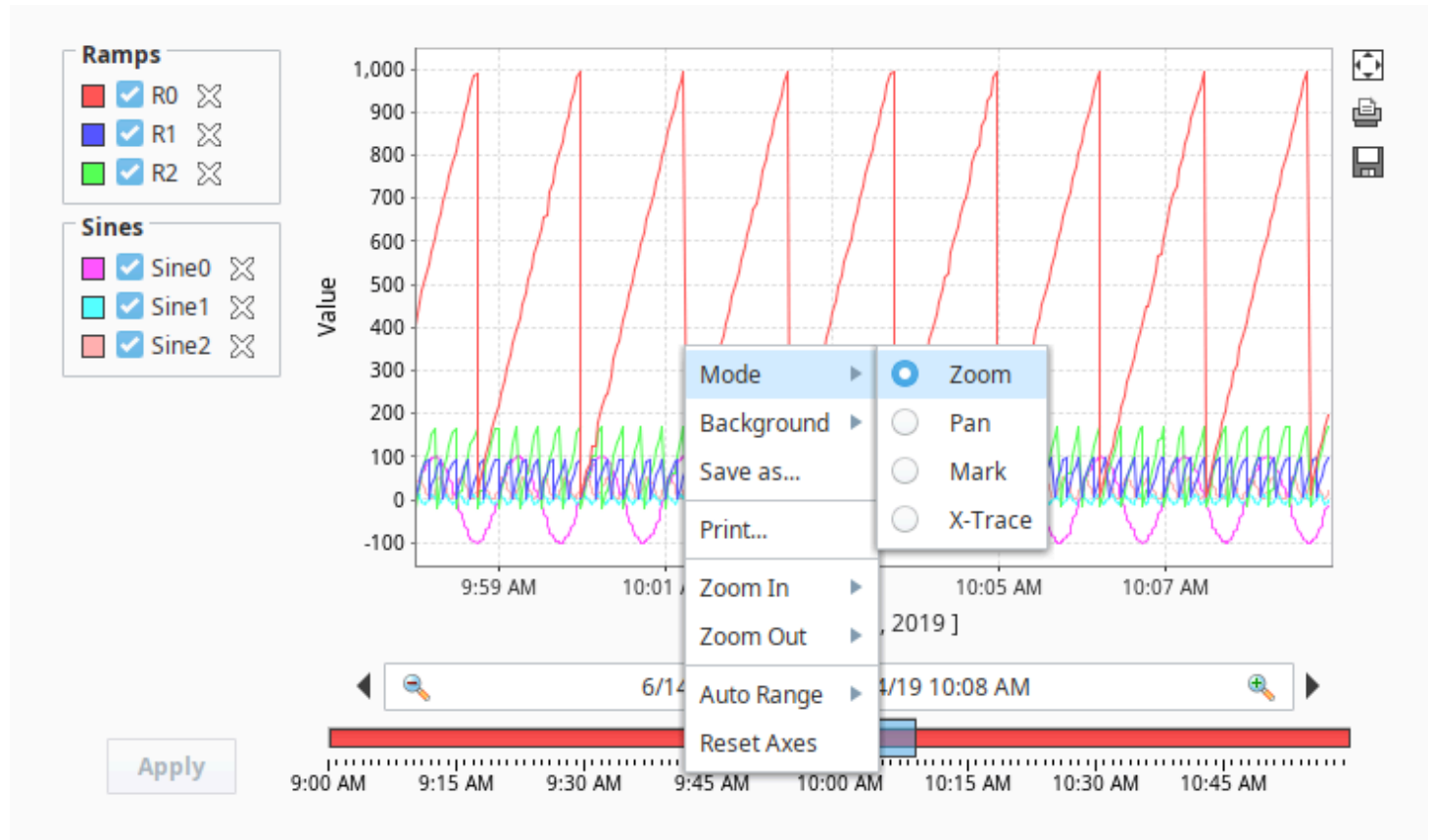
System Property	Description
<code>-Dignition.synthetic.customXml</code>	Applies the referenced XML file to disable the <code>Synthetic.popupRobot.enabled</code> UIManager key, which causes the warnings to appear.

Example

```
-Dignition.synthetic.customXml=C:\Users\username\Desktop\test.xml
```

Charting - Right Click Menu

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



Right Click Menu Functions

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.

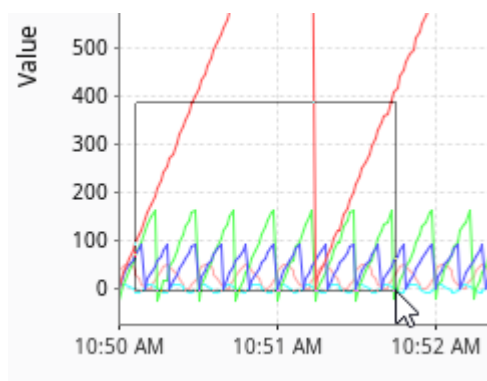
Item Name	Description
Mode	Specifies the functionality of a left-click on the chart. A list of available modes can be found on the Mode Options section.
Background	Allows the user to change the background color. Available options include Black and White .
Save as...	Provides a Save window that allows 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.

Item Name	Description
	• Range axis: Adjusts the viewable space on just the Range, or Y-axis.
Reset Axes	Adjusts the Axes back to their initial state.

Mode Options

Below is a description of the various options under the Mode property.

Zoom

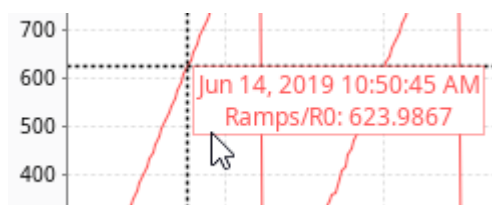


- **Zoom in:** Users can left-click and drag on the chart to zoom in or out. Dragging to the right creates a hollow rectangle. When the user releases the mouse button, the viewable space in the chart will zoom in to a rectangle.
- **Zoom out:** Left-click and dragging to the left will not create a rectangle, but releasing the mouse button will zoom back out.

Pan

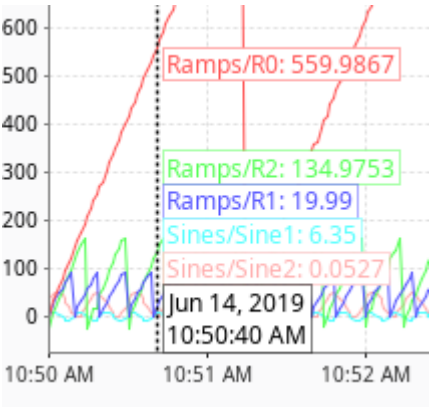
Allows users to **Pan** the chart horizontally or vertically by left-click and dragging. The **Pan** mode may be used after using the Zoom mode: doing so allows panning within the zoomed in view-space.

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.

Gateway Port Reference

Port Reference

An Ignition server uses many ports to manage information between itself and other Ignition servers, OPC servers, devices, and other services such as SMTP and ERP servers. The following table describes the default ports an Ignition server may use for communication. Most are configurable and some of these ports may be on a computer other than the one Ignition is installed on (that is, if MySQL is installed on a different computer, the Ignition server will not be using port 3306). This is not a complete list of TCP ports, and there are a few UDP and broadcast ports in the list. Ignition will potentially use other ports if you are connecting to additional services or devices. Setting ports may be possible for some of the following, but will depend on their respective software.

Ignition Ports

Port	Direction	Protocol	Configurable	Description
80	Incoming	TCP	Yes	Default port setting to access the Ignition Gateway when using Ignition Cloud Edition .
443	Incoming	TCP	Yes	Default SSL port setting to access the Ignition Gateway when using Ignition Cloud Edition .
1883	Incoming	TCP	No	MQTT Unencrypted. Outgoing port if using an external MQTT broker, Listen port on Ignition if using the Distributor module.
4096	Incoming	TCP	Yes	Legacy port for Ignition OPC UA server. Versions 7.9 and prior used this port for UA server communications. When upgrading from Ignition 7 to 8+, the legacy port will initially be used.
4445	Outgoing	UDP/broadcast	Yes	Default send port for multicast messages that makes the Gateway discoverable on a local network.
4446	Incoming	UDP/broadcast	Yes	Default receive port for multicast messages that makes the Gateway

Port	Direction	Protocol	Configurable	Description
				discoverable on a local network.
6501	Incoming	TCP	No	The port utilized by Vision's Local Client Fallback feature.
8043	Incoming	TCP	Yes	Default SSL port setting to access the Ignition Gateway .
8060	Incoming	TCP	Yes	Default SSL port setting for the Gateway Network (metroSSLPort in the Gateway XML file).
8088	Incoming	TCP	Yes	Default port setting to access the Ignition Gateway . Also the non SSL port for the Gateway Network.
8883	Incoming	TCP	No	MQTT SSL/TLS. Outgoing port if using an external MQTT broker, Listen port on Ignition if using the Distributor module.
17342	Incoming	UDP	No	Receive Port for SMS with Alarming.
17341	Outgoing	UDP	No	Send port for SMS with Alarming.
62541	Incoming	TCP	Yes	Default port for Ignition OPC UA server.

Other Common Software/Hardware Ports

 CAUTION

The information listed below has been included for user convenience when attempting to access specific services. All details should be verified by the relevant third party documentation, which may also identify other configuration settings not mentioned on this page that need to be modified to make the listed ports accessible.

Port	Direction	Protocol	Configurable	Description
102	Outgoing	TCP	No	Siemens Step 7 controller software.
135	Outgoing	TCP	No	Default port for DCOM communication (old OPCDA servers).
389	Outgoing	TCP	Yes	Default port for Active directory if this is being used.
465	Outgoing	TCP	No	SMTP protocol used if Alarming is configured.
502	Outgoing	TCP	Yes	Default Modbus port.
1433	Outgoing	TCP	Yes	Default MSSQL port.
1521	Outgoing	TCP	Yes	Default Oracle port.
2222	Outgoing	TCP	No	Allen Bradley Drivers (Ethernet/IP I/O DMA).
3050	Outgoing	TCP	Yes	Default Firebirdsqli port.
3306	Outgoing	TCP	Yes	Default MySQL port.
5060	Incoming	TCP/SIP	No	Send Voice Notification to SIP server .
5432	Outgoing	TCP	Yes	Default Postgresqli port.
8000	Incoming	TCP/RTP	No	Transfer/com for SIP server .
9600	Outgoing	TCP	No	Omron FINS/TCP port.
9600	Incoming + Outgoing	UDP	No	Omron FINS/UDP bind and remote ports.

Port	Direction	Protocol	Configurable	Description
44818	Outgoing	TCP	No	Allen Bradley Drivers (Ethernet/IP Symbolic/General).
47808	Incoming + Outgoing	UDP	No	BACnet device configuration remote port.
49320	Outgoing	TCP	Yes	Default Kepware port.
50000	Outgoing	TCP	Yes	Default IBM DB2 port.

Ignition Activation URLs

Activation Type	URL (TCP communication)
6 Character (1 time/as needed)	https://api.inductiveautomation.com/activation/activate
8 Character (regular communication)	https://licensing.inductiveautomation.com/v1-activation/leased/activate https://licensing.inductiveautomation.com/v1-activation/leased/nonce

For more information, see [Ignition Activation](#).

Color Selector Reference

Color Options in Ignition

The Perspective module, Vision module, and Report module each have color options for customizing the way your projects look. The Color Selector popup has four tabs with options for setting color:

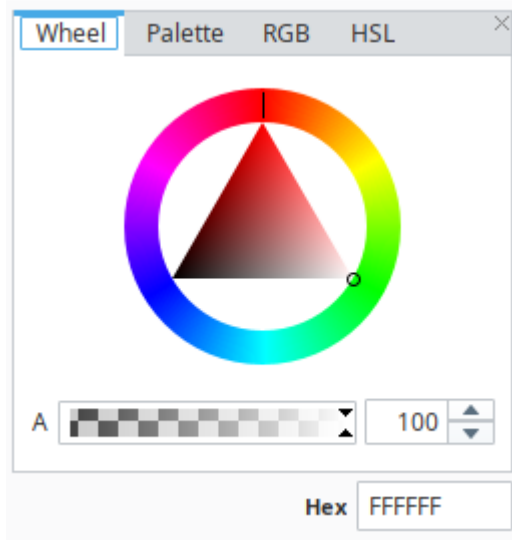
- Color Wheel
- Color Palette
- RGB Color
- HSL Color

Each of these four selectors will give you different options for selecting a single color for your properties. You can combine these in interesting ways with your styles to change foreground, background, border, and many properties. If you want to find multiple colors to create matching or complimentary color selections, you can easily find online color selectors like the one from Sessions College to get color codes and complimentary colors based on a simple selector.

Color Wheel

The Wheel tab brings up the Color Wheel. There are a few distinct areas to make selections to set up your color.

- The outer ring shows the color spectrum similar to the Hue in a HSL color selection. You can click and drag anywhere in the outer ring to select your hue.
- The inner triangle lets you choose shade of the color selected in the outer ring. The edges of the triangle represent the luminosity, and moving toward the center represents saturation.
- The Alpha slider and value allow you to set the transparency level. 0 is completely transparent, and 100 is completely opaque.
- The Hex value representing the RGB color will be shown at the bottom of the tab. You can manually change this or see it update as you change your selections above. If you change the Alpha, a fourth set of values will be added to the Hex.
- To find a color, first click and drag the line in the outer ring. Next click within the triangle.



Color Palette

The Color Palette tab offers a more simple view of colors. Just click on the color you want for your item. The Hex equivalent is also given at the bottom of the palette.

Recently selected colors will show in the row across the top of the palette, and you can import/export your palettes if you have a custom color scheme.



Exporting the Palette

You can export the Color Palette by clicking the export button in the bottom left. The file will be saved as a .CSV file type. You can edit the file in any text editor and import it back into Ignition.

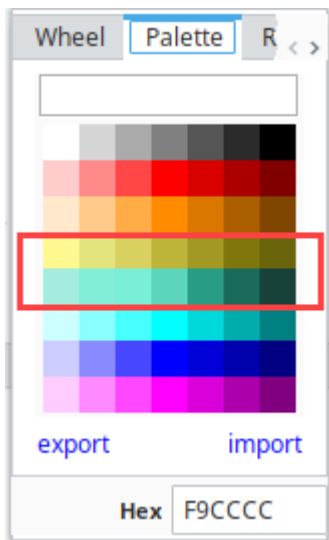
The .CSV file for the standard Color Palette in Ignition is as follows. Each color has a Hex RGB (red, green, blue) or RGBA (red, green, blue, alpha) value in quotation marks and is separated by commas. You can save this file as a .txt file or a .CSV and Ignition will be able to import it. Add as many rows or columns as you want.

```
"#FFFFFF", "#D5D5D5", "#AAAAAA", "#808080", "#555555", "#2B2B2B", "#000000"
"#FFCCCC", "#FF8A8A", "#FF4747", "#FF0000", "#D90000", "#AC0000", "#800000"
"#FFE8CC", "#FFCA8A", "#FFAC47", "#FF8C00", "#D97700", "#AC5F00", "#804600"
"#FFFFCC", "#FFFF8A", "#FFFF47", "#FFFF00", "#D9D900", "#ACAC00", "#808000"
"#CCFFCC", "#8AFF8A", "#47FF47", "#00FF00", "#00D900", "#00AC00", "#008000"
"#CCFFFF", "#8AFFFF", "#47FFFF", "#00FFFF", "#00D9D9", "#00ACAC", "#008080"
"#CCCCFF", "#8A8AFF", "#4747FF", "#0000FF", "#0000D9", "#0000AC", "#000080"
"#FFCCFF", "#FF8AFF", "#FF47FF", "#FF00FF", "#D900D9", "#AC00AC", "#800080"
```

Importing a Palette

You can also import existing .CSV files to the Color Palette. The Palette import function will also accept a .txt file if it is formatted correctly.

In this example, we modified the colors in the fourth and fifth rows of the original palette.



You can also import a new palette as long as the file contains Hex color values in quotation marks, separated by commas. The Color Palette is not limited in rows or columns. You can import a .CSV file with more colors than the standard palette in Ignition.

In this example, we added column with a 50% opacity of the first color in each row. We also added a new row with colors in the purple color family.

Each color has a Hex RGB (red, green, blue) or RGBA (red, green, blue, alpha) value in quotation marks and is separated by commas. Add as many rows or columns as you want.

```
"FFFFFF80", "#FFFFFF", "#D5D5D5", "#AAAAAA", "#808080", "#555555", "#2B2B2B", "#000000"
"FFCCCC80", "#FFCCCC", "#FF8A8A", "#FF4747", "#FF0000", "#D90000", "#AC0000", "#800000"
"FFE8CC80", "#FFE8CC", "#FFCA8A", "#FFAC47", "#FF8C00", "#D97700", "#AC5F00", "#804600"
"FFFFCC80", "#FFFFCC", "#FFFF8A", "#FFFF47", "#FFFF00", "#D9D900", "#ACAC00", "#808000"
"A4EDDE80", "#A4EDDE", "#79EFD7", "#82EED8", "#5BD5BC", "#299C85", "#1A6A5A", "#174139"
"CCFFFF80", "#CCFFFF", "#8AFFFF", "#47FFFF", "#00FFFF", "#00D9D9", "#00ACAC", "#008080"
"CCCCFF80", "#CCCCFF", "#8A8AFF", "#4747FF", "#0000FF", "#0000D9", "#0000AC", "#000080"
"DABEF680", "#DABEF6", "#C193EE", "#9C52E5", "#8647C3", "#601AA7", "#4A088B", "#3C0B6D"
"FFCCFF80", "#FFCCFF", "#FF8AFF", "#FF47FF", "#FF00FF", "#D900D9", "#AC00AC", "#800080"
```



RGB Color

The third tab on the Color options is RGBA (red, green, blue, and alpha). RGB refers to a system for representing the colors to be used on a computer display. Red, green, and blue can be combined in various proportions to obtain any color in the visible spectrum. The color is specified by an RGB value, commonly expressed in either of two ways. The decimal format is a comma separated string with one number for each color ranging from 0 to 255. For example, "0,255,0" represents green. The other is a Hex format starting with a # symbol and then listing two-digit hex values for each color, ranging from 00 to FF. For example, "#00FF00" represents green.

RGBA color values are an extension of RGB color values with an alpha channel, which specifies the opacity for a color. An RGBA color value is specified with: **rgba**(red, green, blue, alpha). In scripting the alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque). In this color selector, it follows the same rules as the other colors.

In the RGB tab, you can change the color by either dragging the selector inside the color box, using the up and down arrows next to a color box, or typing in a value.

The Hex value for the color will be shown at the bottom of the tab.

Wheel
Palette
RGB
HSL

R

255

G

255

B

255

A

255

Hex

FFFFFF

HSL Color

The fourth tab on the Color options is HSL (hue, saturation, and luminosity). HSL settings are used to adjust the way a color is displayed. Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, 240 is blue. Saturation is a percentage value; 0% means a shade of gray and 100% is the full color. Luminosity is also a percentage; 0% is black, 50% is the selected Hue, and 100% is white. Alpha is a percentage value; 0% means completely transparent and 100% is a solid color based on the other three options.

In the HSL tab, you can change the color by either dragging the selector inside the color box, using the up and down arrows next to a color box, or typing in a value.

The Hex value for the color will be shown at the bottom of the tab.

Wheel
Palette
RGB
HSL

H

0

S

0

L

100

A

100

Hex

FFFFFF

Ignition Database Table Reference

Ignition has a lot of systems built-in that will query the database automatically without requiring you to build a query. These systems automatically create the necessary tables in the database, insert the relevant data, and even query it back out for you. However, because this data is all stored in simple database tables, it can be manually accessed using queries to customize how you see the data.

CAUTION

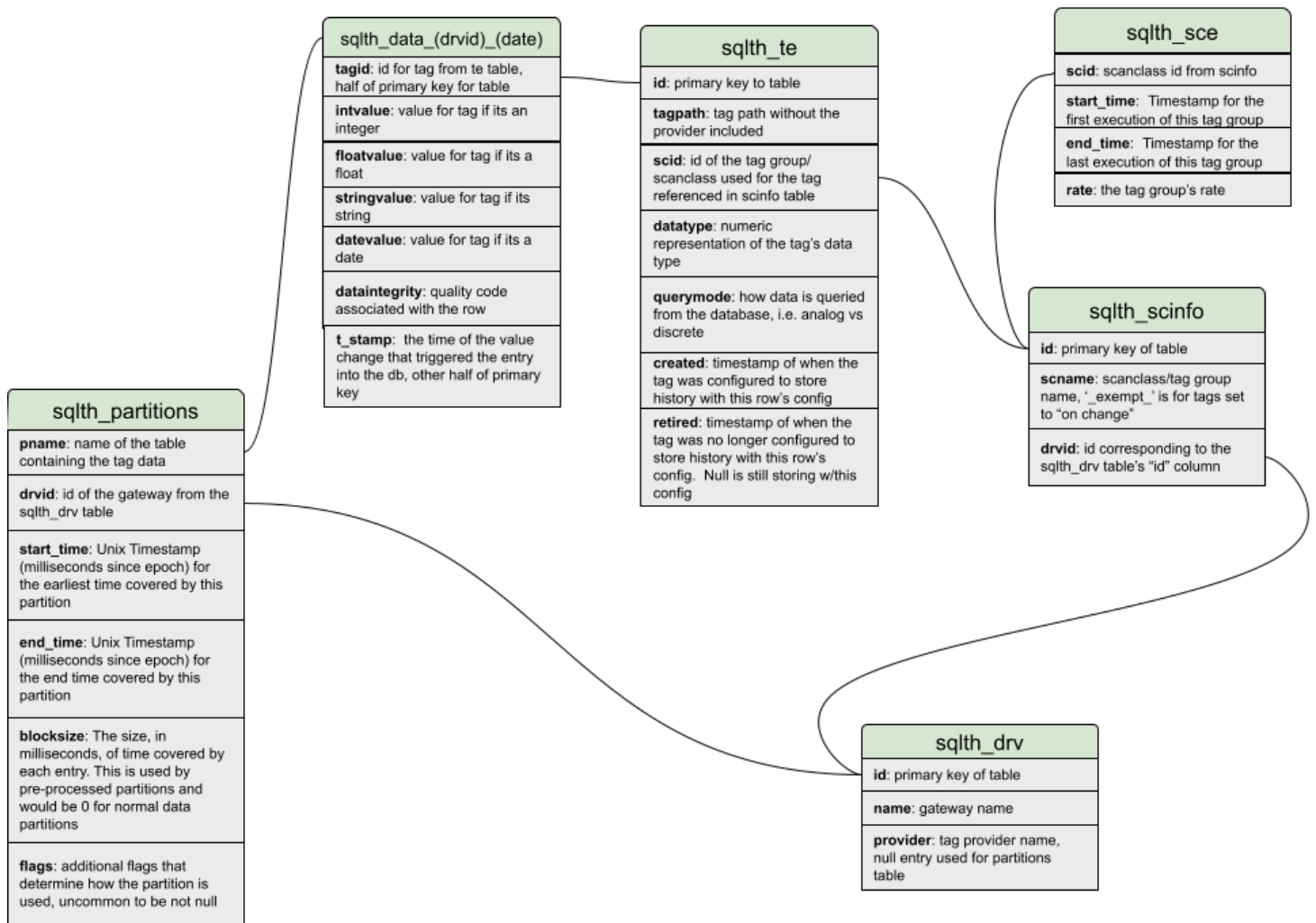
These tables are configured in very specific ways. Altering them may cause unforeseen issues, and is not recommended.

While it can be useful to manually query out data from these tables, we recommend taking a backup of the database tables before making changes, with the understanding that altering the data or tables is done at your own risk.

Tag History

The **Tag History** system utilizes at least six different tables in the database:

Table Name	Table Description	Column References
sqlt_data_X_X	This table stores the raw tag data. There will be multiple tables that fit this format depending on the name of the Gateway and the date. (For example, a table named "sqlt_data_1_2018_01" would store data from the driver with an id of 1, for the year 2018, for the month of January)	<code>sqlt_data_x_x_x.tagid = sqlth_te.id</code>
sqlth_1_data	This table stores raw tag data, and is only used when the provider is configured to use a single partition (The provider's "Enable Partitioning" setting is unchecked).	<code>sqlt_1_data.tagid = sqlth_te.id</code>
sqlth_te	This table stores the non-data details of each tag.	<code>sqlth_te.scid = sqlth_scinfo.id</code>
sqlth_scinfo	This table stores tag group information.	<code>sqlth_scinfo.drvid = sqlth_drv.id</code>
sqlth_sce	This table stores start and end times for tag groups.	<code>sqlth_sce.scid = sqlth_scinfo.id</code>
sqlth_partitions	This table stores start and end times for each sqlt_data table.	<code>sqlth_partitions.drvid = sqlth_drv.id</code>
sqlth_drv	This table stores information about the drivers of the historical data.	none
sqlth_annotations	This table stores annotations for the Tag History system, such as those created by the Power Chart .	none



sqlt_data_X_X

This is the central table that stores the core tag values. The system stores data in tables based on the history provider's partition length and units. For example, a monthly partition would use a table named like `sqlt\_data\__{driverId}\_{year}\_{month}`, whereas a daily partition would use `sqlt\_data\__{driverId}\_{yearMonthDay}`. The duration of each partition is also tracked on the `sqlth_partitions` table.

When pre-processed partitions are enabled, an additional `sql_data` table will be created for each partition. The system tracks which partitions are pre-processed by the "blocksize" column on the `sqlth_partitions` table.

Column Name	Data Type	Notes
tagid	int	Unique id of the tag. References the sqlth_te table.
intvalue	int	Stores the tag's value if it is type 0 (int/byte/short/boolean), otherwise NULL.
floatvalue	double	Stores the tag's value if it is type 1 (float/double/long), otherwise NULL.
stringvalue	string	Stores the tag's value if it is type 2 (string), otherwise NULL.
datevalue	date	Stores the tag's value if it is type 3 (DateTime), otherwise NULL.
dataintegrity	int	Quality of the tag at this timestamp. 192 = Good Quality. Other values represent uncertainty or bad quality. See Tag Quality Overlays .
t_stamp	long	Unix Timestamp (milliseconds since epoch) for the record.
vtype	int	Represents metadata about the record, primarily used for pre-processed partitions. Possible flags include: • 0: Time-weighted average • 1: Minimum value • 2: Maximum value • 4: Exit value (last in window) • 32: First value flag (used with min/max to indicate which came first) • 64: Entry value (first in window) For example, a vtype of 97 (64 + 32 + 1) means this record is the entry value, the first value, and the minimum value in the window.

NOTE

Default indexing includes:

- tagid

- t_stamp

sqlth_1_data

This table stores raw tag data for history providers that have disabled partitioning (i.e., the **Enable Partitioning** setting is unchecked on the history provider). In this mode, data is stored in a single table named **sqlth_1_data**, instead of being split into multiple sqlt_data_X_X partitions.

The structure of this table is functionally identical to the sqlt_data_X_X tables, but without any partitioning logic.

Column Name	Data Type	Notes
tagid	int	Unique ID of the tag. References the sqlth_te table.
intvalue	int	Stores the tag's value if it is type 0 (int/byte/short/boolean), otherwise NULL.
floatvalue	double	Stores the tag's value if it is type 1 (float/double/long), otherwise NULL.
stringvalue	string	Stores the tag's value if it is type 2 (string), otherwise NULL.
datevalue	date	Stores the tag's value if it is type 3 (DateTime), otherwise NULL.
dataintegrity	int	Quality of the tag at this timestamp. 192 = Good Quality. Any other value is considered poor quality or uncertain. See Tag Quality Overlays .
t_stamp	long	Unix Timestamp (milliseconds since epoch) when the value was recorded.
vtype	int	Represents metadata about the record. This value is used with pre-processed partitions. Possible flags include: • 0: Time-weighted average • 1: Minimum value • 2: Maximum value • 4: Exit value (last in window) • 32: First value flag (used with min/max to indicate which came first) • 64: Entry value (first in window) For example, a vtype of 97 (64 + 32 + 1) means this record is the entry value, the first value, and the minimum value in the window.

NOTE

Default indexing includes:

- tagid
- t_stamp

sqlth_te

This table stores metadata for each historical tag. A row in this table represents a unique version of a tag's historical configuration. When a tag is modified in certain ways (renamed, data type changed, etc.), a new row is inserted and the previous entry is "retired."

Column Name	Data Type	Notes
id	int	Unique ID of the tag. Primary key for this table.
tagpath	string	Full path of the tag within the Tag Provider, e.g., <code>Folder1/tag1</code> .
scid	int	Scan class (Tag Group) ID. References the <code>sqlth_scinfo</code> table, indicating the scan class this tag uses.
datatype	int	Determines which column in the partition tables stores the tag's historical value. 0: Stored in intvalue (used for bytes, shorts, integers, longs, and booleans) 1: Stored in floatvalue (used for floats and doubles) 2: Stored in stringvalue (used for strings and text) 3: Stored in datetimevalue (used for dates)
querymode	int	Reflects the Deadband Style setting selected when tag history was enabled. 0: Discrete, which stores values when the difference from the last stored value exceeds the deadband. 3: Analog, which uses compression algorithms to evaluate when to store values.
created	long	Unix timestamp (milliseconds since epoch) for when this tag entry was created.
retired	long	Unix timestamp (milliseconds since epoch) for when this tag entry was retired. NULL means the entry is currently active. A tag is marked as retired when any of the following occur: - The tag is deleted - The tag is renamed - The tag's scan class (Tag Group) is changed - The tag's data type is changed If a tag is simply disabled and then deleted, this value will remain NULL.

NOTE

Default indexing includes:

- id
- scid

sqlth_scinfo

This table stores information about Tag Groups (historically called scan classes). Each group defines how frequently a set of tags should record data. Tag Groups are also tied to a specific history driver.

See [sqlth_drv](#).

Column Name	Data Type	Notes
id	int	Unique ID of the Tag Group.
sname	string	Name of the Tag Group. A value of "exempt" is used when the execution rate is not recorded.
drvid	int	Driver ID this Tag Group belongs to. References the sqlth_drv table. Used to separate data when multiple Gateways or providers write to the same database.

NOTE

Default indexing includes:

- id
- drvid

sqlth_sce

This table tracks execution periods for each Tag Group over time. Each row marks a continuous span during which a Tag Group (i.e., scan class) was actively executing at a specific rate. This is useful for understanding when and how often tags within a group were scheduled to record data.

Column Name	Data Type	Notes
scid	int	References the sqlth_scinfo table. Identifies which Tag Group this execution period belongs to.
start_time	long	Unix Timestamp (milliseconds since epoch) representing the start of the execution period.
end_time	long	Unix Timestamp (milliseconds since epoch) representing the end of the execution period.
rate	int	Execution rate in milliseconds for this Tag Group during the time window defined by start_time and end_time. This value is omitted for tags using On Change scan mode.

 NOTE

Default indexing includes:

- scid
- start_time

sqlth_partitions

This table defines the data partitions used by the Tag Historian. Each row maps a time range to a specific table (usually a sqlt_data_X_X table), allowing for efficient querying across large datasets. Partitioning is key for scaling history storage over time, and the system consults this table to determine where to look for historical values.

Column Name	Data Type	Notes
pname	string	The name of the table that stores data for this partition, e.g., <code>sqlt_data_1_2025_04</code> .
drvid	int	The driver that owns this data table. References the <code>sqlth_drv</code> table. Partitions are created per driver to keep records distinct.
start_time	long	Unix Timestamp (milliseconds since epoch) marking the beginning of the time window covered by this partition.
end_time	long	Unix Timestamp (milliseconds since epoch) marking the end of the time window for this partition.
blocksize	int	The duration (in milliseconds) of each block of pre-processed data in the table. A value of 0 means this is a regular data partition.
flags	int	Additional flags that affect how the partition is used. 1 = No seed query support. The system will not attempt bounding value ("seed") queries on this partition, which can improve performance on databases with limited indexing.

 NOTE

Default indexing includes:

- `pname`
- `start_time`

sqlth_drv

This table stores information about drivers, which represent the source systems writing Tag History data. Each driver typically corresponds to an Ignition Gateway or Tag Provider. This distinction is important when multiple Gateways write to the same database, or when a Gateway has multiple Tag Providers enabled.

Column Name	Data Type	Notes
id	int	Unique ID for the driver. Used by other tables (like sqlth_partitions and sqlth_scinfo) to associate partitions and Tag Groups with a specific driver.
name	string	System name of the Ignition Gateway that originated the data. When multiple Gateways are configured to write to the same historical database, this identifies where each record came from.
provider	string	Name of the Tag Provider within the Gateway. Useful when the same Gateway hosts multiple providers and stores data for each one.

 NOTE

Default indexing includes:

- id

sqlth_annotations

This table stores annotations, user-created notes or metadata, tied to specific points or time ranges in Tag History. These annotations can be added through tools like the Perspective Power Chart or via scripting. They are useful for marking maintenance windows, process events, or any custom context.

Column Name	Data Type	Notes
id	int	Unique ID of the annotation.
tagid	int	The ID of the ag this annotation applies to. References the sqlth_te table.
start_time	long	Unix Timestamp (milliseconds since epoch) marking the beginning of the annotation's time range.
end_time	long	Unix Timestamp (milliseconds since epoch) marking the end of the annotation's time range.
type	string	Type of annotation. Currently, the only supported type is "note", which represents user-entered text.
datavalue	string	The content of the annotation (e.g., the user's note text). Only applicable when type is "note".
annotationid	string	New in 8.1.4 A UUID used to track annotations. This identifier is used to track similar annotations across multiple Gateways, such as when storing annotations in a remote provider.

NOTE

Default indexing includes:

- tagid
- start_time

Tag History - Internal History Provider

Internal History Providers use a different table scheme compared to external providers. These tables are stored in an IDB file located at `IgnitionInstallationDirectory/data/local/tag-historian`.

You'll need a SQLite viewer to explore this data manually, since the system handles it outside of the configured database connections.

Table Name	Table Description
annotations	Stores annotations for specific tags, including notes created via components like Power Chart.
schema_info	Tracks the schema version and creation timestamp of the internal historian. Usually contains one row.
tagdata	Holds the actual recorded values, timestamps, and quality codes for each tag.
tagdetails	Tracks tag metadata such as datatype and time-to-live settings. Does not include scan class/group info.
tagproperties	Stores individual tag properties like interpolation mode, deadband, and datatype.
tags	Maps each tag's ID to its full tag path, allowing for efficient querying.

annotations

These entries mirror those in `sqlth_annotations`, but are specific to local history and internal data storage.

Column Name	Data Type	Notes
id	int	Unique ID of the annotation.
tagid	int	The ID of the ag this annotation is associated with. Maps to the id column in the tags table.
type	text	Annotation type. Typically "note" for user-created textual annotations.
rangestart	int	Start time of the annotation (Unix Timestamp, in milliseconds since epoch).
rangeend	int	End time of the annotation.
data	text	The annotation content (e.g., text of the user note).
syncid	int	Identifier used to synchronize annotations across multiple Gateways.
annotationid	int	A UUID that uniquely identifies the annotation. Enables tracking across systems.
deleted	boolean	Marks whether the annotation has been logically deleted. Used to hide rather than remove data.

NOTE

Default indexing includes:

- tagid
- rangestart

schema_info

Column Name	Data Type	Notes
version	int	Indicates the schema version used by the internal historian. Typically set to 4.
created	int	Unix Timestamp (milliseconds since epoch) representing when the internal historian was first initialized.

NOTE

Default indexing includes:

- version

tagdata

Column Name	Data Type	Notes
tagid	int	References the tag's ID from the tags table. Identifies which tag this record belongs to.
numvalue	numeric	The numeric value of the tag at this time, if applicable. Will be NULL if the value is non-numeric.
strvalue	text	The string value of the tag at this time, if applicable. Will be NULL if the value is numeric.
quality	int	Quality code associated with the Tag value. See Tag Quality Overlays .
t_stamp	int	Unix Timestamp (milliseconds since epoch) representing when the value was recorded.
syncid	int	ID used to track and synchronize records across Gateways.

i NOTE

Default indexing includes:

- tagid
- t_stamp

tagdetails

Column Name	Data Type	Notes
id	int	Unique identifier for this metadata entry.
tagid	int	References the tag ID from the tags table. Associates this detail row with a specific tag.
created	int	Unix Timestamp (milliseconds since epoch) for when this detail row was created. Marks the start of history storage for this version of the tag.
retired	int	Unix Timestamp (milliseconds since epoch) for when this detail row was retired. A value of NULL means it's still active.
datatype	int	Integer representing the stored data type: • 0 = Integer • 1 = Float/Double • 2 = String • 3 = Date/Time • 4 = Boolean
ttl	int	Time-to-live, in milliseconds. If non-zero, represents how long the data is retained before eligible for pruning.
syncid	int	Sync ID used to track and coordinate this record across multiple Gateways.

i NOTE

Default indexing includes:

- tagid
- created

tagproperties

Column Name	Data Type	Notes
tagid	int	References the tag ID from the tags table. Each property row is linked to a specific tag.
name	text	The name of the property. Common values include: • InterpolationMode • Deadband • DeadbandMode • Datatype • QueryMode
value	text	The property's stored value, as a string. Interpreted based on the name property.
datatype	int	Integer representation of the property's data type. Uses the same encoding as sqlth_te.datatype: • 0 = Integer • 1 = Float/Double • 2 = String • 3 = Date/Time • 4 = Boolean

NOTE

Default indexing includes:

- tagid
- name

tags

Column Name	Data Type	Notes
id	int	Unique identifier for the tag. This value is referenced by other internal historian tables.
tagpath	text	Full path of the tag, including its Tag Provider. For example: <code>[default]Folder1/MyTag.</code>

NOTE

Default indexing includes:

- id
- tagpath

Alarm Journal

The **Alarm Journal** system stores a record of all alarm events—activations, clearings, acknowledgments—along with event-specific data.

By default, it uses two database tables, though the table names can be customized in the Alarm Journal profile settings.

Table Name	Table Description	Column References
alarm_events	Stores a row for each alarm event (active, cleared, acknowledged).	<code>alarm_events.id = alarm_events_data.id</code>
alarm_events_data	Stores name-value pairs of metadata for each event. Multiple rows may exist per event.	none

alarm_events

This table captures the main record of each alarm event that occurred. One row is created for each change in state (active, cleared, acknowledged).

Column Name	Data Type	Notes
id	int	Unique ID for this event. Primary key.
eventid	int	Identifier that groups related event records together. All active, cleared, and acknowledged states for a single alarm occurrence share the same eventid.
source	string	The source path of the alarm, such as a tag path or expression item.
displayPath	string	Human-readable display path of the alarm (can be overridden in alarm configuration).
priority	int	Alarm priority level: • 0 = Diagnostic • 1 = Low • 2 = Medium • 3 = High • 4 = Critical
eventtype	int	Alarm state change: • 0 = Active • 1 = Cleared • 2 = Acknowledged
eventflags	int	Bitmask representing metadata about the event (e.g., whether it was system-generated).
eventtime	datetime	Timestamp for when the event occurred.

NOTE

Default indexing includes:

- id

alarm_events_data

This table stores event-specific properties, such as custom alarm properties or tag values at the time of the event. Each row represents one property for one event.

Column Name	Data Type	Notes
id	int	References the id from the alarm_events table.
propname	string	Name of the property (e.g., "AckUser", "Notes", or "IsShelved").
dtype	int	Represents the type of value stored in this row: • 0 = Integer • 1 = Float • 2 = String
intvalue	int	Value of the property if dtype = 0. Otherwise NULL.
floatvalue	float	Value of the property if dtype = 1. Otherwise NULL.
strvalue	string	Value of the property if dtype = 2. Otherwise NULL.

NOTE

Default indexing includes:

- id

Authentication

The **Database Authentication** system allows you to manage users, roles, and authentication-related data using a SQL database.

By default, it uses six tables to represent user information. You can customize the table prefix (e.g., from **scada_** to something else) in the Gateway settings.

Table Name	Table Description	Column References
scada_users	Stores user credentials and basic profile data. Each row represents a user.	none
scada_roles	Stores defined roles. Each row is a distinct role.	none
scada_user_rl	Maps users to their assigned roles. Users with multiple roles will have multiple rows here.	scada_user_rl.user_id = scada_users.id scada_user_rl.role_id = scada_roles.id
scada_user_sa	Tracks upcoming schedule adjustments for users. Each adjustment gets its own row.	scada_user_sa.user_id = scada_users.id
scada_user_ci	Stores contact information for users, such as phone numbers and emails. Each entry is one piece of contact info.	scada_user_ci.user_id = scada_users.id
scada_user_ex	Stores additional user properties (used by modules like Voice Notification). Each property is a name/value pair.	scada_user_ex.user_id = scada_users.id

scada_users

This is the core user table. It contains authentication credentials and general profile information for each user.

Column Name	Data Type	Notes
id	int	Unique ID for the user.
username	string	The user's login name.
password	string	Encrypted password hash.
firstname	string	User's first name.
lastname	string	User's last name.
schedule	string	Assigned schedule name (for On-Call Roster or Scheduled Roster logic).
notes	string	Optional user notes.

** NOTE**

Default indexing includes:

- id

scada_roles

Stores all defined roles that can be assigned to users.

Column Name	Data Type	Notes
id	int	Unique ID for the role.
name	string	Name of the role (e.g., "Administrator", "Operator").

** NOTE**

Default indexing includes:

- id

scada_user_rl

Links users to their assigned roles. One row per user-role pair.

Column Name	Data Type	Notes
user_id	int	References scada_users.id.
role_id	int	References scada_roles.id.

NOTE

Default indexing includes:

- user_id
- role_id

scada_user_sa

Stores schedule overrides or adjustments for individual users.

Column Name	Data Type	Notes
id	int	Unique ID for the schedule adjustment.
user_id	int	References scada_users.id.
starttime	datetime	When the schedule adjustment starts.
endtime	datetime	When the schedule adjustment ends.
type	string	Type of adjustment (e.g., "Available", "Unavailable").

NOTE

Default indexing includes:

- user_id

scada_user_ci

Stores pieces of contact info for users, such as emails or phone numbers.

Column Name	Data Type	Notes
id	int	Unique ID for the contact info entry.
user_id	int	References scada_users.id.
type	string	Type of contact (e.g., "Email", "Phone").
value	string	Contact information value.

 NOTE

Default indexing includes:

- user_id

scada_user_ex

Stores custom or extra properties for a user. Often used by modules (e.g., Voice Notification) to store settings like PIN codes.

Column Name	Data Type	Notes
id	int	Unique ID for the property.
user_id	int	References scada_users.id.
name	string	Name of the property (e.g., "VoicePIN").
value	string	Value of the property.

 NOTE

Default indexing includes:

- user_id

Audit Log

The **Audit Log** system stores its data in the following database table. On Oracle databases, the audit log also uses a sequence named `AUDIT_EVENTS_SEQ`, which must be accounted for if `AUDIT_EVENTS` is manually dropped or renamed.

By default, the Audit Log uses a single table—though you can rename it in the Gateway’s Audit Profile settings.

Table Name	Table Description	Column Reference
AUDIT_EVENTS	Stores every auditable event that occurred, such as edits, tag writes, user logins, saves, and more.	none

 NOTE

Be aware of case sensitivity when referencing audit table and column names, as these names are uppercase.

AUDIT_EVENTS

This table stores each individual auditable event. A new row is created for every logged action across Gateway, Designer, or Client scopes.

Column Name	Data Type	Notes
AUDIT_EVENTS_ID	int	Unique identifier for the event. Primary key.
EVENT_TIMESTAMP	datetime	Timestamp for when the action occurred.
ACTOR	string	The user or system that performed the action.
ACTOR_HOST	string	Hostname or IP of the system where the action originated.
ACTION	string	Brief description of the action (e.g., "Tag Write", "Script Run").
ACTION_TARGET	string	The target affected by the action (e.g., tag path, script name, user).
ACTION_VALUE	string	The value involved in the action (e.g., written value, changed property).
STATUS_CODE	int	A 32-bit integer bitmask representing the result of the action (e.g., good, bad, uncertain). Refer to AuditStatus.SubCode for decoding specific status codes.
ORIGINATING_SYSTEM	string	Name of the project or system that performed the action.
ORIGINATING_CONTEXT	int	Numeric bitmask representing the context of origin: • 1 = Gateway • 2 = Designer • 4 = Client

** NOTE**

Default indexing includes:

- audit_events_id

- event_timestamp

Perspective Event Types Reference

Perspective Events provide the opportunity to handle any type of user input or session event as you see fit. To this end, Perspective exposes dozens of configurable events in the **Configure Events...** dialog. The goal of this page is to explain what circumstances trigger each event.

If you choose to associate a given event with a script action, you will be provided with an **event** parameter in your script. Similar to scripting functions, you can list the properties of that **event** parameter, by typing **event..** Event properties will differ for each broad class of event, and are listed by event class below, such as [mouse](#), [pointer](#), and [keyboard](#).

Component Events

Component Events serve the same purpose as Vision extension functions, and are visible and configurable only on components for which they are applicable. Refer to the following component pages for descriptions of their component events:

Display Palette Components

- [Audio](#)
- [Equipment Schedule](#)
- [Google Map](#)
- [Map](#)
- [Table](#)
- [Tag Browse Tree](#)
- [Tree](#)

Container Palette Components

- [Coordinate Container](#)
- [Split Container](#)

Embedding Palette Components

- [Accordion](#)
- [View Canvas](#)

Input Palette Components

- [File Upload](#)
- [Signature Pad](#)

Navigation Palette Components

- [Horizontal Menu](#)
- [Menu Tree](#)

Refer to the respective component pages for descriptions of component events.

Action Performed Event

The action performed event is on many components where an action of some sort may occur. Typically this is triggered when the component is clicked on, or a selection within the component is made. While the action may differ between components, the event functions in the same way.

Applicable components:

- Date Time Input
- Date Time Picker
- Barcode Scanner Input
- Button
- Checkbox
- Dropdown
- Multi-State Button
- Numeric Entry Field
- One-Shot Button
- Radio Group
- Slider
- Toggle Switch

Event Name	Description
onActionPerformed	Fired when the action of the component is done, such as the button being pressed, or a checkbox being checked.

System Events

System Events are events that are not related to any specific user input, but to status changes of the component or view.

Event Name	React Event	Description
onStartup	componentDidMount	- Fired on a component when the component's view is loaded into the browser window. - Fired on a view when the view is loaded into the browser window.
onShutdown	componentWillUnmount	- Fired on a component when the component's view is removed from the browser window (or when the session closes due to logout or timeout) - Fired on a view when the view is removed from the browser window (or when the session closes due to logout or timeout)

Selection Event Properties

There are no special properties associated with a selection event.

Mouse Events

Event Name	Description
onClick	Fired when a mouse button is pressed and released over the element with the event configured. Note that some components, such as buttons, contain properties that disable the component. Clicking a disabled component will still trigger this onClick event. In cases where a script shouldn't run because its parent component is disabled, it is recommended to either use the onActionPerformed event instead (preferred) or have the script check if the component is enabled as part of the script's logic
onContextMenu	Fired when the right mouse button is pressed and released over the element.
onDoubleClick	Fired when a mouse button is pressed and released twice over the element.
onMouseDown	Fired when a mouse button is pressed (not necessarily released) over the element.
onMouseEnter	Fired when the mouse pointer is moved onto the element.
onMouseLeave	Fired when the mouse pointer is moved off the element.
onMouseMove	Fired continuously while the mouse moves over the element.
onMouseOut	Fired when the mouse pointer is moved off the element.
onMouseOver	Fired when the mouse pointer is moved onto the element.
onMouseUp	Fired when a mouse button is released (not necessarily pressed) over the element.

Mouse Event Properties

Property Name	Description
altKey	Is true if the Alt key is pressed when the event is fired, and false otherwise.
button	Indicates which mouse button was used to perform the click event. Possible options are: • 0: Main button (usually the left mouse button) • 1: Auxiliary button (usually the wheel button or middle mouse button) • 2: Secondary button (usually the right mouse button) • 3: Fourth button (usually the browser Back button) • 4: Fifth button (usually the browser Forward button) For more information on this event, see the MDN documentation.
buttons	For handling situations where multiple mouse buttons are pressed when the event is fired, <i>buttons</i> holds an integer sum, where each button is given the following values: • 1: Main button • 2: Secondary button • 4: Auxiliary button • 8: Fourth button • 16: Fifth button Thus, if the main and fourth buttons are pressed simultaneously, the buttons value will be $1 + 8 = 9$. For more information, see the MDN documentation.
clientX	Indicates the horizontal coordinate of the mouse click relative to the edge of the session.
clientY	Indicates the vertical coordinate of the mouse click relative to the edge of the session.
ctrlKey	Is true if the Ctrl key is pressed when the event is fired, and false otherwise.

Property Name	Description
getModifierState(key)	Returns the current state of the specified modifier key: true if the modifier is active (i.e., the modifier key is pressed or locked), otherwise, false.
metaKey	Is true if the meta key (like the Windows key in Windows) is pressed when the event is fired, and false otherwise.
pageX	Same as <code>clientX</code> , except horizontal scrolling is taken into account. If some of the session is hidden on the left side, pageX will be larger than <code>clientX</code> .
pageY	Same as <code>clientY</code> , except vertical scrolling is taken into account. If some of the session is hidden above, pageY will be larger than <code>clientY</code> .
relatedTarget	A read-only property that is the secondary target for the mouse event, if there is one. For events with no secondary target, <code>relatedTarget</code> returns NULL.
screenX	Provides the horizontal coordinate (offset) of the mouse pointer in global (screen) coordinates.
screenY	Provides the vertical coordinate (offset) of the mouse pointer in global (screen) coordinates.
shiftKey	Is true if the Shift key is pressed when the event is fired, and false otherwise.

Pointer Events

New in 8.1.15 Pointer events will track touch interactions as well as stylus and pen interactions.

NOTE

Pointer events have only recently become available on all of Perspective's supported browsers. For this reason, these events may not be available in older versions of Safari and Firefox. If

pointer events do not fire as expected, we recommend upgrading your browser to the most current version.

Event Name	Description
onPointerOver	Fired when the pointer is moved onto the element.
onPointerEnter	Fired when the pointer is moved onto the element.
onPointerOut	Fired when the pointer is moved off the element.
onPointerLeave	Fired when the pointer is moved off the element.
onPointerUp	Fired when a pointer is released (not necessarily pressed) over the element.
onPointerDown	Fired when a pointer is pressed (not necessarily released) over the element.
onPointerCancel	Fired when a pointer has been disrupted, such as when a touch manipulation like panning or zooming is initiated or when more touch points are added to the surface than the device is capable of registering.
onPointerMove	Fired continuously while the pointer moves over the element.

Pointer Event Properties

Property Name	Description
height	The height (magnitude on the Y axis), in CSS pixels, of the contact geometry of the pointer.
isPrimary	Indicates if the pointer represents the primary pointer of this pointer type.
pointerId	A unique identifier for the pointer causing the event.
pointerType	Indicates the device type that caused the event (mouse, pen, touch, etc).
pressure	The normalized pressure of the pointer input in the range of 0 to 1, where 0 and 1 represent the minimum and maximum pressure the hardware is capable of detecting, respectively.
tangentialPressure	The normalized tangential pressure of the pointer input (also known as barrel pressure or cylinder stress) in the range -1 to 1, where 0 is the neutral position of the control.
tiltX	The plane angle (in degrees, in the range of -90 to 90) between the Y-Z plane and the plane containing both the pointer (e.g. pen stylus) axis and the Y axis.
tiltY	The plane angle (in degrees, in the range of -90 to 90) between the X-Z plane and the plane containing both the pointer (e.g. pen stylus) axis and the X axis.
twist	The clockwise rotation of the pointer (e.g. pen stylus) around its major axis in degrees, with a value in the range 0 to 359.
width	The width (magnitude on the X axis), in CSS pixels, of the contact geometry of the pointer.

Keyboard Events

Event Name	Description
onKeyDown	Fired whenever any key is pressed (not necessarily released) while the element is focused.
onKeyPress	Fired continuously while any key (other than Shift, Fn, and CapsLock) is in a pressed position and the element is focused. This event isn't recommend. Use onKeyDown instead. For more information, see the MDN documentation.
onKeyUp	Fired whenever any key is released (not necesarily pressed) while the element is focused.

Keyboard Event Properties

Property Name	Description
altKey	Is true if the Alt key is pressed when the event is fired, false otherwise.
charCode	Read-only property that returns the Unicode value of a character key pressed during a keypress event. Deprecated. Use the key property instead: MDN charCode
ctrlKey	Is true if the Ctrl key is pressed when the event is fired, false otherwise.
getModifierState(key)	Returns the current state of the specified modifier key. True if the modifier is active (i.e., the modifier key is pressed or locked), false otherwise.
key	Read-only property that returns the value of the key pressed by the user, taking into consideration the state of modifier keys such as Shift , as well as the keyboard locale and layout.
keyCode	Read-only property representing a system and implementation dependent numerical code that identifies the unmodified value of the pressed key. Deprecated: MDN keyCode
location	Read-only property that returns a value representing the location of the key on the keyboard or other input device.
metaKey	Is true if the meta key (like the Windows key on Windows) is pressed when the event is fired, false otherwise.
repeat	Is true if the given key is being held down such that it is automatically repeating.
shiftKey	Is true if the Shift key is pressed when the event is fired, false otherwise.
which	Read-only property that returns the numeric <code>keyCode</code> of the key pressed, or the character code (<code>charCode</code>) for an alphanumeric key pressed. Deprecated: MDN which

Text Composition Events

The following events are triggered via **composition** - meaning cases where supplementary or alternative means are used to input text. Examples include building logograms from base characters, or when voice commands are converted into text via speech recognition.

Event Name	Description
onCompositionEnd	Triggers when composition ends on an input component.
onCompositionStart	Triggers when composition begins on an input component.
onCompositionUpdate	Triggers when a new character is received while composing on an input component.

Selection Event Properties

There are no special properties associated with a selection event.

Focus Events

Event Name	Description
onFocus	Fired whenever the element is focused. Usually, a component or container becomes focused when the user starts to interact with it.
onBlur	Fired when the element loses focus. This usually happens when the user focuses something else.

Selection Event Properties

There are no special properties associated with a selection event.

Selection Events

Event Name	Description
onSelect	Fired after a section of text is selected in the element.

Selection Event Properties

There are no special properties associated with a selection event.

Touch Events

Event Name	Description
onTouchCancel	Fired when a touch point has been disrupted, such as when a touch manipulation like panning or zooming is initiated, or when more touch points are added to the surface than the device is capable of registering.
onTouchEnd	Fired when the user removes a touch point from the touch surface, including cases where the touch point physically leaves the touch surface, such as being dragged off of the screen.
onTouchMove	Fired when a touch point has moved along the touch surface.
onTouchStart	Fired when the user has touched the surface of a touch capable device.

Touch Event Properties

Property Name	Description
altKey	Is true if the Alt key is pressed when the event is fired, false otherwise.
ctrlKey	Is true if the Ctrl key is pressed when the event is fired, and false otherwise.
metaKey	Is true if the meta key (like the Windows key in Windows) is pressed when the event is fired, false otherwise.
shiftKey	Is true if the Shift key is pressed when the event is fired, false otherwise.

Wheel Events

Event Name	Description
onWheel	Fired whenever the mouse's scroll wheel is used over the element.

Wheel Event Properties

Property Name	Description
deltaMode	Indicates the unit associated with the other three delta properties. Possible values are: • 0: The other properties specify the wheel movement in pixels. • 1: The other properties specify the wheel movement in lines. • 2: The other properties specify the wheel movement in pages.
deltaX	Indicates the length of a horizontal scroll. Scrolling right will return a positive value, and scrolling left will return a negative value. Most mouse devices do not possess horizontal scrolling capability.
deltaY	Indicates the length of a vertical scroll. deltaY will be positive when scrolling down, and negative when scrolling up. This is the most common type of scroll, and coincides with the upward and downward movement of a typical mouse scroll wheel.
deltaZ	Indicates the length of a depth scroll. The value is positive when scrolling in, and negative when scrolling out. Not many mouse devices deliver this kind of scroll. Typically, only a 3D mouse would control this scroll type.

Style Reference

The following tables describe the style user interface for style properties in Perspective. These styles are based on Cascading Style Sheets (CSS), a style sheet language used for describing the presentation of a document or webpage. For an overview on using styles in Perspective, see also [Styles](#) and [Style Classes](#).

NOTE

Examples on this page are provided for demonstrative purposes. However CSS documentation is easy to obtain on the internet. For more information on the properties listed below, check out [MDN's CSS Reference documentation](#).

Text Menu

▼ Text

Font family

Size

▼

Color

Weight

#D900D9

▼

☐ Italic

Line height

Letter spacing

Word spacing

Alignment

Text align

▼

Indent

White space

▼

Text Options

Transform

Decoration

Tt

TT

tt

S

↻

S

☐ Break-word

Shadow ☒

Color

X-Offset

Y-Offset

Blur

Advanced ▲

Overflow wrap

normal

break-word

Text overflow

clip

ellipsis

Property Name	Description	Example	Output
Font Family	Specifies the font for the component. Examples of a Font Family are Times, Courier, Arial, Monospace, etc.	fontFamily: Monospace	Hello world
Size	Specifies the size of the font. All CSS length units are available and can be applied by specifying the unit after the number. If you enter just a number, Perspective assumes the value is in pixels (px).	fontSize: 26px	Hello world
Color	Specifies the color of the text. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference .	color: #FF0000	Hello world
Weight	Sets how thick or thin characters in the text are displayed. • normal - the default • bold - thick characters • lighter - thin characters • bolder - thicker characters • 100 to 900 - Define a weight outside of those specified above, with 100 being thin and 900 being thick. 400 is the same as normal, 700 is the same as bold.	fontWeight: Bold fontSize: 26px	Hello world
Italic	Checkbox to indicate if the font style should be italic.	fontStyle: italic	<i>Hello world</i>
Line Height	Specifies the height of a line. Can be specified as one of the following: • normal - A normal line height. This is the default	lineHeight: normal	Hello world

Property Name	Description	Example	Output
	• number - A number that will be multiplied with the current font-size to set the line height • length - A fixed line height in px, pt, cm, etc. • percentage - A line height in percent of the current font size. • 100 to 900 - Define a weight outside of those specified above, with 100 being thin and 900 being thick. 400 is the same as normal, 700 is the same as bold.		
Letter Spacing	Increases or decreases the space between characters. The default is 0.	letterSpacing: 12px	H e l l o w o r l d
Word Spacing	Increases or decreases the white space between words. The default is normal.	wordSpacing: 24px	Hello world

Alignment

Property Name	Description	Example	Output
Text Align	The horizontal alignment of text in a component. • start - Aligns text to the left. • end - Aligns text to the right. • left - Align text to the left. • right - Aligns text to the right. • center - Centers the text. • justify - Stretches the text so the text has equal width on each side. • justify-all - Stretches the text so the text has equal width on each side.	textAlign: right	Hello world
Text Indent	The indentation of the first line in a text-block. The default value is 0.	textIndent: 24px	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent fringilla, mi nec ultrices auctor, metus ante
White Space	The horizontal alignment of text in a component. • normal - Sequences of whitespace will collapse into a	whiteSpace: pre	Whitespace is preserved

Property Name	Description	Example	Output
	single whitespace. Text will wrap when necessary. This is the default. • nowrap - Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a <code>br</code> tag is encountered. • pre - Whitespace is preserved by the browser. Text will only wrap on line breaks. Acts like the <code>pre</code> tag in HTML. • pre-wrap - Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary, and on line breaks. • pre-line - Whitespace is preserved by the browser. Text will wrap when necessary, and on line breaks.		

Property Name	Description	Example	Output

Text Options

Property Name	Description	Example	Output
Transform	Capitalization setting for the text.		
	Tt Capitalize: Transforms the first character of each word to uppercase.	textTransform:capitlize	Hello World
	TT UpperCase: Transforms all characters to uppercase.	textTransform:uppercase	HELLO WORLD
	tt Lowercase: Transforms the first character of each word to lowercase.	textTransform:lower	hello world
Decoration	Text decoration setting. <u>S</u> Underline: Horizontal line under text.	textDecoration:underline textDecoration:line-through textDecoration:overline	<u>Hello World</u> Hello World <u>Hello World</u>

Property Name	Description	Example	Output
	 Line-Through: Horizontal line through the middle of the text.  Overline: Horizontal line above the text.		
Break-word	Allows unbreakable words to be broken.	wordWrap: break-word	The word is: Incomprehensibilities
Shadow	Adds shadow to text. Shadow is set in pixels. Color - Color for the shadow. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference. x-offset - Offset distance on	textShadow: #ff0000 6px 6px 2px	Hello World 

Property Name	Description	Example	Output
	x axis, in pixels • y-offset - Offset distance on y axis, in pixels • blur-radius - The blur-radius. The default value is 0.		
Overflow wrap	Allows the browser to break a line of text inside a component into multiple lines in an otherwise unbreakable place, thus avoiding a long string of text causing layout problems due to overflow. • normal - Breaks lines according to normal line breaking rules. Words or unbroken strings will	overflow wrap: break-word	This example uses normal: Incomprehensibilities This example uses break-word: Incomprehensibilities

Property Name	Description	Example	Output
	not break, even if they overflow the container. This is the default. • break-word - Words or strings of characters that are too large to fit inside their component will break in an arbitrary place to force a line break. A hyphen character will not be inserted, even if the hyphen property is used. Unlike word-break, this property will only create a break in cases where a whole word		

Property Name	Description	Example	Output
	could not be placed without overflowing		
Text overflow	Specifies how overflowed content that is not displayed should be signaled to the user. It can be clipped or it can display an ellipsis (...). Note that this property doesn't force overflow to occur. You'll typically need to use other properties, such as overflow and white-space. See the MDN documentation for more details. • clip - The text is clipped and not accessible. This is the default. • ellipsis - Render an ellipsis ("...") to	textOverflow: clip textOverflow: ellipsis	This example uses clip: Lore This example uses ellipsis:

Property Name	Description	Example	Output
	represent the clipped text.		

Background Menu

▼ Background

Background color

Background image

Background position

Advanced ▲

Background clip

border-box

padding-box

content-box

Background repeat

repeat

repeat-x

repeat-y

no-repeat

Background attachment

scroll

fixed

local

Background size

Box shadow

Property Name	Description	Example
Background Color	Sets the background color of a component. The background of a component is the total size of the component, including padding and border (but not the margin). You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference .	backgroundColor: #D5D5D5
Background Image	Sets one or more background images for a component. By default, a background-image is placed at the top-left corner of a component, and repeated both vertically and horizontally. This setting also has options for setting a linear or radial gradient as the background image. For more information, see the MDN docs .	backgroundImage: url(" http://example/hostedFile.png ")
Background Position	Sets the starting position of a background image. By default, a background-image is placed at the top-left corner of a component, and repeated both vertically and horizontally. Values can be provided in explicit sizes, percentages, or keywords (e.g., top, left, etc). For more information, see the MDN documentation for background-position .	backgroundPosition: 100px 100px

Advanced

Property Name	Description	Example
Background Clip	Defines how far the background (color or image) should extend within a component. (Be sure to set up the border, padding, background properties, and add some content before configuring the background-clip, otherwise you will not see any changes when you add a value for the backgroundClip. Options are: • border-box - Default value. The background extends behind the border. • padding-box - The background extends to the inside edge of the border. • content-box - The background extends to the edge of the content box. See the MDN documentation for background-clip for more information.	backgroundClip: content-box
Background Repeat	Sets if/how a background image will be repeated. By default, a background-image is repeated both vertically and horizontally. Options are: • repeat - The background image is repeated both vertically and horizontally. The last image will be clipped if it does not fit. This is default. • repeat x - The background image is repeated only horizontally. • repeat y - The background image is repeated only vertically. • no-repeat - The background-image is not repeated. The image will only be shown once. See the MDN documentation for background-repeat to learn more.	backgroundRepeat: repeat

Property Name	Description	Example	Output
Padding Top/Left/Bottom/Right	Padding is the space inside of a component, between the border and the content of the component. This property sets the padding for each of the four sides inside of a component.	<pre>paddingBottom: 2 paddingLeft: 30 paddingRight: 13 paddingTop: 13</pre>	
Margin Top/Left/Bottom/Right	Margin is the space outside of a component. This property sets the padding for each of the four sides surrounding a component.	<pre>marginBottom:10px marginLeft:10px marginRight: 10px marginTop: 10px</pre>	

Border

▼

Border

Border



Top style



solid

▼

Top width

3px

Top color



Left style



solid

▼

Left width

3px

Left color



Bottom style



solid

▼

Bottom width

3px

Bottom color



Right style



solid

▼

Right width

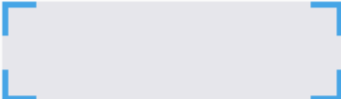
3px

Right color



Border radius

☐ All corners



Outline

Outline style

▼

Outline width

Outline color



Property Name	Description	Example	Output
Border Unlink Sides / Link Sides	The first item on the Border menu is the Border Unlink sides  icon. When you click this icon, the four border sides of the component are "unlinked," and the Border Styles settings can be applied individually to each side. When the Link Sides  icon is selected, then the Border Styles settings will apply to all four borders equally.		
Border Style	Sets the style for an component's four borders. For an example of each style, see the Border Style Options Table below.	borderStyle: dotted	
Border Width	Sets the width of the border. If the Unlink sides  icon is selected, you can set separate border widths for the top, left, bottom, and right sides.	borderWidth: 7px borderStyle: solid	
Border Color	Specifies the color of the border. You can set the color with a	borderColor: #AC00AC borderStyle: solid	

Property Name	Description	Example	Output
	HEX, RGB, or HSL value. See also Color Selector Reference. If the Unlink sides  icon is selected, you can set separate border colors for the top, left, bottom, and right sides.		
Border Radius	Defines the radius of the component's corners allowing rounded borders.	<code>borderRadius: 10px</code> <code>borderStyle: solid</code>	
Border Radii Top/Left/Bottom/Right	Sets the radius for each of the component's corners: Border top left radius, Border top right radius, Border bottom left radius, Border bottom right radius.	<code>borderTopLeftRadius: 10px</code> <code>borderBottomRightRadius: 10px</code> <code>borderStyle: solid</code>	
Outline Style	Specifies the style of an outline. For an example of each style, see the Border Style Options Table below.	<code>outlineStyle: dotted</code>	

Border Style Options

Style	Description
none	Specifies no border. This is default.
solid	Specifies a solid border.
dashed	Specifies a dashed border.
dotted	Specifies a dotted border.
double	Specifies a double border.
groove	Specifies a 3D grooved border. The effect depends on the border-color value.
ridge	Specifies a 3D ridged border. The effect depends on the border-color value.
inset	Specifies a 3D inset border. The effect depends on the border-color value.
outset	Specifies a 3D outset border. The effect depends on the border-color value.
hidden	The same as "none", except in border conflict resolution for table elements.

Outline Style

Style	Description
none	Specifies no outline. This is default.
solid	Specifies a solid outline.
dashed	Specifies a dashed outline.
dotted	Specifies a dotted outline.
double	Specifies a double outliner.
groove	Specifies a 3D grooved outline. The effect depends on the outline-color value.
ridge	Specifies a 3D ridged outline. The effect depends on the outline-color value.
inset	Specifies a 3D inset outline. The effect depends on the outline-color value.
outset	Specifies a 3D outset outline. The effect depends on the outline-color value.
hidden	Specifies a hidden outline.

Shape Menu

Style	Description
Fill	Sets a color for the fill of an SVG shape.
Stroke	Sets a color for the stroke. The stroke is a border around SVG shapes. Specifies the color of the text. You can set the color with a HEX, RGB, or HSL value. See also Color Selector Reference .
Stroke width	Sets the width of the stroke, in pixels.

Misc Menu

▼ Misc.

Opacity

Cursor

▼

Overflow

visible

hidden

scroll

auto

Overflow x

visible

hidden

scroll

auto

Overflow y

visible

hidden

scroll

auto

Property Name	Description	Example	Output
Opacity	Describes the transparency level for a component, where 1 is not transparent at all, 0.5 is 50% see-through, and 0 is completely transparent. Opacity applies to text, border, and outline properties.	opacity: 0.4	Hello World
Cursor	Specifies the mouse cursor to be displayed when pointing over a component. For a full list of options, see the MDN documentation .	cursor:pointer	
Overflow	Specifies what should happen if content overflows a component's box: whether to clip content or add scrollbars when a component's content is too	overflow:scroll	Lorem ipsum dolor sit amet, consectetur

Property Name	Description	Example	Output
	big to fit in a specified area. For an description of each value, see the Overflow Property Table .		
Overflow X	Specifies whether to clip the content, add a scroll bar, or display overflow content of a block-level component, when it overflows at the left and right edges. For an description of each value, see the Overflow Property Table .	N/A	N/A
Overflow Y	Specifies whether to clip the content, add a scroll bar, or display overflow content of a block-level component, when it overflows at the top and bottom edges. For an description of each value, see	N/A	N/A

Property Name	Description	Example	Output
	the Overflow Property Table .		

Overflow Property Table

Below is an expaination of the various values for overflow properties. Additional details can be found in the [MDN documentation](#).

Value	Description	Example
visible	The overflow renders outside the component's box. It is not clipped. This is the default.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent fringilla, mi nec ultrices auctor, metus ante
hidden	The overflow is clipped, and the rest of the content will be invisible.	Lorem ipsum dolor sit amet, consectet
scroll	The overflow is clipped, but a scrollbar is added to see the rest of the content.	Lorem ipsum dolor sit amet, consectet
auto	If overflow is clipped, a scrollbar will be added to see the rest of the content.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent fringilla, mi nec ultrices auctor, metus ante

Symbol Reference

The following reference pages provide tables showing the symbols and icons used in Ignition along with the symbol name and description.

 **Tag Browser Symbol Reference**
This page contains reference images for icons located in the Tag Browser

 **Project Browser Symbol Reference**
This page contains reference images for icons located in the Project Browser

Tag Browser Symbol Reference

This reference page provides tables showing the symbols used in Tag Browser along with the symbol name and description.

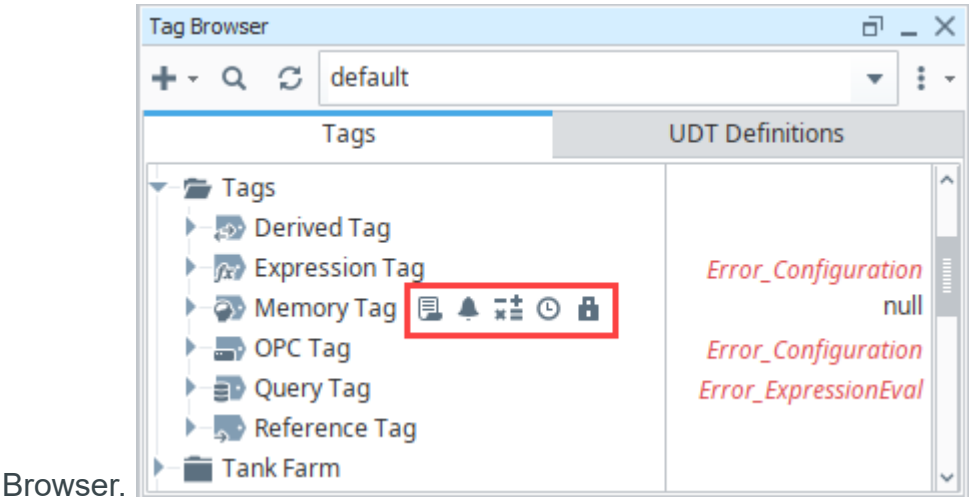
Tag Browser Icons

Icon	Name	Description
	Add	Add button used in many areas in the Tag Browser such as adding a new Device, Tag Folder, Data Type Instance, and a Tag.
	Column Selection	Options for what columns are displayed in the Tag Browser: Value, Data Type, and Traits. Check the option in the dropdown list to display that column.
	Copy	Copies the selected item, for example, a Tag or a UDT.
	Copy Tag Path	Copies the full tag path to the clipboard.
	Data Types Folder Closed	Icon displayed when the data types folder is closed.
	Data Types Folder Open	Icon displayed when the data types folder is open.
	Data Type Instance	Creates new instance of an existing Data Type. Dropdown list shows the available instances.
	Tag Groups (formerly Scan Classes)	Opens the Tag Group Editor (formerly scan class editor).
	Export Tags	Export Tags.
	Folder Closed	Icon displayed when a folder is closed.
	Folder Open	Icon displayed when a folder is open.
	Import Tags	Opens import window from which you can import a file

Icon	Name	Description
	Multi-Instance Wizard	Opens the multi-instance wizard .
	New Standard Tag	Dropdown list from which you can select an option to create a new data type, a new data type instance, or a new standard Tag. You can also create a new folder for Tags or data types.
	OPC Server	Browses the OPC servers.
	Paste	Pastes the contents of the clipboard into the current context.
	Paste Cancel	Cancels the paste action.
	Refresh	Refreshes the Tag providers.
	Rename	Enables text editing of the Tag name.
	Search	Search for Tags.
	Tag	Icon for a Tag. Appears next to the Tag name in the Tag Browser tree.
	Tag Disconnected	Indicates the Tag is disconnected.
	Tag Edit	Opens the Tag Editor screen.
	Tree Node	Icon for a node in the tree that's expanded from an individual Tag.

Tag Traits

Tag configurations or settings are visually represented by a unique icon next to the Tag in the Tag



Browser.

The following icons enable you to note some important settings on the Tag at a glance. A description of the icons are listed below.

Icon	Name	Description
	Scaling	The Scale Mode property under the Numeric Tag Properties section of the Tag Editor has been set to a value other than "Off." The value on the Tag will be scaled to some degree.
	Alarming	At least one alarm has been configured for this Tag.
	Tag History	This Tag has been configured to log data into the Tag Historian system.
	Tag Event Script	At least one Tag Event Script has been enabled on this Tag.
	Tag Permissions	The Access Rights property under the Security Tag Properties section of the Tag Editor has been set to Read only or Custom. For more information, see Tag Security Properties .

UDT Icons

The following table represents various symbols that UDT definitions can utilize.

Icon	Name	Description
	Color	The color assigned to the UDT, which will show up on the left side of the UDT icon. This property is purely cosmetic, though it can help certain UDTs stand out.
	Disabled	This UDT has been disabled.
	Alarming	At least one alarm has been configured for this UDT.
	Security	The Access Rights property under the Security Tag Properties section of the Tag Editor has been set to Read only or Custom. For more information, see Tag Security Properties .
	Scaling	The Scale Mode property under the Numeric Tag Properties section of the Tag Editor has been set to a value other than Off . The value on the tag will be scaled to some degree.
	Scripting	At least one Tag Event Script has been enabled on this UDT.
	History	This tag has been configured to log data into the Tag Historian system.
	UDT Inheritance	The UDT or tag has inherited at least one property.
	Override	Indicates that some or all inherited settings have been manually overridden .

Project Browser Symbol Reference

This reference section provides tables showing the symbols used in Project Browser along with the symbol name and description.

Project Browser Icons

Miscellaneous

Symbol	Name	Description
	Alarming	Workspace for Alarm Notification Pipelines . Appears under the Global resources. The same icon is displayed next to the name of each pipeline.
	Sequential Function Charts	Workspace for Sequential Function Charts (SFC). Same icon is displayed next to the name of each SFC.
	Properties	Opens the Project Properties Editor dialog box.

Scripting

Symbol	Name	Description
	Project Library	Workspace for the Project Library.
	Script	Icon for an individual script. Appears next to the name of the script.
	Gateway Events	Scripts for events that execute on the Gateway.

Perspective

Additional symbols (Badge icons) can show up to the right of these items. See the [Perspective Designer Interface](#) page for an explanation.

Symbol	Name	Description
	Perspective Session Events	Scripts for events that occur during a Perspective Session.
	Styles	Styles folder within Perspective.
	Views	Views folder within Perspective.
	View	Icon for an individual View. Appears next to the name of the View.
	Edit View	Edits a View within Perspective.

 Transaction Groups

Symbol	Name	Description
	Standard Group	Icon for a block Standard Group.
	Standard Group - Disabled	Indicates that capability to create Standard Groups is disabled.
	Block Group	Icon for a Block Transaction Group.
	Block Group - Disabled	Indicates that capability to create Block Groups is disabled.
	Historical Group	Icon for a Historical Group.
	Historical Group - Disabled	Indicates that capability to create Historical Groups is disabled.
	Stored Procedure Group	Icon for a block Stored Procedure Group.
	Stored Procedure Group - Disabled	Indicates that capability to create Stored Procedure Groups is disabled.

 Vision

Additional symbols (Badge icons) can show up to the right of these items. See the [Vision Designer Interface](#) page for an explanation.

Symbol	Name	Description
	Client Events	Scripts for events that execute in Vision Clients.
	Windows	Folder for Windows within the Vision Module.
	Window	Icon for a Main Window. Appears next to the Window name.
	Window - Editing	New in 8.1.21 Indicates that this Window is currently open for editing.
	Popup Window	Icon for a Popup Window. Appears next to the Popup Window name.
	Docked Window	Icon for a Docked Window. Appears next to the Docked Window name.
	Templates (project)	
	Vision Templates (shared)	Workspace for the <i>shared</i> Vision Templates. Same icon is displayed next to the name of each Template.
	Vision Templates - Editing	New in 8.1.21 Indicates that this Template is currently open for editing.

Named Queries

Symbol	Name	Description
	Named Query	Create new Named Query.
	Named Query - Disabled	Indicates that access to Global resources is disabled.
	Named Query - Editing	Icon for an individual Named Query. Clicking opens the Named Query for editing.

Reports

Symbol	Name	Description
	Report	Create new Report.
	Report - Disabled	Indicates that capability to create a Report is disabled.
	Report - Editing	Icon for an individual report. Clicking opens the report for editing.

Right-Click Menu

Symbol	Name	Description
	Rename	Renames the current selection.
	Cut	Cuts the current selection into the clipboard.
	Copy	Copies the current selection into the clipboard.
	Copy Path	Copies the current path into the clipboard.
	Paste	Pastes the content in the clipboard into the selected context.
	Delete	Deletes the current selection.
	Protect	Locks the individual project resource from inside Designer.

Data Type Formatting Reference

This is a reference page for Date, Number, and String formatting.

Date Format

Date as a string, formatted according to a pattern. The pattern is a format that is full of various placeholders that will display different parts of the date. These are case-sensitive! These placeholders can be repeated for a different effect. For example, M will give you 1-12, MM will give you 01-12, MMM will give you Jan-Dec, MMMM will give you January-December.

Functions that use this date formatting include:

- [system.date.format](#)
- [system.dataset.formatDates](#)

NOTE

The table below demonstrate some common formatting elements for convenience. Date formatting in Ignition is largely driven by Java's SimpleDateFormat class. See the [SimpleDateFormat](#) documentation for more information.

Symbol	Description	Presentation	Examples	Other Notes
a	Am/Pm marker	Text	PM	
D	Day in year	Number	189	
d	Day in month	Number	10	
E	Day name in week	Text	EEEE=Tuesday; E=Tue	
F	Day of week in month	Number	2	2nd Sunday of the month
G	Era designator	Text	G=AD	
H	Hour in day (0-23)	Number	0	
h	Hour in am/pm (1-12)	Number	12	
k	Hour in day (1-24)	Number	24	
K	Hour in am/pm (0-11)	Number	0	
M	Month in year	Month	MMMM=July; MMM=Jul; MM=07	
m	Minute in hour	Number	30	
s	Second in minute	Number	55	

Symbol	Description	Presentation	Examples	Other Notes
S	Millisecond	Number	978	
u	Day number of week	Number	1	(1 = Monday, ..., 7 = Sunday)
w	Week in year	Number	27	If Dec31 is mid-week, it will be in week 1 of the next year
W	Week in month	Number	2	
X	Time zone	ISO 8601 time zone	X=-08; XX=-0800; XXX=-08:00	
y	Year	Year	yyyy=1996; yy=96	Lowercase y is the most commonly used year symbol
Y	Week year	Year	YYYY=2009; YY=09	Capital Y gives the year based on weeks (ie. changes to the new year up to a week early)
z	Time zone	General time zone	zzzz=Pacific Standard Time; z=PST	
Z	Time zone	RFC 822 time zone	Z=-0800	

Number Format

Number Format returns a string representation of a number formatted according to a pattern provided. The pattern string includes special characters to instruct how a number needs to be formatted. They can be used to specify the number of decimal places to display, as well as more advanced formatting like scientific notation or including parentheses.

The `numberFormat` function uses the number formatting described below.

 NOTE

The tables below demonstrate some common formatting elements for convenience. Number formatting in Ignition is largely driven by Java's DecimalFormat class. See the [DecimalFormat](#) documentation for more information.

Symbol	Description
0	Specifies a required digit.
#	Specifies an optional digit.
,	The grouping separator.
.	The decimal separator.
-	A minus sign.
E	Scientific notation.
;	Used to separate positive and negative patterns. The negative subpattern will only be used to specify the prefix and suffix. The number of digits, minimal digits, and other characteristics are all the same as the positive pattern.
%	Multiplies the value by 100 and shows as a percent.
'	Used to quote special characters.

Example

This table shows some numbers, and the result of using various format strings to format them.

Number	Pattern	Result
5	0	5
5	0.05.0	
5	00.0	05.0
123	#,##0	123
1024	#,##0	1,024
1337	#,##0.#	1,337
1337.57	#,##0.#	1,337.6
87.32	#,##0.0000	87.3200
-1234	#,##0	-1,234
-1234	#,##0;(#)	(1,234)
4096	0.###E0	4.096E3
.348	#.00%	34.80%
34.8	#0.00'%'	34.80%

String Format

Functions that use this data formatting include:

- `stringFormat`

Date/Time Formatting Elements

Date/Time Formatting Element Suffix	Description
H	Hour of the day for the 24-hour clock, formatted as two digits with a leading zero where necessary, i.e., 00 - 23.
I	Hour of the 12-hour clock, formatted as two digits with a leading zero as necessary, i.e., 01 - 12.
k	Hour of the day for the 24-hour clock, i.e., 0-24.
k	Hour of the 12-hour clock, i.e., 1 - 12.
m	Minute within the hour formatted as two digits with a leading zero where necessary, i.e., 00 - 59.
S	Seconds within a minute, formatted as two digits with a leading zero where necessary, i.e., 00 - 59.
L	Millisecond within the second formatted as three digits with leading zeros as necessary, i.e., 000-999.
B	Locale-specific full month name i.e. "January", "March".
b	Locale-specific abbreviated month name i.e. "Jan", "Mar".
A	Full name of the day of the week i.e. "Monday".
a	Abbreviated name of the day of the week i.e. "Mon".
Y	Year formatted as a 4 digit numeric value with leading zeros where necessary i.e. 0005 would be the year 5 in the Gregorian calendar.
y	Last two digits of the year formatted with leading zeros where necessary.
j	Day of the year formatted as three digits with leading zeros where necessary, i.e., 001 - 366 for the Gregorian calendar.

Date/Time Formatting Element Suffix	Description
M	Month, formatted as a two digit number with leading zeros where necessary, i.e., 01-13.
d	Day of the month formatted as two digit number with leading zeros where necessary, i.e., 00 - 31.
e	Day of the month formatted as two digits, i.e., 0-31.

Formatting Elements

Element Character	Data Type to Substitute Element Character	Description
b, B	Boolean	If the corresponding argument arg is NULL, then a False is substituted into the format string. If arg is boolean, then the string conversion of arg will be substituted into the format string. For every other condition, True is substituted into the format string.
s, S	String	The string value arg is substituted into the formatted sting.
c, C	Character	The unicode value arg is substituted into the formatted string.
d	Integral	The decimal value arg is substituted into the formatted string.
f	Floating Point	The floating point value arg is substituted into the formatted string.
t, T	Date/Time	This is the prefix for date/time arg values to be used for string formatting.

Scripting Object Reference

The following sections describe the structure of some prominent objects returned or expected by Ignition system functions. Not all classes, fields, and methods are included in the lists below. For a fully comprehensive reference of all Ignition types, refer to the [Javadocs](#) or the [Ignition SDK Programmers Guide](#).

Alarming

AlarmQueryResult

Used By: [system.alarm.queryStatus](#), [system.alarm.queryJournal](#)

The [AlarmQueryResult](#) object represents a collection of [AlarmEvent](#) objects with some additional helper methods.

Methods

Name	Returns	Description
<code>getDataset()</code>	Dataset	Returns the alarms as a dataset. Columns for realtime (queryStatus) calls are: • EventId • Source • DisplayPath • EventTime • State • Priority Columns from historical calls (queryJournal) have the following columns: • EventId • Source • DisplayPath • EventTime • EventState • Priority • IsSystemEvent

PyAlarmEvent

Used By: [Perspective - Alarm Journal Table](#), [Perspective - Alarm Status Table](#), [system.alarm.queryJournal](#), [system.alarm.queryStatus](#).

This object contains information about alarm events. See the ScriptableBlockPyAlarmEvent section below for a list of methods.

ScriptableBlockPyAlarmEvent

Used By: This object is the "event" object made available to [Script Blocks](#) in alarm pipelines.

The methods listed in the table below are shared between the **PyAlarmEvent** and **ScriptableBlockPyAlarmEvent** objects, with the latter being the "event" object made available to [Script Blocks](#) in alarm pipelines. However, there is one method called `cancelNotification` that is **exclusive** to `ScriptableBlockPyAlarmEvent`.

Methods

Name	Returns	Description
<code>cancelNotification()</code>	None	When invoked, cancels the running pipeline instance.
<code>contains(String propertyName)</code>	Boolean	Returns a boolean, based on whether or not the alarm event contains a value for the named property. Useful when testing for associated data.
<code>get(String propertyName)</code>	Any	Returns the value of propertyName. Useful in cases when trying to view the value of associated data. A full list of built-in alarm properties can be found on the Tag Alarm Properties page.
<code>getAckData()</code>	EventData	Returns an object containing information acknowledgement information for the event, assuming the event has been acknowledged.
<code>getActiveData()</code>	EventData	Returns an object containing information detailing when the alarm went active.
<code>getClearedData()</code>	EventData	Returns an object containing information detailing when the alarm transitioned to a cleared status, assuming it has done so.
<code>getDisplayPath()</code>	StringPath	Returns the Display Path of the alarm.

Name	Returns	Description
<code>getDisplayPathOrSource()</code>	String	If the Display Path for the alarm in an empty string, then returns the Source Path, otherwise, returns the Display Path.
<code>getId()</code>	java.util.UUID	Returns the UUID of the alarm event.
<code>getLabel()</code>	String	The label of the alarm event.
<code>getLastEventState()</code>	AlarmStateTransition	Returns the last state of the alarm. Possible return values are Active , Acknowledged , or Cleared . The last event is always returned, so if you're looking to see if an alarm has been both Acknowledged and Cleared, use getState() instead.
<code>getName()</code>	String	Returns the name of the alarm.
<code>getNotes()</code>	String or None	Returns the value of the Notes property. Returns a None-type object if a Note has not been configured on the Alarm.
<code>getOrDefault(String propertyName)</code>	Any	Returns the value of the propertyName, or the default value if one isn't defined. Can return null if the value is null.
<code>getOrDefault(String propertyName, Object fallbackValue)</code>	Any	Returns the value of propertyName. If there is property by that name, fallbackValue is returned instead. Can return null if the value is null.
<code>getPriority()</code>	AlarmPriority	Returns an AlarmPriority object representing the Priority of the alarm. Can easily be converted to a String with str().

Name	Returns	Description
<code>getProperties()</code>	Set	Returns a set of all properties on the event.
<code>getSource()</code>	QualifiedPath	Returns the Source path of the alarm.
<code>getState()</code>	AlarmState	Returns the current state of the alarm: i.e., Active, Unacknowledged.
<code>isAked()</code>	Boolean	Returns a boolean flag indicating that the alarm has been Acknowledged.
<code>isCleared()</code>	Boolean	Returns a boolean flag indicating that the alarm has been Cleared.
<code>isShelved()</code>	Boolean	Returns a boolean flag indicating that the alarm has been Shelved.

In addition, PyAlarmEvents have the following keys that can be used to identify certain property values. For example:

```
results = system.alarm.queryStatus()
for i in results:
    print i["EventId"]
```

Key	Description
EventId	A unique identifier for the alarm event.
Source	The source path of the alarm event.
DisplayPath	The display path of the alarm event.
EventTime	A date representing when the alarm event changed State/EventState.
State	queryStatus Only. A string representing the state of the alarm. Expected values are: - Active, Unacknowledged - Active, Acknowledged - Cleared, Unacknowledged - Cleared, Acknowledged
EventState	queryJournal Only. A string representing the state of the alarm. See State for a list of possible values.
Priority	A string representing the priority of the alarm.
IsSystemEvent	queryJournal Only. A boolean representing if this event was a system event.

ShelvedPath

Used By: [system.alarm.getShelvedPaths](#)

This class provides information about an alarm that has been "shelved", including the alarm path, when it was shelved, and by whom.

Methods

Name	Returns	Description
<code>getPath()</code>	Path	Returns a Path object representing the path to the alarm. Can be cast to a string.
<code>getExpiration()</code>	Date	Returns the time that the shelve expires.
<code>getUser()</code>	QualifiedPath	Returns the actual AlarmEvent object for the specified event UUID.
<code>isExpired()</code>	Boolean	Returns True if the shelve has expired for the alarm; otherwise returns False.

Datasets and PyDatasets

Used By: Many facets of Ignition including queries, Vision and Perspective components, and Tags

See [Datasets](#).

Queries

SProcCall

Used By: [system.db.createSProcCall](#), [system.db.execSProcCall](#)

See [system.db.createSProcCall](#).

Tags

Results

Used By: [system.tag.browse](#), [system.tag.browseHistoricalTags](#)

Methods

Name	Returns	Description
<code>getResults()</code>	List	Returns a list of node dictionaries, one for each Tag (including folders and UDT instances) in the result set. See Tag Dictionary Keys below for a summary of available keys in the Tag dictionaries.
<code>getReturnedSize()</code>	Integer	Returns the number of results contained in this object. If different than <code>getTotalAvailableSize()</code> , only a partial browse was performed, and the continuation point should be non-null.

Tag Dictionary Keys

All Nodes (Tags, Folders, UDT Definitions, and UDT Instances)

Name	Type	Description
<code>fullPath</code>	BasicTagPath	A fully qualified Tag path to the node, including the name of the node. The value returned by this key is a BasicTagPath. However, Python's <code>str()</code> method can be used to convert the path to a string.
<code>hasChildren</code>	Boolean	Represents whether the node contains sub-nodes, such as folders and UDT definitions.
<code>name</code>	Unicode	The name of the node. The value returned by this key is unicode. However, Python's <code>str()</code> method can be used to convert the type to a string.
<code>tagType</code>	TagObjectType	The type of the node. Possible values are: • AtomicTag: Represents a standard Tag. • Folder: Represents a Tag folder. • UdtType: Represents a UDT definition. • UdtInstance: Represents a UDT instance. The value returned by this key is a TagObjectType. However, Python's <code>str()</code> method can be used to convert the type to a string.

Tags (tagType = AtomicTag)

Name	Type	Description
dataType	String	The data type of the Tag.
valueSource	String	Represents how the Tag derives its value.
value	Varies	The last known qualified value on the Tag.

UDT Definitions and Instances (tagType = UdtType or UdtInstance)

Name	Type	Description
typeId	String	Represents the parent data type for the UDT. UDT definitions without a specified parent type will have a value of None .

Users

Used By: [system.user.getUser](#), [system.user.getUsers](#)

This class provides information about basic user properties, except for the user's password.

Methods

Name	Returns	Description
<code>get()</code>	String	Returns the specified value, or a null value if the requested item is not present. The following list represents the various values you may use in this manner: • Username • Firstname • Lastname • Notes • Schedule • Language • Parameter - this method also accepts a parameter string. Note that it will not always return a string.
<code>getOrDefault()</code>	String	Returns the specified value, or a default value if the requested item is not present. For example, <code>user.getOrDefault(user.Schedule)</code> will return that user's schedule, or the value of Always if no schedule has been set as that is the default schedule. The following list represents the various values you may use in this manner: • Username • Firstname • Lastname • Notes • Schedule • Language • Parameter - this method also accepts a parameter string. Note that it will not always return a string.

Name	Returns	Description
<code>getId()</code>	String	Returns the internal identifier object that the backing user source needs to identify this user.
<code>getRoles()</code>	String	Returns a sequence of strings representing the roles that this user belongs to.
<code>getContactInfo()</code>	ContactInfo	Returns a sequence of ContactInfo objects. Each of these objects will have a contactType and value property representing the contact information, both strings.
<code>getScheduleAdjustments()</code>	ScheduleAdjustment	Returns a sequence of ScheduleAdjustment objects. Each of these objects will have two date properties, "start" and "end", a boolean property, "available", and a string property called "note".
<code>getPath()</code>	QualifiedPath	Returns a QualifiedPath object that represents this user in a deterministic manner.

Perspective

See [Perspective Component Methods](#).

Misc Objects

QualifiedValue

Used By: Many facets of Ignition including Tag Event Scripts, various [system.tag](#) functions, and Perspective Property Change scripts

The **QualifiedValue** class is a representation of an Ignition value often used to represent a Tag's current condition (as in the result of a Tag read) and is meant to provide more context than a value alone. Two qualified values are considered equal if their quality and value are equal. The properties and methods on a **QualifiedValue** object are as follows:

Properties

Name	Type	Description
<code>value</code>	Varies	The data associated with the QualifiedValue object. Type depends on the type of the underlying value.
<code>quality</code>	QualityCode	The quality associated with the returned value. See QualityCode .
<code>timestamp</code>	Date	The time associated with the returned value.

Methods

Name	Type	Description
<code>getValue()</code>	Varies	Gets the data associated with the QualifiedValue object. Type depends on the type of the underlying value.
<code>getQuality()</code>	QualityCode	Gets the quality associated with the returned value. See QualityCode .
<code>getTimestamp()</code>	Date	Gets the time associated with the returned value.

QualityCode

Used By: QualifiedValue Objects

The **QualityCode** class describes a quality for a value.

Properties

Name	Type	Description
<code>good</code>	Boolean	Returns True if this code represents a Good quality level. Otherwise, returns False.
<code>uncertain</code>	Boolean	Returns True if this code represents an Uncertain quality level. Otherwise, returns False. Note that Bad and Error quality levels return False.
<code>bad</code>	Boolean	Returns True if this code represents a Bad quality level. Otherwise, returns False. Note that Uncertain and Error quality levels return False.
<code>error</code>	Boolean	Returns True if this code represents an Error quality level. Otherwise, returns False. Note that Bad and Uncertain quality levels return False.
<code>notGood</code>	Boolean	Returns True if this code represents an Uncertain, Bad, or Error quality level. Otherwise, returns False.
<code>badOrError</code>	Boolean	Returns True if this code represents a Bad or Error quality level. Otherwise, returns False.
<code>code</code>	Integer	The 32-bit integer code associated with the quality.
<code>diagnosticMessage</code>	String	The diagnostic message associated with this quality, if present. Otherwise, returns None.

Methods

Name	Returns	Description
<code>getCode()</code>	Integer	Returns the 32-bit integer code associated with the quality. For more information on quality codes, including a list of available codes, see Quality Codes and Overlays .
<code>getDiagnosticMessage()</code>	String	Returns the diagnostic message associated with this quality, if present. Otherwise, returns None.
<code>isGood()</code>	Boolean	Returns True if this code represents a Good quality level. Otherwise, returns False.
<code>isUncertain()</code>	Boolean	Returns True if this code represents an Uncertain quality level. Otherwise, returns False. Note that Bad and Error quality levels return False.
<code>isBad()</code>	Boolean	Returns True if this code represents a Bad quality level. Otherwise, returns False. Note that Uncertain and Error quality levels return False.
<code>isError()</code>	Boolean	Returns True if this code represents an Error quality level. Otherwise, returns False. Note that Bad and Uncertain quality levels return False.
<code>isNotGood()</code>	Boolean	Returns True if this code represents an Uncertain, Bad, or Error quality level. Otherwise, returns False.
<code>isBadOrError()</code>	Boolean	Returns True if this code represents a Bad or Error quality level. Otherwise, returns False.
<code>toValue()</code>	QualifiedValue	Returns this quality as the quality component of a QualifiedValue object.
<code>is(QualityCode other)</code>	Boolean	Compares the integer quality codes of another QualityCode object to this one. Returns True if they match.

Name	Returns	Description
<code>isNot(QualityCode other)</code>	Boolean	Compares the integer quality codes of another QualityCode object to this one. Returns True if they do not match.

Crontab Formatting Reference

Several features in Ignition allow you to specify a recurring schedule using crontab syntax. Crontab is a well documented syntax across the web. However, this page will provide a short reference to crontab as it's implemented in Ignition.

Syntax Overview

Crontab formatting is made up of five fields, each of which represent a different unit of time. Normally crontab is represented horizontally, with the left most field representing minutes.

```
* * * * *  
| | | | |  
| | | | ----day of the week  
| | | ----month  
| | ----day of the month  
| ----hour  
----minute
```

Each field has a range of expected values, in addition to the special characters listed elsewhere on this page.

- minute expects a value from 0-59
- hour expects a value from 0-23
- day of the month expects a value from 1-31
- month expects a value from 1-12
- day of the week expects a value from 0-6, with 0 representing Sunday and 6 representing Saturday.

Special Characters

Additional characters can be provided to a field in cases where multiple values or distinct intervals are required.

Value	Description	Example (using the Minutes field)
All (*)	Denotes any possible value for the given unit.	The following executes every minute, starting at 2:00 AM, with the last execution being at 2:59 AM. <code>* 2 * * *</code>
Comma (,)	Allows multiple values to be specified for the unit of time.	The following would run at the start of every hour, and at the 25th minute of every hour. <code>0,25 * * * *</code>
Dash (-)	Used to specify a range of values. Useful if you want to have the action occur at consecutive intervals, but not at every interval.	The following would run every minute, starting at 2:00 AM, with the last execution at 2:06 AM. <code>0-6 2 * * *</code>
Last (L)	The last possible unit. Only available on the Days field. Useful in cases where something needs to run on the last day of the month.	The following would execute at 11:59 PM on the last day of the month. <code>59 23 L * *</code>
Every Other (/#)	Allows for a range of intervals to be defined, where "#" is a number to step by. For example, setting the day field to <code>*/2</code> would execute the schedule every other day (assuming the other fields are set to <code>*</code>).	The following would execute every three minutes. <code>*/3 * * * *</code>

Crontab In Ignition

- Ignition Gateways can automatically **collect backups of themselves**. The schedule for which uses a crontab formatted string.
- Gateway Scheduled Scripts** are scripts that can run on a specific schedule.

- **Reports** can be scheduled to execute on a schedule.

Logback XML File Reference

This section details the various configuration changes that can be made to the `logback.xml` file. Ignition-specific properties will be covered here, with official documentation linked for Logback-specific functions.

In a usual Ignition environment, logs come from the **Java Service Wrapper** and are displayed in two locations:

- The **console**, using **standard out (stdout)**.
- A log file (also known as a **wrapper log**).

Ignition will log to stdout, which will then be used by the wrapper. This leads to the wrapper log(s) displaying the information you would typically see, such as the logging level, Java Wrapper Service process identifier (PID), date and time, and log message. Most log events are between 600 to 800 bytes on disk.

Configuration File Location

The `logback.xml` file can be found at:

```
%IgnitionInstallationDirectory%/data/logback.xml.
```

See the [Installing and Upgrading Ignition](#) page for default installation directories.

Official Logback Properties and Documentation

There are general, non-Ignition related settings in the `logback.xml` file that can be configured and changed. Documentation for these settings and arguments, such as `appender`, `encoder`, and `pattern`, can be found in the [Logback manual](#).

Ignition Logback Properties

The table below lists properties and their descriptions that are specific to Ignition:

Maintenance Settings

Property	Description
entryLimit	The maximum number of entries in the database. However, at any given time, there may be more than this number, due to how cleanup works. Default is 50000.
maxEventPerMaintenance	The number of events that can happen before a maintenance cycle occurs. Default is 5000.
minTimeBetweenMaintenance	The minimum time in milliseconds (max frequency) between maintenance events. Takes precedence over max events. Default is 60000, meaning maintenance events will not occur more often than once per minute.
vacuumFrequency	The number of maintenance cycles before a "vacuum" is performed, to recover disk space. Default is 3.
diskspaceCleaningEventCount	New in 8.1.21 The number of oldest events that will be removed when max database size is exceeded. Default is 500.
maxDatabaseSize	New in 8.1.21 Database size in bytes. Going above this value will trigger a maintenance cycle to remove diskspaceCleanupEventCount of oldest events. Default is 104857600, or around 104.86 megabytes.

Migrating Loggers

You can change Ignition's logging output location. This requires setting up several elements such as the new folder location to store the log files, an appender, a size limit for the log file, and more. Full instructions on migrating Ignition loggers can be found [here](#).

Gateway XML File Reference

 NOTE

This page talks about the `gateway.xml` file. If you are instead looking for information on the Gateway Configuration file, also called `ignition.conf`, click [here](#).

This section details the various configuration changes that can be made to the `gateway.xml` file. Select networking properties, such as port configuration, Gateway SSL properties, and more will be covered here. Many of these settings can also be modified in the Gateway's [Web Server settings](#).

Property Format

The properties in the `gateway.xml` file follow the convention below:

Format	Example
<code><comment>Context Settings</comment></code>	<code><entry key="gateway.port">8088</entry></code>

Configuration File Location

The `gateway.xml` file can be found at:

```
%IgnitionInstallationDirectory%/data/gateway.xml
```

See the [Installing and Upgrading Ignition](#) page for default installation directories.

Gateway XML Properties

The table below lists all the properties found in the `gateway.xml` file.

 CAUTION

Keep in mind that while these properties can be modified, most installations won't need to. If you are interested in changing these properties, please read through the descriptions thoroughly, and consult with our support department before proceeding.

System Property	Description
<pre><entry key="gateway.publicAddress.httpPort"> </entry></pre>	The public facing HTTP port that Vision Clients and Perspective Sessions must use to connect, such a default value.
<pre><entry key="gateway.publicAddress.autoDetect"> </entry></pre>	Set this property to <code>false</code> to specify an explicit HTTP address that Vision Clients and Perspective Sessions use. Most users will leave autodetect on. Default is <code>true</code> .
<pre><entry key="gateway.publicAddress.httpsPort"> </entry></pre>	The public facing HTTPS port that Vision Clients and Perspective Sessions must use to connect. If Force Secure Redirect is enabled, redirected connections will use this port. No default value.
<pre><entry key="gateway.forceSecureRedirect"> </entry></pre>	If both this property and SSL/TLS are enabled, all HTTP traffic will be redirected to its HTTPS counterpart. Default is <code>false</code> .
<pre><entry key="gateway.metroSSLPort"> </entry></pre>	The port to which Ignition will listen for incoming HTTP traffic over the Gateway Area Network (GAN). Default is <code>8060</code> .
<pre><entry key="gateway.includedCipherSuites"> </entry></pre>	Whitelist of included cipher suits for clients connecting to Ignition using SSL/TLS. Default is an <code>empty array</code> .
<pre><entry key="localdb.autobackup.delay"> </entry></pre>	Changed in 8.1.27 This setting determines how long a delay there is in between backups of Ignition's configuration file, in <code>minutes</code> . The backup file is generated in the <code>%IgnitionInstallationDirectory%/data/db/autobackup</code> folder. Default is 60.
<pre><entry key="context.startup.useautobackups"> </entry></pre>	Whether or not the Gateway will use autobackups. Setting this property does not prevent the Gateway from creating autobackups, as this is managed by <code>localdb.autobackup.count</code> . Default is <code>true</code> .
<pre><entry key="gateway.sslport"></entry></pre>	The port to which Ignition will listen for incoming web traffic, such as HTTPS traffic, such as Designers, Vision Clients, and Perspective Sessions.

System Property	Description
	Perspective sessions, and Gateway web configuration settings. Default is <code>8043</code> .
<code><entry key="context.props.version"> </entry></code>	Identifies which version of the context properties the <code>gateway.xml</code> file uses. Default is <code>3</code> .
<code><entry key="gateway.port"></entry></code>	The port to which Ignition will listen for incoming HTTP traffic. Default is <code>8088</code> .
<code><entry key="gateway.useProxyForwardedHeader"> </entry></code>	When enabled, the Gateway inspects each incoming request in search for headers that indicate it has been forwarded by one or more proxies. If these headers are present, then the request is updated so that the proxy is seen as the other end point of the connection from the request originated. Default is false. Warning: Enabling this setting when users can directly connect to the Gateway is a security risk. This setting is intended to be used in scenarios where untrusted clients cannot be able to bypass a trusted proxy that is responsible for setting the appropriate headers. For more information, see Use Proxy Forwarded Headers Explained.
<code><entry key="localdb.autobackup.count"> </entry></code>	The number of automatic Gateway backups that will be retained before old backups are overwritten. Setting this property to 0 will prevent autobackups from being created. Default is 5.

System Property	Description
<pre><entry key="gateway.maxThreads"> </entry></pre>	The maximum number of webserver threads that are allowed. Webserver threads are needed when communicating with the Gateway or if a subsystem does not have its own thread pool. Some example subsystems that use webserver threads are: • The Gateway webpage • The Designer • Vision Clients • WebDev module requests Default is 300.
<pre><entry key="localdb.faultbackup.count"> </entry></pre>	The number of backups from Ignition's faulted configuration file. Backups of faulted config.idb files will follow a naming convention containing faulted_ as a prefix. Default is 3.
<pre><entry key="data.dir"></entry></pre>	The relative directory in which the data directory is located. Used when the web server parses information for resources and settings outside of a Gateway context. This allows for the data directory to be used for both Gateway and commissioning. Modifying this property will require that files from the previous data folder are migrated to the new destination. Default is data\.
<pre><entry key="gateway.resolveHostNames"> </entry></pre>	When enabled, Ignition's web server will attempt to resolve the remote HTTP client's hostname by performing a reverse DNS lookup using the remote HTTP client's IP address where appropriate. Enabling this setting can have a performance impact as the Gateway may have to perform an expensive hostname lookup when handling requests. When disabled, Ignition's web server will not attempt to resolve hostnames, and any queries for the remote client's hostname will result in their IP address instead. When enabling this setting, it is highly recommended that reverse DNS is configured to prevent host lookup failures. This includes configuring valid mappings from user IP addresses to their hostnames, and from users' hostnames to their IP addresses.

System Property	Description
	back to their IP addresses. If reverse DNS isn't correct then DNS queries could block certain requests until queries time out (default of 10 seconds). Default is <code>false</code> .
<pre><entry key="gateway.acceptCount"> </entry></pre>	Controls how many connections are allowed against Gateway port, also known as the accept queue size or accept backlog. Default is <code>100</code> .
<pre><entry key="gateway.excludedCipherSuites"> </entry></pre>	Blacklist of excluded cipher suites for clients connecting using SSL/TLS. Takes precedence over all other cipher suites. Default is an <code>empty array []</code> .
<pre><entry key="gateway.publicAddress.address"> </entry></pre>	The public facing address that Vision Clients and Perspective Sessions must use to connect, such as <code>yourcompany.com</code> . If Force Secure Redirect is enabled, redirected connections will use this address. No default value.

resource.json File

Each Ignition project resource contains a `resource.json` file within the resource folder. The `resource.json` file is comprised of both user and calculated metadata. It also contains a manifest of other resource data files that compose the actual resource's data to the rest of Ignition. In addition to improving conflict resolution and version tracking, the `resource.json` file can be used to audit modifications that may happen directly on the file system, outside of Ignition. The following table describes the objects listed in the `resource.json` file.

Object	Description
Scope	Identifies where a resource is used: • G: Gateway • C: Client • A: All
Version	Defines the type of resource and how to parse data internally. For example, named queries in version: 2 makes data available as listed attributes when they are not in version: 1.
Restricted	If true, only users assigned proper role requirements will have access.
Overrideable	If true, the ability to override the inherited resource is enabled. This object does not apply in child projects.
Files	Displays the location resource data is stored, most commonly data.bin or config.json.
Attributes	Contains resource property values. Two attributes are stored automatically, lastModification and lastModificationSignature. Modification updates are useful for audit purposes and determining if updates were made internally or otherwise. Other desired attributes can be added by users.

Note, the `project.json` file functions similarly to store fundamental data and is included all projects. The `project.json` file displays the project's title, parent project name (if applicable), project description, inheritable status, and if the project is enabled on the Gateway.

System Name Usage Reference

The system name of a Gateway is fairly important since many systems within Ignition use that name as a reference. Changing the system name could cause one or more of these dependent systems to fail. This page contains a listing of features that uses a Gateway's system name.

Alarming

Remote Gateway Notification Profiles

Remote Gateway Notification Profiles reference a specific Gateway by name. If the system name of the remote Gateway was changed, then the remote notification profile may not function as intended.

Remote Alarm Journal Profiles

Configuring a **remote alarm journal profile** involves specifying the system name of the remote Gateway. Changing the remote Gateway's system name could cause the profile to stop functioning.

EAM

Agent Configurations

Agent Gateways are identified by their system name. Changing the system name on the agent could result in unintentional behavior.

Edge

Edge Sync Services

Edge **Sync Services** forwards data to a standard edition Gateway. Changing the remote Gateway's system name without updating the Data Sync configuration on the Edge Gateway could cause synchronization attempts to fail.

Gateway Network

Security Settings

The Connection Policy and Specified List settings under the [Gateway Network general settings](#) can be used to restrict Gateway network connections based off the system name on the incoming connection. Changing the system name of a Gateway may prevent other gateways that use a **SpecifiedList** connection policy from accepting the incoming connection.

Scripting

Message Handling Functions

Some functions can trigger message handling events on separate Gateways. These functions make use of a **remoteServer** parameter to specify the target Gateway by system name. Changing a Gateway's system name might cause the functions below to no longer target the Gateway. In these cases you can simply update the calling scripts so they use the updated system name.

- [system.util.sendMessage](#)
- [system.util.sendRequest](#)
- [system.util.sendRequestAsync](#)

Tag History Functions

System functions that directly interact with the historian tables may be impacted by changing a Gateway's name, since these functions typically include a historian driver in their path. See [History Tag Search and Export](#) for more details. The functions below may fail if the targeted Gateway had its system name changed. Should this occurs, you simply need to update the scripts to account for the new name.

- [system.tag.browseHistoricalTags](#)
- [system.tag.deleteAnnotations](#)
- [system.tag.queryAnnotations](#)
- [system.tag.queryTagHistory](#)
- [system.tag.storeAnnotations](#)

Security Zones

Security Zone Configuration

Security Zone configurations have a **Gateway Names** setting that is used to approve of Gateways by their system name. Changing the system name of a Gateway that was previously accepted by a zone's **Gateway Names** setting may result in the newly named Gateway being placed into a different security zone than before.

Tag Historian

History Records

The tables used to store records for the Tag Historian system can support records from multiple Gateways. This is facilitated by entries in the **sqlth_drv table**. Each tag provider on a Gateway that inserts values into the historian provider receives a separate entry on the sqlth_drv table, and all recorded values are mapped to their associated "drv" or driver entry.

Thus, changing the system name of a Gateway will cause future records to be stored under a different driver entry in the sqlth_drv table. These new driver entries will not be linked to the previous records, meaning any recorded values will not be associated with older entries. This typically manifests as historical queries with "missing" records from before the change. For example, a charting component might only be able to show data points from after the change, making it appear as if there are no records from before the change.

This can typically be fixed by modifying the sqlth_drv table entries. However it's highly recommended to reach out to our **support department** should this happen, instead of modifying the tables yourself.

Internal Historian Provider Configuration

System names are directly referenced by the Sync Settings on **Internal Historian Providers**. Changing the system name of a Gateway that is being targeted by these sync settings could interrupt the synchronization process.

Remote History Provider Configuration

System names are directly referenced by the Remote Gateway settings on **Remote Historian Providers**. Changing the system name of a Gateway that is being targeted by these settings could

interrupt the transfer of historical values to the remote Gateway.

Tag Providers

Remote Tag Provider Configuration

The Gateway setting on a **Remote Tag Provider** references the system name of the target Gateway. Changing the system name of the remote Gateway could prevent the remote tag provider from functioning properly.

Vision

Tag Browse Tree Component

The Root Node Path property on the **Tag Browse Tree component** can use a historical tag path, which includes the system name. Changes to the target system name could cause the component to not work.

Resource Guides

Tutorials and Helpful Tricks

Ignition 8 Deployment Best Practices

Learn more about best practices for setting up development and testing workflows

Ignition Historian Guide

Ignition's data historian functionality is a robust set of features that are built into Ignition modules, providing data...

External Resource Guides

General Resources

- [Ignition Server Sizing and Architecture Guide](#)
- [Ignition Security Hardening Guide](#)
- [Hosted/Multi-tenant Ignition Cloud Edition Guidance](#)

Service Integration

- [Using Keycloak with Ignition](#)
- [Let's Encrypt Guide for Ignition](#)

Industry Specific

- [NERC CIP Best Practices with Inductive Automation and Ignition](#)
- [21 CFR Part 11 and Pharmaceutical Best Practices with Ignition](#)

Tutorials and Helpful Tricks

This section includes helpful information and guides for doing specific tasks within Ignition.

Mapping a Network Drive

Learn how to make a network drive available to Ignition

Gateway Loggers

Tips for examining the Gateway's loggers

Development Server Best Practices

General recommendations for setting up your own development server

Defining SSL on a Perspective Reverse Proxy

Reverse Proxy Use Cases

Optimize Gateway Network Performance

Tips on how to better optimize your gateway network

Scale-Out Architecture Guide

A guide demonstrating how to migrate to a scale-out architecture

Secure Disposal Guide

Learn how to securely remove Ignition from a server

Tag History vs. Transaction Groups

In this article, you will be informed on different manufacturing process types and which of Ignition's data logging ...

Recommended Anti-Virus and Endpoint Security Settings with Ignition

A list of common anti-virus software exclusions to allow full Ignition functionality.

Accessing Projects Remotely

This page provides an overview of different methods to access your Ignition Vision and Perspective projects fro...

Migrating Servers

This page outlines the steps to migrate an Ignition Gateway from one server to another. A common use case for t...

Mapping a Network Drive

Windows makes it possible to map drives on network servers to local drive letters so they can be accessed by users as if they were a local drive. The problem, however, is that Windows is not very consistent about how it handles such mapped drives when accessed from a Windows Service, such as the Ignition Gateway. When the service is started manually, the drives will be available, but when the system is rebooted, the service will no longer be able to access them. There may be users that wish to read or write data in Ignition using shared drives, and don't want to manually set up shared drives each time.

! INFO

Must be using at least Ignition version 7.5+ and Java Service Wrapper 3.5.4

To make shared drives available to Ignition on startup, place the following lines in the `ignition.conf` file, which is located in the data folder of the main Ignition installation folder (usually `C:\Program Files\Inductive Automation\Ignition\data` for Windows users):

```
//Note that target "Z:" is the drive letter assigned to the mapped drive.  
  
wrapper.share.1.location=\\fileserver\folder  
wrapper.share.1.target=Z:  
wrapper.share.1.type=DISK
```

Then Stop the Gateway and Start the Gateway. Restarting does not enable this functionality.

Change the appropriate data for location and target to match the computer's actual setup. If your shared drives require authentication, add the following lines, filling in the appropriate data for user, domain, and password:

```
wrapper.ntservice.account=user  
wrapper.ntservice.password=password  
wrapper.share.1.account=domain\user  
wrapper.share.1.password=password
```

To turn on debugging to see what is causing network share connection issues, make sure to enable `wrapper.debug = TRUE` by removing the pound (#) sign in front of that line of code. The location of the log file is in the main Ignition installation folder as `wrapper.log`.

NOTE

Remember, the Gateway must be stopped then started to get the changes to the `ignition.conf` file to take effect. Restarting the Gateway will not work.

Notes

Un-map drive on shutdown

```
wrapper.share.1.shutdown.unmap=TRUE
```

How often to retry server connection:

```
wrapper.share.1.startup.max_retries=2
```

What interval (seconds) between retries:

```
wrapper.share.1.startup.retry_interval=10
```

Set wrapper to fail to startup if network share not found:

```
wrapper.share.1.startup.failure=SHUTDOWN
```

Troubleshooting

Here are a couple of the common problems that are encountered when mapping network shares:

Server not found

The debug output will show something like this if a drive can't be reached:

```
wrapper | Attempting to map the "\\fileserver\folder" share to "S:"...  
wrapper | Unable to map "S:". Attempt #1 (The network name cannot be found.  
(0x43))  
wrapper | Attempting to map the "\\fileserver\folder" share to "S:"...  
wrapper | Unable to map "S:". Trying to continue. (The network name cannot be  
found. (0x43))
```

Incorrect Login Data

If the configured account or password are incorrect (or are missing) then the mapping will fail with a message like the following:

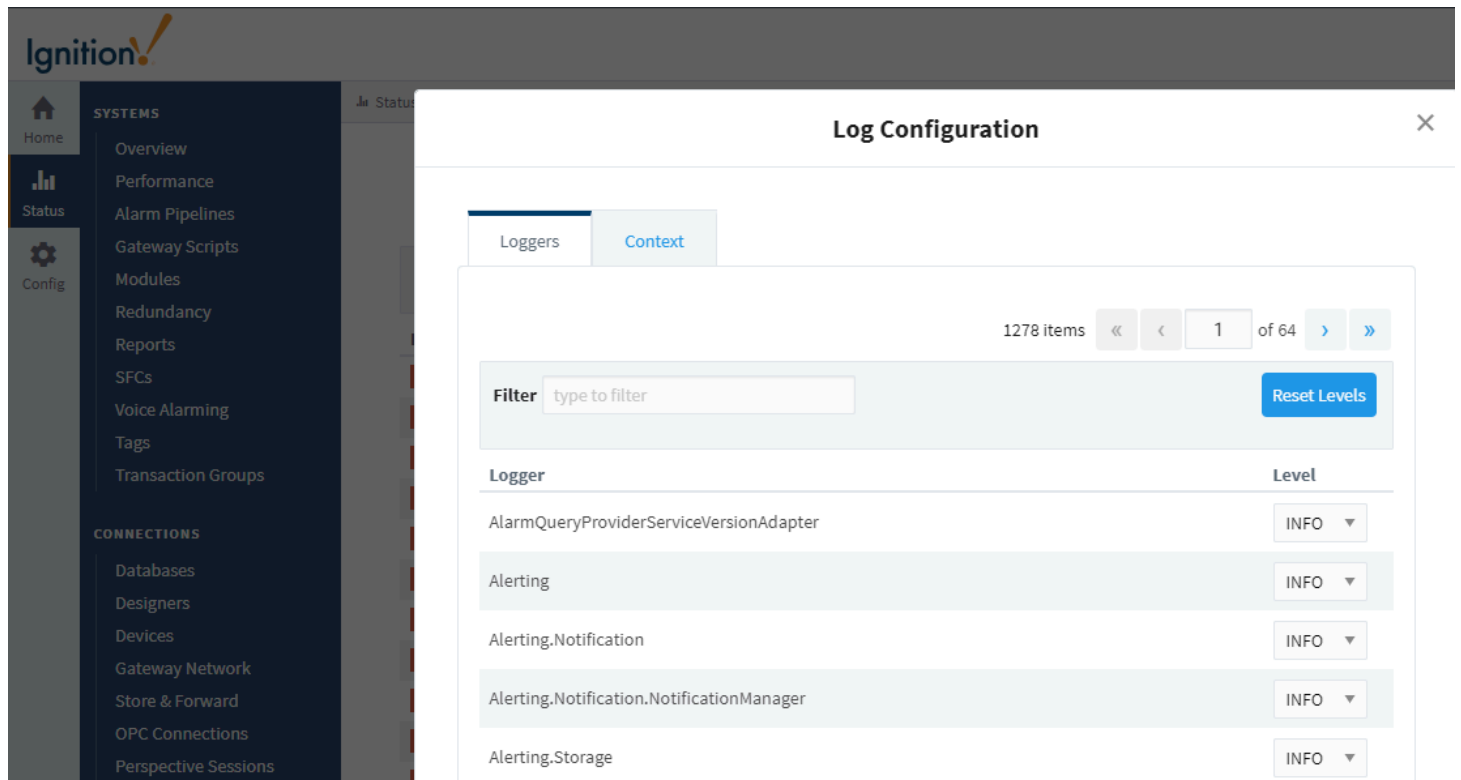
```
wrapper | Attempting to map the "\\myfileserver\commonshare" share to "S:"...  
wrapper | Unable to map "S:". Trying to continue. (Access is denied. (0x5))
```

Gateway Loggers

When troubleshooting gateway scoped issues, loggers on the gateway's console can be used to find more details about the issue. To find them, navigate to the Gateway Status page. Once there, go to the the Logs page under Diagnostics.

The screenshot shows the Ignition Gateway Status page. The left sidebar contains a navigation menu with sections: SYSTEMS (Overview, Performance, Alarm Pipelines, Gateway Scripts, Modules, Redundancy, Reports, SFCs, Voice Alarming, Tags, Transaction Groups), CONNECTIONS (Databases, Designers, Devices, Gateway Network, Store & Forward, OPC Connections, Perspective Sessions, Vision Clients), and DIAGNOSTICS (Execution, Logs, Threads). The 'Logs' option under DIAGNOSTICS is highlighted. The main content area shows the 'Logs' page with a breadcrumb trail: Status > Diagnostics > Logs. At the top right, there is a 'Help' icon and a 'Get Designer' button. Below the breadcrumb, it says '52079 items' and '1 of 521'. A filter bar contains a 'Filter' input field, a 'View' dropdown set to '100', a 'Min. Level' dropdown set to 'ALL', and a 'Live Values' toggle set to 'ON'. The log entries are displayed in a table with columns: Logger, Time, and Message. The table shows several log entries from 'DefaultExceptionHandler' with error messages about headers already being written to the response. There are also logs from 'SubscriptionManager' and 'MetroWebSocket'. The bottom of the table shows a log entry from 'WebSocketConnection'.

On this page, you can then click the Settings  button on the right side of the page. This will bring up the Log Configuration window. Here, you can enter in one of the loggers that are listed on the troubleshooting page into the filter, and then change its logging level to either DEBUG or TRACE. This will cause that specific logger to report more detailed logs to the Logs page, allowing you to better diagnose the issue. If you set a parent logger to a specific level, it will set all children of that logger to the same level.



MDC Keys

MDC Keys allow you to quickly set all loggers that are related to a specific part of your Gateway. Instead of setting individual loggers to DEBUG, using the MDC key will set a group of loggers to the level you select all at once.

Using MDC Keys is very simple. Again, you will want to navigate to the Gateway Status page, and go to the logs page under Diagnostics. Click on the settings button to bring up the Log Configuration but this time click on the 'Context' tab. Here, there are a few fields where you can enter in information. The first field, **Key** is where you will enter in the appropriate MDC Key. The field will list out appropriate Keys for you as you type, making it easier to find what you are looking for.

For the **Value** field, you will want to enter in a value that is appropriate for the given MDC Key. This field also will list appropriate values for you to enter. In the example in the image below, with the Key being **database**, the value field has listed out all of the database connection names, making it easy to select which database you want to troubleshoot.

Finally, you set the appropriate logging level in the **Set Level** field, typically to **DEBUG** or **TRACE**, and then hit the **Add Level** button. You can add as many of these as you would like, even with the same Key. So in the example below, you can first add my database called MSSQL, and then using the database Key again, add in the database called MySQL. This allows you to quickly and easily set the appropriate loggers to find out more information of the system you are trying to troubleshoot.

Log Configuration



Loggers

Context

Key database

Value enter values

Set Level INFO

Add Level

Key

Database

Values

Level

Action

No Log Properties Configured

MSSQL

MySQL

D FaultedRetryDaemon 24Apr2017 10:03:05 Completed test for datasource "Database" in 1 ms, status is now: valid

D FaultedRetryDaemon 24Apr2017 10:03:05 Running test for datasource "Database" (Current status = Valid)

Development Server Best Practices

Overview

Many users find a Development Server useful so changes can be tested in a safe environment before being pushed to production. This page details several recommendations and issues to consider when utilizing a Development Server.

Device Considerations

- The Development Server should have its own PLCs. This prevents the Development Server from accidentally writing to a Tag that the Production Server is connected to.
- Device connection names should be the same on both the Development and Production Servers. OPC Tags typically reference the name of the Device Connection in their OPC Item Path, so using the same name on both servers means that Tags can easily be migrated from one server to the other.
- Using PLC simulators is an alternative to the Development Server having its own PLCs. Generally this means finding simulators for your specific devices outside of Ignition.

Licensing Considerations

- The Development Server does not require a licensed Ignition Gateway, but one may be necessary depending on your testing requirements. All modules can run on the trial for a two-hour testing window, and may be started an unlimited number of times. If you want a more flexible test system for your Development Server, you can add a license to it.

Database Considerations

- The Development Server should have its own **database**, separate from the production database. This is especially useful when designing screens that manually modify a database table.
- The names of the database connections used by the Development Ignition Gateway should match the names of the connections on the Production Server. Many system functions that interact directly with the database, such as **system.db.runPrepUpdate**, take the name of the

database as a parameter, so using similar names for connections on both servers means you will not have to modify the script once it has been moved to the Production Server.

Network Considerations

- Some users prefer to place their Development Server on a separate network from the Production Server. This way there can be no confusion as to which server is being accessed. Keep in mind that Clients can only be launched on panels/computers that have network access to the host Gateway, so how this development network is configured will impact where testing clients can be launched.
- If you are restoring a copy of your Production Server on your Development Server, you will need to be cautious: two copies of the same Gateway, on the same network, will both have access to all PLCs and Databases. In this case it is possible for duplicate historical records to be generated, among other potential problems. To prevent this, we recommend keeping the Development Server on a separate network. Alternatively, you could utilize the **Restore Disabled** checkbox when restoring on the Development Server. Once the disabled Development Gateway is up and running, you can edit each connection and point them to testing equipment.

Transferring Resources from Development

- When you are ready to move resources from the Development Server to the Production Server, Gateway Backups should not be utilized as they will include connections to the development resources. Instead, use the **Project Exports** and **Tag Exports** functionality to transfer resources.
- In cases where there are multiple Ignition Servers (Production or Testing), the **Enterprise Administration Module** can make the resource migration easier.

Defining SSL on a Perspective Reverse Proxy

Reverse Proxy Use Cases

While a forward proxy proxies on behalf of clients (or requesting hosts), a reverse proxy proxies on behalf of servers. There are many possible use cases for placing Perspective behind a reverse proxy:

- Control what resources the user is able to access
- Direct traffic to a portion of the web server
- Limit which projects are accessible to users via subdomains
- Define an SSL certificate on the reverse proxy

For this example, we will use [NGINX](#) to define an SSL certificate and key on a reverse proxy. NGINX is a popular choice for reverse proxies, though other open source servers such as [Varnish](#), [Apache](#), and [Traefik](#) provide similar functionality.

Installation

Install NGINX or NGINX Plus following the relevant [installation instructions](#). Obtain a [Signed Security Certificate](#) (SSL) Place the SSL certificate and key in the directory where you installed NGINX under `etc/nginx/ssl/`

Configuring NGINX

In this example, we will create an `nginx.conf` file that specifies the configuration of our server. This section details the modules and properties that are required to configure Perspective behind a reverse proxy; for a more general overview of a proxy server configuration, see the [NGINX Admin Guide](#).

Required NGINX Modules

Module	Documentation
ngx_http_core_module	https://nginx.org/en/docs/http/ngx_http_core_module.html
ngx_http_proxy_module	https://nginx.org/en/docs/http/ngx_http_proxy_module.html
ngx_http_ssl_module	https://nginx.org/en/docs/http/ngx_http_ssl_module.html

Websocket Proxying

For compatibility with Perspective's Websocket connections, we change the HTTP version to 1.1. In order to pass **hop-by-hop** headers from the client to the proxied server, the "Upgrade" and "Connection" headers must be passed explicitly in the configuration file:

```
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
```

Custom Headers

In cases where a reverse proxy or other middleware is located between the Gateway and a running Perspective session, the proxy server will need to whitelist custom headers. Doing so ensures that the proxy allows requests from the session to pass through to the Gateway. To allow requests with custom headers, you will need to enable `proxy_pass_request_headers` in your configuration file:

```
location / {
    proxy_pass https://localhost:8043/data/perspective/client/demo/;
    proxy_http_version 1.1;
    ...
    proxy_pass_request_headers on;
    ...
}
```

Additional Location Contexts

Perspective sessions require resources within several folders that must be defined as additional location contexts within the configuration file. The following additional locations are required to enable

the Perspective session to run the reverse proxy:

```
server {  
  ...  
  location ^~ /data/ {  
    ...  
  }  
  location ^~ /system/ {  
    ...  
  }  
  location ^~ /res/ {  
    ...  
  }  
  location ^~ /idp/ {  
    ...  
  }  
  location ^~ /.well-known/ {  
    ...  
  }  
}
```

Example Configuration

nginx.conf

```
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    server_name demo.xxx.com;

    underscores_in_headers on;

    # Specify the paths to your SSL certificate and key:
    ssl_certificate /etc/nginx/ssl/reverse_proxy.crt;
    ssl_certificate_key /etc/nginx/ssl/reverse_proxy.key;

    # The next few location contexts define how to process resources required by
the Perspective session.
    # If the longest matching prefix location has the "^~" modifier then regular
expressions are not checked.
    location ^~ /data/ {
        proxy_pass https://localhost:8043/data/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_pass_request_headers on;
        proxy_cache_bypass $http_upgrade;
    }

    location ^~ /system/ {
        proxy_pass https://localhost:8043/system/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_pass_request_headers on;
        proxy_cache_bypass $http_upgrade;
    }

    location ^~ /res/ {
        proxy_pass https://localhost:8043/res/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_pass_request_headers on;
        proxy_cache_bypass $http_upgrade;
    }
}
```

```
location ^~ /idp/ {
    proxy_pass https://localhost:8043/idp/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_pass_request_headers on;
    proxy_cache_bypass $http_upgrade;
}
```

```
location ^~ /.well-known/ {
    proxy_pass https://localhost:8043/.well-known/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_pass_request_headers on;
    proxy_cache_bypass $http_upgrade;
}
```

*# Here we are setting the protocol type, address, port, and uri (optional)
that will be the destination of our proxied server:*

```
location / {
    proxy_pass https://localhost:8043/data/perspective/client/demo/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_pass_request_headers on;
    proxy_cache_bypass $http_upgrade;
}
```

```
}
```

Optimize Gateway Network Performance

Gateway Network Services

The [Gateway Network](#) is an Ignition to Ignition communication network, allowing you to connect multiple Ignition Gateways together. Although licensing is unlimited, the limitations on hardware can cause performance problems. This is where splitting modules onto separate Ignition servers or reconfiguring an existing architecture becomes helpful or even necessary to better connect sites, share and aggregate data, increase visibility, and centralize management.

Separating out functionality not only improves performance, but can also increase security and simplify scaling out systems in the future. Splitting servers between Back-End and Front-End is a common way to dedicate servers to separate functionality where the Back-End may contain the drivers to the PLC, the tag historian, and alarm notification, but the Front-End handles applications.

See the [Scale Out Architecture](#) User Manual page for more information on scaling out a system.

System Baseline

Since the services included in the Gateway Network work to distribute features and load, fully utilizing the Gateway Network resources allows you to better detect and eliminate bottlenecks in your system if you understand your connection baseline. Everything you need to monitor your system baseline is on the Gateway Network status page where you'll find queue statistics and throughput information. Use this information to anticipate how changes will affect memory and performance and give you a head start on adjusting your system settings as you build out and grow your system.

You should also familiarize yourself with the remote services you have active and what settings are adjustable. Remote services like tags, tag history, and alarm history should be frequently checked to make sure each service is maximizing its performance potential.

Alone, these settings won't make or break your system performance, but they all stack up, especially in larger systems, and give you many courses of action for optimization. It is important to check all servers in the network to understand throughput and load throughout.

Remote Tag History

Remote tag history is possible through the remote tag provider or through a remote tag history provider. Both options query through the Gateway Network, but use different tag paths. Remote tag providers use realtime tag paths and remote tag history providers use historical tag paths.

Remote Tag Providers connect to a remote installation of Ignition and access tags, making remote tags a local representation of tags that exist on the remote system. By default, providers are set to read only, but can be changed to read and write in order to continue preserving memory, without limiting control. Remote tag history sends historical data to another server to be stored, such as a corporate database. Remote tag history functionality extends to querying historical data as well.

Two aspects to check in your tag history query process are whether you're using fully qualified historical tag paths and querying history directly against an SQL database for maximum performance. Since tag history can be queried from multiple sources, it's important to be aware of where tag history is queried from. Tag history queries can be slow, which may increase load on the receiving side and cause bottlenecks on the Gateway Network. Since Historical tags use fully qualified historical tag paths, they display provider information. However, realtime tags do not specify which tag history provider to query from, and therefore may be querying through the Gateway Network without your control.

You can check provider information displayed on the **Config > Tags > History** page to confirm the provider **Type** and make sure you're querying through your desired location.

Provider Name	Enabled	Type	Description	Status
History	true	Datasource History Provider		Running
Splitter	true	Tag History Splitter		Running
ignitionb_History	true	Remote History Provider		Running

The above image also shows there is an option to incorporate a splitter, which allows you to create a database to mirror data so that it can be queried locally. Using a **splitter** means the data is available on both sides. This enables a faster response time and eliminates a potential need to control who has access on either side. For instance, you will not need to worry about users on a corporate server putting additional load on the Network outside of standard operation needs.

If you are querying through realtime tag providers, make sure your History Settings have **Database** selected as the **History Access Mode**.

Remote Tag Provider Alarm Subscription

Remote tags are a very efficient service by default, but can be more useful if you adjust the Alarm Mode for systems split between Back-End and Front-End servers. Alarms should already be visible on the Back-End and are queried by default. Selecting **Subscribed** for the **Alarm Mode** pushes alarm notifications immediately on the Front-End server.

Alarm Settings	
Alarms Enabled	☒ (default: true)
Alarm Mode	Subscribed ▼ How alarm state should be monitored. In 'queried', state will be queried through the Back-End and will be sent asynchronously. Subscribed provides better performance, but uses more memory. (default: Queried)

This is a manual selection because depending on the amount of alarms, it could cause a delay due to the higher memory use. You must first determine whether subscription benefits outweigh the added load on the Gateway Network. Although the delay will be relatively short, it may cause confusion to an operator or other Front-End user.

Enterprise Administration Module (EAM)

All agents configured to the central EAM controller send diagnostic data based on a set interval. If you have a high number of servers in your Gateway Network, you'll have a high throughput of requests. To counteract this, you can lower the interval to better control queues and optimize overall function. This interval can be modified in the Agent Settings by accessing the agent under the **Config** section on the Gateway Webpage and selecting **Enterprise Administration > Agent Settings**. Here you can adjust the **Send Stats Interval** to a lower number to send statistics less often.

Agent Settings

→ [Uninstall Agent](#)

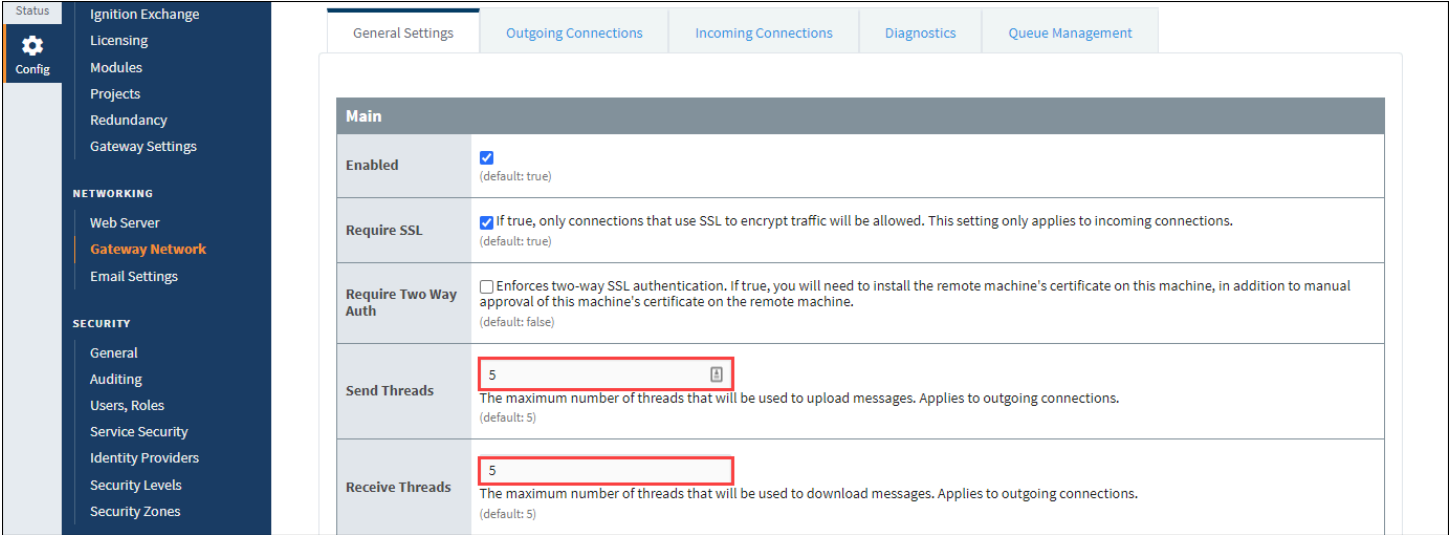
Main	
Gateway Network Server	hub ▼ The name of the Controller that holds a connection to the controller Gateway.
Send Stats Interval	5 The interval in seconds that the agent will send its statistics to the controller Gateway.

Save

Queue Management

A great way to understand your system’s data flow is to closely review your Gateway Network Status page, checking queues and thread pools. Queue statuses can illuminate issues with large numbers of alarms or tag history queries you might not be expecting to see. Maxed out thread pools or a high number of active and pending requests in the queues can unnecessarily slow performance.

Once a thread pool maximum number of requests is reached, the local server will be unable to respond to anything else until those are processed. By default, thread pools have a limit of 5 requests. The maximum limit can be adjusted for different workloads in the **Send Threads** and **Receive Threads** settings under **Config > Network > Gateway Network Settings > General Settings**. Although you don’t want infinite threads, which can cause a crash, you can raise or lower this limit to clean up your system. For example, if there are a lot of expected requests for tag history, increasing the limit will reduce issues the local server may experience.



Also, if you are querying tag history through the Gateway Network, adjusting the Max Active number in the Queue Settings may help avoid bottlenecks or system shutdowns. Navigate to **Config > Network > Gateway Network Settings** and select the **Queue Management** tab to change the **Max Active** amount from unlimited to a number that best suits your system.

Queue Settings	
Queue Name	Tag History
Description	Retrieves tag historical data over the Gateway Network
Synchronous Delivery	☐ This setting is configured by the queue and is unchangeable. If true, the queue will not use asynchronous delivery. Note that when a queue uses synchronous delivery, the maximum number of allowed active tasks is limited to 1. (default: false)
Max Active	2 The maximum number of active tasks allowed at a time. A task is considered active when it is in the process of being executed. This can set a limit to ensure that the Gateway Network connection will not become overloaded. (default: -1)

Note this may cause errors on the Front-End in the same way as the maximum limit in a thread pool prevents the local system from processing other requests, reaching the Max Active task limit will appear as an error.

Ping Rate and Latency

The Gateway Network continuously sends ping messages to monitor connection health. Ignition settings allow monitoring of ping response times for both outgoing and incoming connections. The Diagnostics tab on the Gateway Network page can be used to test remote server response times on both sides to help identify potential latency issues. Slow responses may indicate the need for tuning, as servers can perform poorly due to high latency, maxed-out threads, or high request loads.

When multiple consecutive pings time out, the connection is marked as faulted, which may indicate a server issue. Persistent faulting causes the connection to automatically close and reopen, which can degrade performance due to repeated reconnections, particularly in large networks. Adjusting ping rates on the Gateway Network Settings page can stabilize the system. For example, consider setting the Incoming Ping Rate to 10 seconds, and adjust the Outgoing Ping Rate to twice that (20 seconds). Adjusting the Ping Timeout, to 1–2 minutes for example, may help manage intermittent connection issues while allowing timely fault detection.

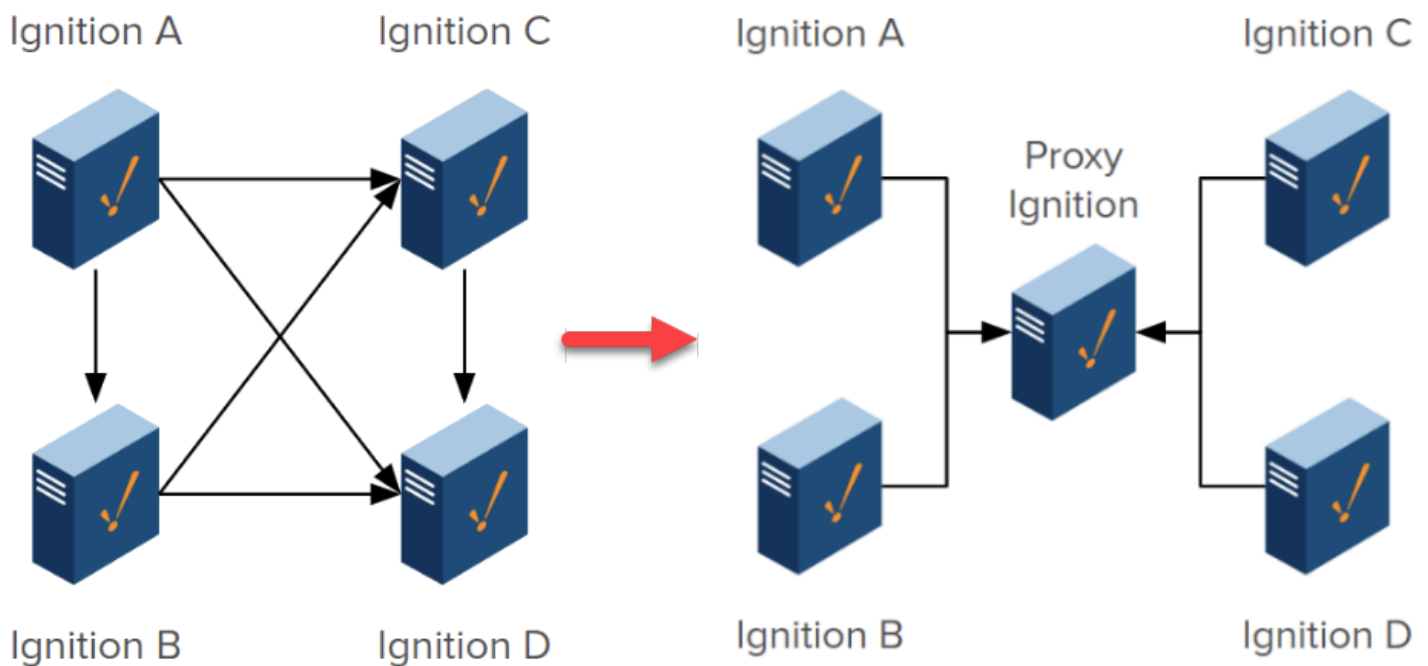
In addition to ping monitoring, idle requests will appear on the Stats page, showing details about service enumeration, subscription validation, and incoming and outgoing data bytes. Observing this data can help identify bottlenecks or unusual patterns that may signal underlying network issues requiring immediate troubleshooting.

Secure Connections

Proxy Nodes

Depending on what you want for your architecture, the use of proxy nodes may be essential. Proxy nodes make architecture simpler by removing complexity in connections and avoiding busy connection paths. Proxy nodes are most effective in highly segmented networks. For instance, when working with an Operations network and a Business network that are unable to talk directly, you can create a proxy Ignition server as a middle layer. There is no configuration required beyond outbound connections from the sites to the proxy server to allow data transfer between them.

Placing Ignition as a proxy to take advantage of the Gateway Network features in this way also strengthens security since simpler networks require less firewalls to maintain. In the example below, the use of a proxy server eliminates many unnecessary connections to increase stability.



Connection Security

Although direction does not matter for Ignition, be mindful of IT practices. It is recommended to connect on one side only and connect upwards. Using the fewest connections possible helps IT lockdown and secure connections. Let's consider the example of connecting a local Ignition outgoing connection to a central Ignition platform. This single connection will still be bidirectional, but setting up in this way prevents exposing the OT network to inbound connections and requires just one port to be open in a firewall. Also, Ignition can handle further communication through a mesh network as needed.

You can also use whitelists to limit incoming connections. Doing so ensures the proxy only accepts requests to pass through to the Gateway if they originate from an approved server. Note that using whitelists on a proxy still requires approving each individual server even if you previously accepted a root CA certificate for the servers, as the approvals fulfill separate purposes.

Service Security

Service Security defines the security policies established at the source for what data is accessible. It can define if a specific service is available at all, read-only, read-write, and so forth regardless of connection direction. Zero-trust is not the default security policy, but it should be highly considered since being strict about security is the best way to control the flow of data. A zero-trust policy will reject all unwanted requests. To update your Default policy to zero-trust, navigate to **Security > Service Security** and select **Edit**. Use the dropdown on each Alarm section to select **Deny**.

Alarm Journal Access	
Service Access	Deny
Default Profile Access	Query Only
Alarm Notification	
Service Access	Deny
Accessible Pipeline Filter	A comma separated list of alarm pipeline names on this gateway (which can include the wildcard **) that will be made available for use in the alarm notifications of other gateways in this security zone.
Alarm Status	
Service Access	Deny
Allow Acknowledge	☒

Now, you can define new security zones and edit zone-based settings as needed for your system to allow wanted requests. See the Tag Access image below for an example.

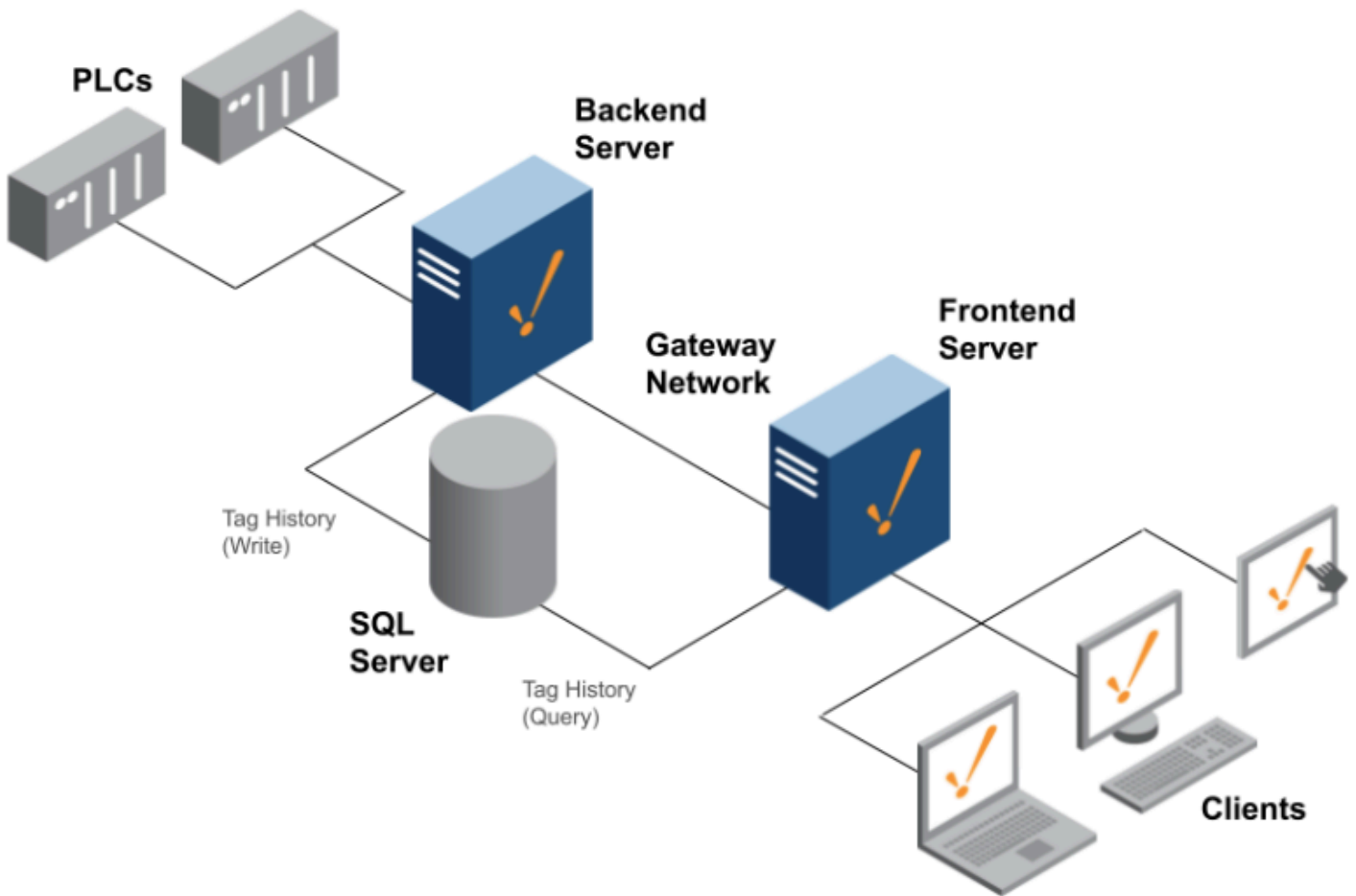
1. Navigate to **Security > Security Zones** and select **Create new Security Zone**.
2. Enter a zone name and fill in any of the three Identifier fields to add services to the zone.

3. Return to the Service Security page.
4. Edit your policy settings to choose what fields you want to enable and how.

Tag Access	
Service Access	Allow
Default Provider Access Level	ReadWrite
Trust Remote Security Levels	☒ If checked, security levels passed from the remote Gateway will be used for determining access to tags on the local Gateway. If unchecked, or if the remote Gateway is on a version which does not support this feature, the remote Gateway's security zones and the impersonation role will be used as the security levels.
Impersonation Role Name	This role name will be injected into the security model of the tag for read/write/subscribe/edit operations.
Access Level: 'default'	Inherited
Access Level: 'System'	Inherited

Scale-Out Architecture Guide

As stated on the [Scale Out Architecture](#) page, a scale-out architecture can help ease the load between Front-End and Back-End tasks by separating them into their own Ignition installations. Since Ignition provides various options in regards to Gateway to Gateway subsystem communication, some configuration routes will be more optimal than others. If you're planning to scale out an Ignition server or would like to get started, review these guidelines to determine the route that works best for you.



Migrating Resources

It is **strongly recommended to first create a test environment** for migrating your resources before making changes on your production environment. Doing so will help you take note of any special scenarios that need to be addressed in the [Considerations for Each Resource](#) section.

1. Take a Gateway Backup and store it in a secure location - not on the server's drive!
 - i. You may want to download a copy of the installer for the existing Ignition instance for disaster recovery preparedness.

2. Determine the new purpose of the existing Gateway. For this guide, it is assumed that the existing Gateway will be repurposed as the Back-End server. However, there are instances where this may not be the optimal solution. For example, if you have 100 clients all pointing to the existing server to provide visualization, it may be easier to create a new Back-End Gateway.
3. Export all visualization/Front-End elements from projects in the Back-End.
 - i. Perspective views/Vision windows.
 - ii. **Images and Icons**
 - iii. **Reporting resources**
 - iv. If you are restoring a full Gateway backup instead of importing individual resources, ensure that the backup is restored with Restore Disabled set to True. This will keep all duplicate Back-End resources inactive so that they can be deleted later.
4. If you're planning to scale out using **Alarming**, **Auditing**, **Reporting**, **SQL Bridge**, or **Tag History**, consider the following on the new Front-End server:
 - i. Create a matching database connection by profile name and connect URL.
 - ii. Create matching Alarm Journal or Audit profiles by both profile name, database connection name, and database table name(s) if needed.
5. On the Front-End server, create the appropriate user sources or identity providers by both profile name and domain or endpoint URLs.
 - i. Ensure that all custom security levels or user mappings are properly redefined.
 - ii. If your original server was exposed to any external authorization services (Active Directory, OpenID Connect, etc.), consider the **different options** for both Front-End and Back-End login configurations.
6. On the Back-End server, establish a Gateway Network connection to the Front-End.
 - i. It is recommended to establish an outgoing connection from the Back-End server
7. On the Front-End server, create Remote Tag Providers that correspond with the Back-End's Realtime Tag Providers.
 - i. Match the tags by name so that migrated projects can continue to work without changing any Tag Provider references.
 - ii. Set **Alarm Mode** to **Subscribe**.
 - iii. Depending on your database exposure and Tag History usage, ensure the **History Settings** are configured appropriately.
 - iv. See Step 9 regarding Service Security if tags are not visible/writeable.
8. On the Front-End server, import the exported visualization/Front-End resources from step 2.
9. On the Back-End server, create a new **security zone profile** and specify under Identifiers > IP Addresses the IP address of your Front-End server.
 - i. This will provide a basic security feature that enforces exclusive access between your Front-End and Back-End servers.

10. On the Back-End server, configure **service security** and to allow read access of all necessary subsystems.
 - i. Allowing write access on your service security will depend on your requirements for Alarm Journals, Auditing, Tags, and Tag History on the Front-End server.
11. If all previous steps are successful, implement any Back-End message handlers and Front-End sendRequest scripts that may be required for **Reporting**, **SFC**, system functions with Gateway scope access, **Web Dev**, etc.
12. If you are creating multiple Front-End servers behind a load balancer, configure a DNS entry to hit the load balancer.

Considerations for Each Resource

Before you make any changes to your production Gateway, take a moment to review your projects and configurations on every module that is currently in use, especially after you performed a test migration. This is an important step since you will need to take note of which modules, project components, and configuration profiles should be considered an exclusive Back-End resource or Front-End resource.

Alarming

What to keep on the Back-End:

- Alarms configured on various tags
- Alarm Pipelines
- Notification Profiles

What to move to Front-End:

- Perspective views and Vision windows monitoring Alarm Status Table events.
- Perspective views and Vision windows monitoring Alarm Journal Table events.
 - This will require your Front-End server to have access to the same database tables, which can be accomplished using one of two methods.
 - The more optimal approach is to allow the Front-End Gateway to create a direct connection to the Back-End's same database.
 - A Remote Alarm Journal can be created which will query alarm journal events over the Gateway Network.
 - When creating an Alarm Journal profile on the Front-End, ensure the profile name is the same as the Back-End.

- When creating an Alarm Journal profile on the Front-End that will access the Back-End's database directly, ensure both **Table Name** and **Event Data Table Name** are defined in the advanced properties match the name of any existing Alarm Journal Tables.

☒ **Show advanced properties**

Advanced	
Table Name	alarm_events The table name for the core event table. (default: alarm_events)
Event Data Table Name	alarm_event_data The table name for event data associated with alarms. (default: alarm_event_data)

- Create a Remote Gateway Notification profile if you require alarms to be configured on the Front-End.
- Keep in mind, this will also require alarm notification modules to be present on both servers instead of only one.

Auditing

What to keep on the Back-End:

- Existing Auditing profiles and their database tables.

What to move to Front-End:

- Perspective views and Vision windows monitoring audit events.
 - This will require your Front-End server to have access to the same database tables but it can be accomplished using one of two methods
 - The optimal approach is to allow the Front-End Gateway to create a direct connection to the Back-End's same database.
 - A second option is to create a Remote Audit Profile which will query audit events over the Gateway Network.
 - When creating an Audit profile on the Front-End, ensure the profile name is the same as the Back-End.
 - When creating an Audit profile on the Front-End that will access the Back-End's database directly, set the **Auto Create** setting to **false** and ensure the **Table Name** property defined in the Database Settings matches the name of an existing audit table.

Database Settings	
Database	DatabaseName ▼ The database connection to use to store audit events.
Auto Create	☐ If true, the table schema specified here will be auton (default: true)
Pruning Enabled	☐ If false, this audit profile will never prune records, re (default: false)
Table Name	AUDIT_EVENTS The name of the table to store audit events. (default: AUDIT_EVENTS)

Enterprise Administration Module (EAM)

EAM does not have any special requirements. Keep the EAM controller on the Back-End and distribute the agent configurations to other remote gateways as usual.

Identity Providers (IDP) and User Sources

What to keep on the Back-End:

- A primary, internal-type user source.
 - Retaining a local authentication strategy ensures accessibility in the event of production issues since the Back-End server is primarily communicating with local, on-premise devices.
- Existing user source and IDP profiles that communicate with external authorization services (Active Directory, OpenID Connect, etc.) will depend on your requirements for accessibility and security.
 - If the Back-End server is required to avoid any direct requests from public networks, keep in mind that these types of user sources and IDP strategies require Back-End networks to be open to the public as well.
 - Also keep in mind that Back-End servers will typically limit access to administrative accounts and not operators which reduces the pool of users that need to be configured.

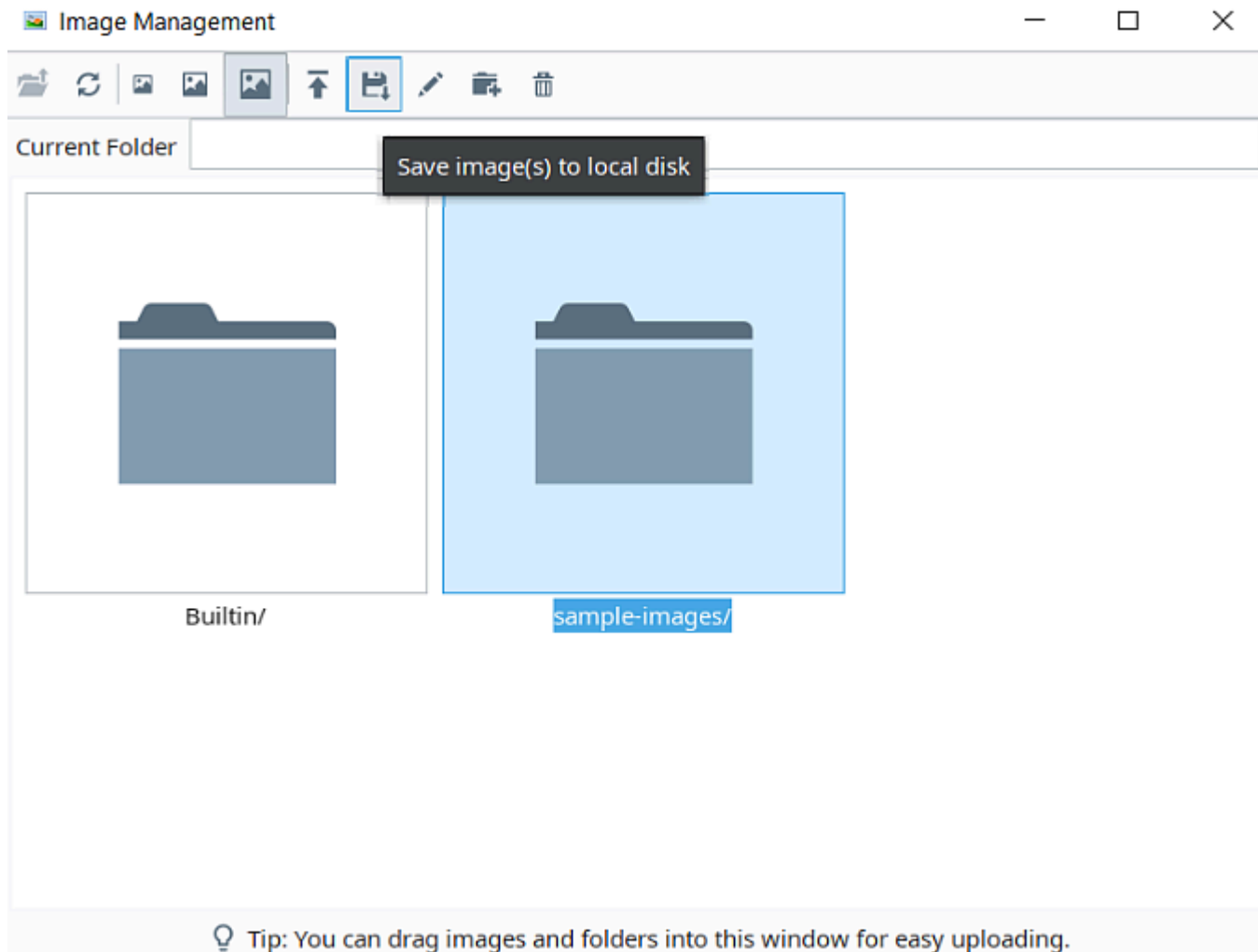
What to move to Front-End:

- Any user sources and IdP profiles that are utilized in Vision or Perspective sessions.

- Existing user source and IDP profiles that communicate with external authorization services (Active Directory, OpenID Connect, etc.).
 - If your original server already utilized these strategies, your Front-End clients will benefit from using them. Especially if the server will be public-facing.

Images and Icons

Custom images in the Back-End server's **Image Management** will need to be manually imported into the new Front-End Gateway. The Image Management tool allows the option to **Save image(s) to local disk** which includes individual files or nested files within folders.



OPC Drivers

Device connections can be left alone on the Back-End.

Reporting

If only one Front-End Gateway exists, reporting resources can reside on either Gateway.

- The more optimal approach will depend on the requirements of your Front-End clients:
 - Do you need to display these reports for clients/sessions?

- Do you prefer having reports stored on a shared drive? Only Back-End or Front-End storage?
- Which server would you prefer for executing report construction?
- Reporting resources to consider:
 - Report profiles in your projects
 - Perspective views and Vision windows using the Report Viewer.
 - Database tables uses for reports
 - Reporting modules

What to keep on the Back-End:

- If you plan to have more than one Front-End Gateway, keep scheduled reports in the Back-End to ensure single executions without any duplicates.
 - In this scenario, migrating schedule reports to Front-End would require redundant report configurations, resulting in unnecessary report duplicates and modules.
- If reports will be stored on the Back-End or on a shared drive, all reporting resources can be left alone.
- If reports are not meant to be viewed by the client, all reporting resources can be left alone.

What to move to Front-End:

- If reports need to be displayed in clients/sessions:
 - Perspective views and Vision windows using the Report Viewer.
 - If Front-End reporting requires Tag History data, review the [Tag History](#) section.
 - If Front-End will handle creation, execution, and storage of reports:
 - Reporting Modules
 - Report profiles in your projects
 - Database tables used for reports

Sequential Function Charts (SFC)

What to keep on the Back-End:

- If your Front-End server and clients require interaction with an SFC, consider implementing a [message handler](#).

What to move to Front-End:

- To interact with a Back-End server's message handler, consider using the scripting function [system.util.sendMessage](#) wherever you see fit.

SQL Bridge

What to keep on the Back-End:

- Existing transaction groups and their corresponding database connections.

What to move to Front-End:

- Perspective views and Vision windows querying from transaction group-stored data. This will require your Front-End server to have direct access to the same database.

Miscellaneous

Third party modules are not developed by Inductive Automation and should be reviewed on an individual basis to determine the appropriate Gateway for them.

What to keep on the Back-End:

- Scripting functions such as `system.eam`, `system.device`, `system.opc`, `system.report`, `system.sfc`, etc. possess Gateway scope access but cannot reach across remote Ignition Gateways. Any use of these functions within scripts will need to be left on the Back-End.
 - If your Front-End server and clients require interaction with these scripts defined in the Back-End, consider implementing a [message handler](#).

What to move to Front-End:

- To interact with a Back-End server's [message handler](#), consider using the scripting function `system.util.sendMessage` wherever you see fit.
- Third party Python libraries are typically imported for constructing complex data structures and objects that go beyond the capabilities of our system functions.
 - These types of libraries and dependent, user-defined scripts should be imported to the Front-End server. If left on the Back-End, the scripts' execution could potentially interfere with other Back-End processes since they share the same resources.

Scripting Examples

Message Handlers

A useful application of a message handler could involve execution of reports on a Back-End server triggered by a Front-End server request. Below is a button script defined for a project that could be modified to use a message handler:

```

# Get the date range entered by the user.
startDate = self.parent.getChild("Start
Date").getChild("DateTimeInput").props.value
endDate = self.parent.getChild("End Date").getChild("DateTimeInput").props.value

# Define the settings and parameters for the report.
settings = {"path": "C:\\Reports", "fileName": "Test Report %s.pdf" %
system.date.format(system.date.now(), "yyyy_MM_dd_HH_mm_ss"), "format": "pdf"}
parameters={"StartDate": startDate, "EndDate": endDate}

# Execute the report and save it in the specified path.
system.report.executeAndDistribute(path="Test Report", project="Test_Project",
parameters=parameters, action="save", actionSettings=settings)

```

On the Back-End server's Gateway-scoped message handler, the script could be modified to:

```

# Get the report settings and parameters from the payload.
settings = payload['settings']
parameters = payload['parameters']

# Execute the report and distribute it.
system.report.executeAndDistribute(path="Test Report", project="Test_Project",
parameters=parameters, action="save", actionSettings=settings)

```

On the Front-End server, the project button script could be migrated to this location and be modified to interact with the Back-End server's Gateway message handler:

```

# Get the date range entered by the user.
startDate = self.parent.getChild("Start
Date").getChild("DateTimeInput").props.value
endDate = self.parent.getChild("End Date").getChild("DateTimeInput").props.value

# Define the settings and parameters for the report.
settings = {"path": "C:\\Reports", "fileName": "Test Report %s.pdf" %
system.date.format(system.date.now(), "yyyy_MM_dd_HH_mm_ss"), "format": "pdf"}
parameters={"StartDate": startDate, "EndDate": endDate}

# Send a message to the backend gateway so that it can
# execute and distribute the report.
system.util.sendMessage(project='Test_Project', messageHandler='Execute And
Distribute Report', payload={'settings':settings, 'parameters':parameters},
remoteServers=['BackendGateway'])

```

When using `system.util.sendMessage` to send a message to a Gateway message handler, errors will be logged in the Gateway (accessible on the Gateway Webpage). If there is a need to return a value from the message handler to the Front-End Gateway, then `system.util.sendRequest` can be used rather than `system.util.sendMessage`.

Third Party Libraries

A useful application of third party Python libraries could involve the `croniter` library. This library provides iteration for datetime objects with cron-like format. Below is an example that utilizes the `croniter` library to figure out the next scheduled datetime based on a schedule in the cron format:

```
def getNextScheduledDatetime(cronSchedule):
    """
    Reads a schedule in the Cron format and
    returns the next scheduled datetime
    based on the current time.

    Aargs:
        cronSchedule: A string that contains
        the schedule in the Cron format
        (example: 10 14 * * 1).

    Returns:
        The next scheduled datetime according
        to the schedule.
    """
    from croniter import croniter
    from datetime import datetime

    cron = croniter(cronSchedule)

    # Get the next scheduled time in unixtime.
    nextScheduledTime = cron.get_next()

    # Convert the unixtime into a datetime and return it.
    return datetime.utcfromtimestamp(nextScheduledTime) # Current datetime is
11/11/22 4:55pm (Friday)
# The following Cron schedule represents 2:10pm every Monday: '10 14 * * 1'
# So, when calling getNextScheduledDate('10 14 * * 1'), we get 11/14/22 2:10pm
(Monday).print getNextScheduledDatetime('10 14 * * 1')
```

A possible scenario that requires the script and its **dependencies above to be placed in the Front-End server** could involve large sets of Cron time values being queried from a database and then being converted to datetime for displaying on a client/session.

- If a Back-End server is already responsible for retrieving very large datasets containing a number of Cron values, consider the time and resources needed to perform both an expensive query and execution of getNextScheduledDatetime() per iteration of the returned query's dataset.
- Separating these types of tasks to individual Gateways ensures dedication of resources.

Tag History

What to keep on the Back-End:

- Existing historical tags and their corresponding database connections.

What to move to Front-End:

- Create a Remote Tag Provider with the same name as the Back-End's Realtime Tag Provider hosting the historical tags.
- All Perspective views and Vision windows for tag history.
 - To display tag history, your Front-End server needs access to the data. This is accomplished through the Remote Tag Provider's **History Access Mode**. History Access Mode is set to **Gateway Network** or **Database**.
 - Gateway Network: Tag History Bindings will request data over the Gateway Network connection.
 - Database: Tag History Bindings will request data directly from a database. The Front-End server will need a Database Connection created to the same database being utilized for the Back-End server's Tag Historian for this Tag Provider. Because only a single History Datasource can be specified for the Remote Tag Provider, the Back-End server's Tag Provider should not have tags using various History Storage Providers. If multiple Storage Providers are used, the tags should be grouped and moved to separate Tag Providers on the Back-End server. Corresponding Front-End Remote Tag Providers can then be created and configured with corresponding History Datasources.

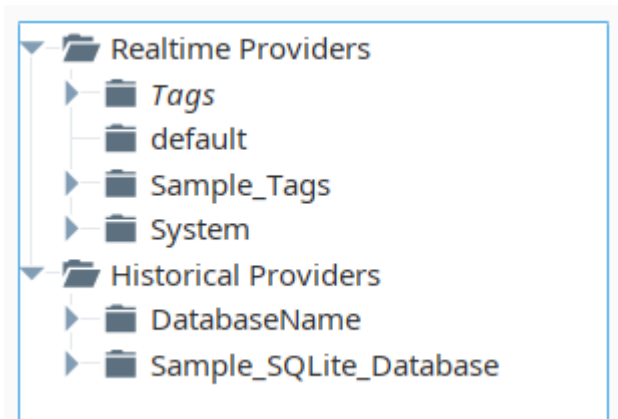
History Settings	
History Access Mode	Database ▼ This setting dictates how tag history is queried for. You can choose to query that directly. (default: GatewayNetwork)
History Datasource	DatabaseName ▼ If History Access Mode is 'database', this is the database name.
History Driver	OriginalSourceGatewayName If querying the database directly, this is the gateway name.
History Provider	OriginalTagProviderName If querying the database directly, this is the name of the tag provider.

- Setting the Historical Access Mode to Database provides a more direct line of communication for sending queries compared to Gateway Network; however, all historical tags being referenced on Front-End component bindings may need to be changed to **historical** tag paths since **realtime** tag paths are used by default.
 - Historical data will not appear in the result of your queries if tag history bindings are using realtime tag paths against Remote Tag Providers. When Remote Tag Providers are configured to query from a database connection directly, only full historical tag paths provide the information they need to properly locate tag history.
 - In order to check for realtime tag paths in your tag history queries, check the tag path properties for any tag history configurations. If the tag path is structured to resemble a path within a Tag Provider, it is a realtime tag path. For example, `[Sample_Tags]Random/RandomInteger1`.

Tag Path
<code>[DatabaseName/originalsourcegatewayname:originaltagprovidername]random/randominteger1</code>
<code>[Sample_Tags]Random/RandomInteger1</code>

- Historical binding configurations provide browsing for historical tags. If a tag has been storing historical data, expand the browsing folders to locate the proper historical tag path equivalent. You will be able to see this by searching for paths starting with:

- A database connection name
- The original Gateway the tag resides in
- The original Tag Provider the tag resides in. For example, [DatabaseName/originalignition:originaltagprovidername]random/randominteger1] is a historical tag path.



- If Historical Access Mode is set to Gateway Network, realtime tag paths are allowed and bindings will not require any modification, however, tag history queries will need to share resources with other requests being sent through a Gateway Network connection.

Web Dev

- Leaving resources on the Back-End could require exposing the Back-End to http requests by Front-End users.
 - Take time to review which resources will be safe enough to expose to the Front-End Gateway. This depends on what's being hosted and the intended use case. For example, Tag History might be more performant from the Back-End Gateway.

Secure Disposal Guide

Environment Creation Considerations

A secure disposal plan begins at the creation of a new environment. Keeping detailed documentation of the services, locations, and where data is stored is vital to keeping information secure. In the event that the environment is removed, the proper documentation can then be referenced to ensure that all information about the project is purged and securely removed.

Ignition

Ignition is installed in the default directories for each operating system listed below. This directory includes Gateway configuration files, Ignition's internal database, project files, etc.

System	Command
Windows	C:\Program Files\Inductive Automation\Ignition
Mac	/usr/local/ignition
Linux	/usr/local/bin/ignition
ARM (zip installation)	<installation path>

Keep in mind that Ignition can be configured to take and store scheduled Gateway backups, and store them in any specified folder on the server, including to a mapped network drive. The folder location where automatic backups are stored can be set and found on the Gateway webpage, by navigating to **Config > Gateway Settings > Schedules Backup Settings > Backup Folder**.

The folder locations where Gateway backups are stored, and the Gateway backup files themselves should be included, and accounted for, in the disposal process. If they are not accounted for, then all Gateway, project, and tag data residing in the Gateway backup files could be left to be accessed, exploited, or restored by anyone with access to the directory.

For larger architectures that take advantage of remote backups with the Enterprise Administration Module (EAM), it is worth keeping track of where the agent backups are stored on the controller Gateway's filesystem. Please refer to [Agent Task - Collect Backup](#) for more information on the archive path.

CAUTION

Please note that the folder where Gateway backups are stored can be changed to multiple folders/directories over the lifetime of the environment. It is useful to run a search on the server's file system for any files with the Gateway backup file extension, .gwbk, during the disposal process of the environment.

Databases

Depending on the database service being installed the installation directory and data directory will be reflected in the respective manual. If the database is being installed on a separate environment take note of the location as well, as that system will need to be reviewed and purged in the event that the environment is removed.

I/O Peripheral Devices

Although Ignition does not store information to devices connected through the Gateway Network, such as PLCs, they should still be taken into careful consideration as a system that should be cleared when the system is no longer in use. Keeping record of the devices connected in the environment, their locations, and how to properly factory reset the device before removal is vital to ensuring that information does not end up in the wrong hands.

Prior to the creation of an environment, it's important to categorize how media will be removed in the event that the environment is modified or disposed. Systems in an environment may evolve over time and the filesystems and storage locations can change between upgrades. It is always important to document all modifications to the environment and update the media type and methods that will be taken in order to securely remove the environment. This will ensure that all media is accounted for when a user needs to remove media and that the proper sanitization decisions are followed.

Version Control

Version Control Systems (VCS) such as Git, Mercurial, or Subversion are powerful tools for managing changes to files over time. They allow tracking modifications, reverting to previous versions, and collaboration.

VCS can store a wealth of information about the Ignition Gateway, projects, tags, and more. This data can include code changes, configuration settings, and even user activity.

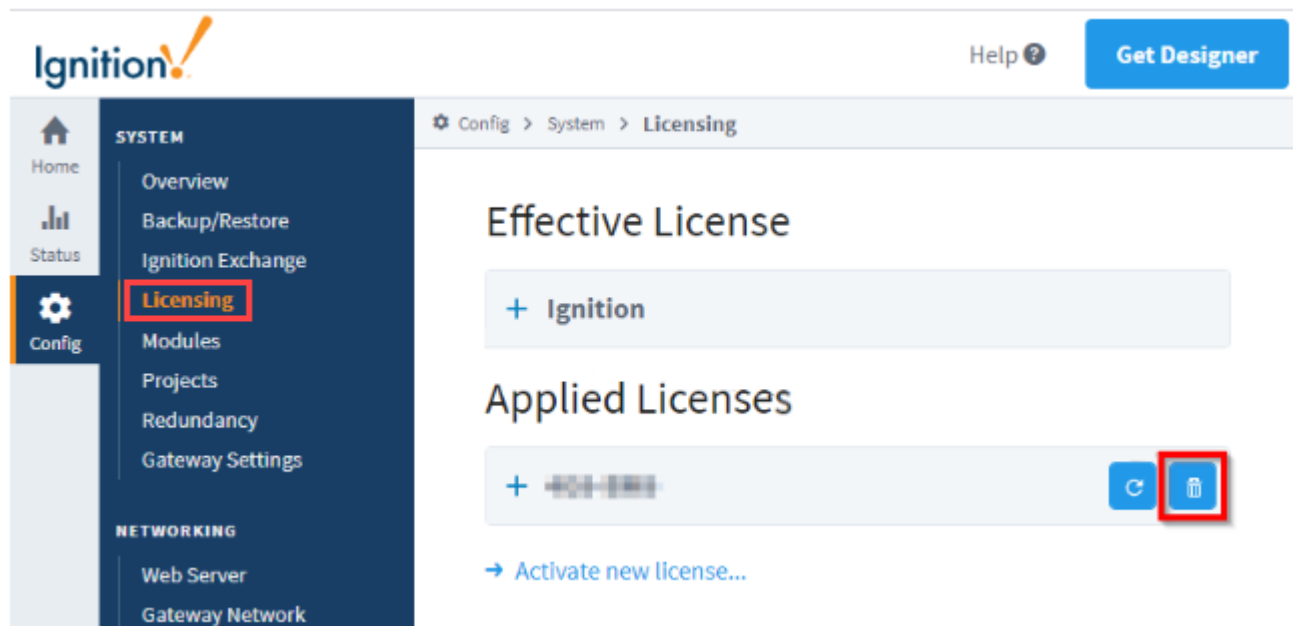
The repositories of these VCS are typically stored in specific locations on your system, network, or hosted in the cloud. It's crucial to document these locations for future reference, especially if the environment is being decommissioned or removed.

Decommissioning Ignition Removal Steps

Properly uninstalling and removing Ignition guarantees that the configurations and data stored while the environment was active are completely removed and unrecoverable by unwanted parties.

Release License

Before uninstalling Ignition, check and make sure the license has been deactivated. If not, navigate to **Config > Licensing** and select the **trash** icon to the right of the license key.



Uninstall Ignition

To uninstall Ignition, run the uninstaller file which can be found in the main Ignition installation directory.

⚠ CAUTION

Uninstalling Ignition will remove any projects or Gateway level configurations that were made. If there are projects or configurations that you would like to save, make sure to take a Gateway backup first.

System	Command
Windows	<code>C:\Program Files\Inductive Automation\Ignition\Uninstall.exe</code>
Mac	<code>/usr/local/ignition/Uninstall.app</code>
Linux	<code>'/usr/local/bin/ignition/Uninstall.run'</code>
ARM (zip installation)	<code><installation path>./ignition.sh remove</code>

Directory Removal

Once the uninstallation process has finished the Inductive Automation directory can be deleted. Please refer to the table below for the proper directory to remove.

Standard Removal

System	Command
Windows	<code>Remove-Item 'C:\Program Files\Inductive Automation' -Recurse</code>
Mac	<code>rm -r /usr/local/ignition</code>
Linux	<code>rm -r /usr/local/bin/ignition</code>
ARM (zip installation)	<code>rm -r <installation path></code>

Secure Removal

 NOTE

Secure Removal is generally preferred over Standard Removal. This table provides examples of secure removal tools. Use discretion when downloading packages and running commands.

System	Package	Command
Windows	Microsoft SDelete Package	<code>sdelete -s -r 'C:\Program Files\Inductive Automation'</code>
Mac	SRM (secure_deletion toolkit)	<code>srp -r /usr/local/ignition</code>
Linux	SRM (secure_deletion toolkit)	<code>srp -r /usr/local/bin/ignition</code>
ARM (zip installation)*	SRM (secure_deletion toolkit)	<code>srp -r <installation path></code>
*Secure removal software depends on the Operating System running on the ARM device.		

Uninstalling and Removing Databases

Proper database removal is of utmost importance when ensuring secure data sanitization. Database services and file systems will remain on the installed system even when Ignition has been stopped, uninstalled, or removed. Therefore, databases used in conjunction with Ignition should be uninstalled and removed following the removal of Ignition. Due to the array of database management software that Ignition can connect to, please follow the uninstallation guidelines provided in the respective database documentation.

Media Sanitization

Once all conduits and services have been uninstalled and removed from the environment, media sanitization is recommended in order to render access to target data unrecoverable for a certain level of given effort. There are various standards that can be followed in order to sanitize storage devices, such as **NIST Special Publication 800-88**.

Proper removal requires understanding **hard** and **soft** copy media storage, and the types of sanitization. Understanding the type of data that is being sanitized can help you understand which of the following three methods is needed in order to sanitize the media device.

- **Clear Method:** This method is the simplest sanitization method. It typically involves using the Read and Write commands standards of the operating system to remove pointers to the addressable storage space on the media. This makes it difficult, but not impossible, for the media to be recovered. This may also include factory resets, which do not always guarantee the removal of sensitive information and simply delete the file pointers instead.
- **Purge Method:** This option takes the Clear method a step further by implementing an overwrite, cryptographic erase, or block erase to help in rendering the media unrecoverable. These methods rewrite over the stored information so not only the pointers to the files are removed, but the files are unrecoverable. Purge also includes degaussing as a step to render the storage media unrecoverable. This method uses a strong magnet to clear the memory from magnetic storage devices, though it is worth noting that this method does not work on non-magnetic storage and should not be solely relied on for flash storage devices.
- **Destroy Method:** This method is the most destructive of the three options and should be used for the most sensitive information. There is always a possibility that the information cleared or purged can be recovered, or that remnants of sensitive information was overlooked and not purged from the device. This method includes pulverizing, incinerating, or shredding techniques that make the media device data irretrievable even with the use of the latest laboratory techniques.

Sanitization of Storage Devices:

- **NIST Special Publication 800-88 Revision 1, Guidelines for Media Sanitization**
- **ITL Bulletin Feb 2015: NIST Special Publication 800-88 Revision 1, Guidelines for Media Sanitization**

Tag History vs. Transaction Groups

Capturing useful data during your manufacturing processes will require you to make a decision between implementing Tag History or Transaction Groups. Such a decision will also revolve around understanding two manufacturing process types: **Discrete** and **Continuous**.

Recognizing essential aspects of your specific manufacturing process and how it pertains to one of the existing process types allows you full advantage of Ignition's data logging capabilities. This is because both data logging implementations are flexible enough to apply to the same given scenario, where one of them will typically provide a more efficient approach than the other.

On this page, you will be informed on how concepts from these process types apply towards existing manufacturing industries, and which of Ignition's data logging solutions are known to work best.

Discrete and Continuous Processes

Discrete and continuous processes are typically understood concepts by integrators and manufacturers when they define important aspects of a product's creation. When observing your own processes on the plant floor, each of them should be able to fall into one of those categories.

Discrete

A discrete process involves a specified count of units moving from one workstation to another to experience modifications. Each unit retains a unique identifier for keeping track of its progress per station. Examples of where this is primarily used can be found in the following manufacturing industries:

- Automotive Suppliers
- Steel Production
- Food and Beverage (Batching)

The categories listed above all involve a process where a finite, countable amount of units (specialized parts for either automobiles or raw uncut solid material) will undergo work with one or more workstations on the plant floor. Each workstation will keep track of start and stop timestamps per unit, where the unique identifier will allow you to request those timestamps later.

Continuous

A continuous process describes a continuous amount of material for means of ensuring consistency in production. To accomplish this, the process may run for long durations with little to no interruptions.

Examples of where this is primarily used can be found in the following manufacturing industries:

- Fiber Extrusion
- Oil and Gas
- Water and Wastewater

The categories listed above all involve material that's difficult to quantify. Somewhere in the process, a consistent rate of some material will have to run through a given workstation. Monitoring a specified rate of material for long and uninterrupted durations will ensure said consistency.

Intertwining Discrete and Continuous

Understanding which type defines the most critical parts of your manufacturing process will help you make well informed decisions during your Ignition project development. Keep in mind that although some industries, like those listed above, primarily use either discrete or continuous processes, most industries typically involve other secondary manufacturing processes that require utilizing both concepts.

For instance, water and wastewater is an industry where people typically want to know the timestamps of specific intervals for when a pump turns on or when a tank alarm goes off. Essentially, they need ways to monitor the state of their overall system at intervals of some time. However, water and wastewater cannot be quantified as units, so there must also be a way to monitor the flow rate of material, as well as any interruption events.

Recognizing Data Logging and Monitoring Methods

Ignition provides two primary means of database logging for its users:

- Tag Historian
- SQL Bridge

The Tag Historian's design allows timestamped values of individual tags to be stored asynchronously due to its use of Tag Groups and internal structure of associated SQL tables. The SQL Bridge's design provides a more event-based procedure for logging a set of tags to a user-defined SQL table.

The key differences between these two methods is how data is stored and timestamped. This is because those differences result in varying complexities for how they are queried to the type of resulting visual you are looking for.

Tag Historian

If you are considering using Tag Historian, take a moment to review where your process expects to keep track of individual event-based data capturers using various Tag Group types/intervals. Event-based data will usually focus on recording workstation interruptions or changes from its expected flow rate of a given material. In order to keep track of such events, various moving parts may need separate monitoring to generate individual datasets and trends. The list below describes where key Tag History features can apply to this kind of setup:

- **Deadband Discrete Style and Analog Compression:** Storage for tags is based on value changes and the changes themselves exceeding a user configured deadband.
 - No matter what type of historical tag you have configured, discrete or analog, deadband configurations ensure the values of each and every historical tag stores its next incoming value to your discretion.
- **Asynchronous Tag Storage:** The storage of each tag happens independently (For example: Tag A might be stored every 10 seconds due to constant value changes while Tag B might be stored twice a day due to lack of change). Once data has been stored from a successful batch, the review of trends for each and every part of your various workstations can be narrowed down to a single tag's set of data if desired without any complex SQL querying.
- **Data Partitioning:** Historical data is stored into monthly partitions by default. Historical data from different Gateways are also placed into its own partition table structure when sharing the same database. Pre-processed partition options are also available for even faster query times.
 - This structure allows for the abstraction of enterprise architectures to store large sets of tag data from multiple Ignition Gateways into a centralized database if desired.
- **Fixed Table Structure:** Multiple, tall data formatted, SQL tables are used by the Tag Historian to keep track of tags being stored, their Tag Group configuration and their Gateway origin.
 - This allows users to focus more on storage and retrieval of data without the worry of defining optimal database tables yourself.
- **Querying and Built-In Aggregation:** Aggregation such as Avg, Sum, Max/Min, DurationOn, and DurationOff can be enabled through various tag history query functions and user interfaces such as the **Power Chart**.
 - Additional to the abstractions placed during Tag History storage, that same can be said for querying the same data onto trends in your Ignition projects.
 - The Power Chart also provides a simpler way to query these trends using drag and drop functionality from tag browse lists to the components themselves.

FOR CONTINUOUS PROCESSES, IT'S RECOMMENDED TO USE TAG HISTORY

Tag Historian is more asynchronous in its data inserts and which results in a complex structure of associated SQL tables where groups of tags being recorded at the same rate will typically have timestamps that differ in milliseconds. Continuous processes will work better with Tag Historian because of this due to its ability to create independent sets of data points where each can be quickly and individually monitored for changes in its trends. Durations in a continuous process can be potentially un-ending with no start or end events. Since the most meaningful data comes from easily locating deviations in your trends, your setup may want to focus on constantly monitoring for changes, unique anomalies, and interruptions of some expected flow of material instead of logging individual sets of time-based data per event (discrete).

SQL Bridge

If you're considering SQL Bridge for your setup, take a moment to review where your process expects to keep track of time-based data. Such time-based data may involve records of when those workstations began and finished work for each unit of product which would require multiple item values in your workstations to be recorded under the same timestamp for accuracy. In addition to inserting new data into what will be a user-defined SQL table, existing row(s) may instead need replacing based on some type of event success or failure. The list below describes where key SQL Bridge features can apply to this kind of setup.

- **Transaction Group Types:** Storage of various **items** under a single timestamp, SQL updates, and OPC writes are possible with transaction group types options. SQL Bridge's Transaction Group will create a single custom table of data that receives inserts of a single timestamped row of data due to an event trigger by default.
- **Execution Triggers:** Transaction group executions will depend on the state of a single, user-defined item. Handshake mechanisms for execution success/failure are also provided to execute other events if needed.
 - All instructions between your connected devices and databases can be defined from transaction groups alone if desired.
- **Synchronous storage:** All items in a transaction group are stored together as a single SQL transaction when inserted.
 - Once your monitored events have successfully fired enough write executions, you can expect to review your table using standard SQL querying against simple table structures.
- **Customizable Table Structure:** A wide data formatted table where items defined in your transaction group will represent table columns.
 - Optimization of tabular data is in the hands of the user when using transaction groups. More importantly, a custom SQL table integrated with transaction group UI provides the most flexibility for your plant floor instructions while reducing complex SQL querying.

- **Direct SQL Querying**

- Tables both defined by a transaction group will be much simpler to understand when read or written to in your database.
- This also allows for integration of external stored procedures defined in your database if needed.



FOR DISCRETE PROCESSES, IT IS RECOMMENDED TO USE SQL BRIDGE

SQL Bridge is best used for discrete processes since the focus is placed on capturing the state of multiple tags under a shared timestamp during some given change event or instruction defined by the user. Time-based data stored in a single, user-defined table is more optimal, especially when a large dataset of logging is required for many individual tags/items per widget (countable unit), each with their own start and end time in a given assembly line. Each widget is an event in itself that would probably need to provide a litany of information within its duration on the assembly line, such as build ID, elapse build time, upstream batch IDs, length, weight, and QA pass/fail. Such a process probably doesn't need constant, unending monitoring (continuous) but instead a process of nested sub-loggers that is triggered per given unit.

Recommended Anti-Virus and Endpoint Security Settings with Ignition

As part of normal operation, some anti-virus software will attempt to restrict access to directories or ports that Ignition needs to function. This page describes some exclusions to be mindful of, but is by no means a comprehensive list. It is recommended to work with your IT Department to determine if any other whitelisting or configuration within the anti-virus software is needed.

Ignition Gateway

- Exclude the Ignition installation directory.
- Exclude the Ignition process from inspection.
- If the anti-virus software includes network traffic inspection, exclude the ports Ignition will be communicating on.

For example, ports 8088 and 8043 are commonly used default ports on new Ignition installations. However, these ports are user configurable and can always be changed, so you may need to review your Ignition configuration to determine what ports are used. More information about the ports an installation may use can be found on the [Gateway Port Reference](#) page.

- Exclude temp locations.

You should also note any temp locations that may need to be whitelisted depending on what your system requires. For instance, when working with EAM configurations, a Controller Gateway will distribute an Ignition update to an Agent Gateway via ZIP file, where the file is stored in a temp location until the whole file has arrived to alleviate issues with slow networks or intermittent connectivity. This means the update will get stuck if the temp location isn't whitelisted. Temp locations are based on what Java uses for the temp folder on the system. For Windows, the temp folder is `%USER%\AppData\Local\Temp` by default, and for Linux it is `/tmp`. You can change the location of the temp folder by adding a `java.io.tmpdir` system property to `ignition.conf`. Refer to the [Gateway Configuration File Reference](#) page for more information.

Ignition Designer

- Exclude the Ignition cache directory: `<user-directory>/.\ignition`

Ignition Vision Client

- Exclude the Ignition cache directory: `<user-directory>/.ignition`

Ignition Perspective Workstation

There are no specific exclusions for Perspective Workstation.

Accessing Projects Remotely

This page provides an overview of different methods to access your Ignition Vision and Perspective projects from a remote location. Once created, Perspective and Vision projects can be opened without additional configuration by using the Gateway's IP address. If you want more secure options, you can connect to your Gateway's local network using a VPN, expose specific ports to the internet via port forwarding, or use a reverse proxy for an additional layer of security.

Using URLs - Perspective Only

Open your web browser and navigate to your Ignition Gateway using the Gateway's IP address:

```
http://<local-gateway-ip>:8088.
```

Project Examples

HTTP

```
http://<local-gateway-ip>:8088/data/perspective/client/YourPerspectiveProject
```

HTTPS

```
https://<local-gateway-ip>:8043/data/perspective/client/YourPerspectiveProject
```

For more details on launching Perspective Sessions, refer to [Perspective Sessions](#) and [Ignition Perspective App - Launch a Perspective Session](#).

Using VPN

A Virtual Private Network (VPN) allows you to securely connect to your Gateway's local network from a remote location.

Set Up a VPN Server

1. Install and configure a VPN server on your network. Common options include OpenVPN, WireGuard, or a built-in VPN server on your router.

2. Ensure the VPN server is properly configured to allow remote connections and has the necessary firewall rules to permit traffic.

Connect to the VPN

1. Install the VPN client software on your remote device.
2. Configure the VPN client with the server's address and the necessary authentication details.
3. Connect to the VPN. Once connected, your remote device will be part of your Gateway's local network.

Using Port Forwarding

Port forwarding allows you to expose the specific ports of your Gateway's local network to the internet.

Configure Port Forwarding on Your Router

1. Log in to your router's admin interface.
2. Locate the port forwarding section and create rules to forward the Ignition Gateway ports to the internal IP address of your Ignition Gateway server.
 - Port 8088 for HTTP
 - Port 8043 for HTTPS

```
External Port: 8088 -> Internal IP: <local-gateway-ip> -> Internal Port: 8088  
External Port: 8043 -> Internal IP: <local-gateway-ip> -> Internal Port: 8043
```

Using Reverse Proxy

A reverse proxy can provide an additional layer of security and simplify access to your Ignition projects.

Set Up a Reverse Proxy

1. Install a reverse proxy server on your network. Common options include Nginx, Apache, or HAProxy.
2. Configure the reverse proxy to forward requests to your Gateway.

Example Nginx configuration

```
server {
    listen 80;
    server_name <your-domain>;

    location / {
        proxy_pass http://<local-gateway-ip>:8088;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

server {
    listen 443 ssl;
    server_name <your-domain>;

    ssl_certificate /path/to/your/certificate.crt;
    ssl_certificate_key /path/to/your/certificate.key;

    location / {
        proxy_pass https://<local-gateway-ip>:8043;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Migrating Servers

This page outlines the steps to migrate an Ignition Gateway from one server to another. A common use case for this would be if you are upgrading server hardware running Ignition in order to increase performance of the Ignition Gateway by providing additional server hardware resources. If you are instead interested in scaling out your Ignition functionality into multiple Ignition servers, please refer to our [Scale Out Architecture Guide](#).

The general process steps are:

1. Collect a Gateway backup of the current Ignition server.
2. Install the same version of Ignition from your current server to your new server to restore the Gateway backup.
3. Unactivate the license key on the old server.
4. Activate the license key on the new server.

These steps are further detailed in the sections below, starting with the collecting and restoring your Gateway backup before transferring your license.

Gateway Backup Restore

Complete the following steps to get a copy of your Ignition Gateway onto the new server:

1. Take a [Gateway backup](#) from the current server.
2. Download and install the exact same version of Ignition running on the current server onto the new server. Use the [version archive](#) on our website to find your download.
3. Restore the Gateway backup on the new server. See the [Gateway Restore](#) procedure on the Gateway Backup and Restore page for more details.

License Activation

Once your Ignition Gateway backup is loaded on the new server, you can migrate the license activation over.

1. [Unactivate the license key](#) for your Ignition Gateway on the old server.
2. Now, on the new server, you can [Activate](#) the license key again.

EMERGENCY ACTIVATION MODE

If you run into any issue with migrating the license, you will end up in [Emergency Activation Mode](#) and will need to contact your Sales Representative for assistance.

Ignition 8 Deployment Best Practices

This page describes the process of deploying Ignition applications and provides best practices for setting up your development and testing workflow. Having a solid deployment workflow allows for better change management, minimizes mistakes, and leads to a more productive team.

The following information is aimed at Ignition users who are looking to set up a scalable, flexible, enterprise-grade system. Whether you start from a different architecture or model and move toward this, these guidelines are not intended as an all-or-nothing proposition. Many installations could benefit from employing even a subset of these best practices. For any enterprise-grade system, Inductive Automation highly recommends employing all the techniques covered in this document.

Workflow

Deployments should be treated as part of a development workflow, not as an afterthought. Your workflow will usually include at least three environments: Development, Testing, and Production. In that case, the workflow might look like this:

1. Developers work on bugs and features in a Development environment.
2. Once features are implemented, they are merged into the testing branch and deployed to the Testing environment for quality assurance and testing.
3. After testing is complete, the development branch is merged into production and then deployed to the Production environment.

Before taking a closer look at each environment, it is important to look at Ignition's configuration in more detail to better understand deployment.

Ignition's Configuration

Ignition is a server-based software and is installed on a central server, which means all of Ignition's configuration is stored on the server. It is all in one place to install, license, configure, backup, and manage. Client applications are downloaded from the Ignition server (Vision) or are native HTML (Perspective) and as a result are automatically updated as new changes are deployed. That way you don't have to worry about installing, licensing, or deploying changes to individual clients, especially if there are hundreds of clients. For deployment model, focus solely on the server.

When you are developing in Ignition, there are 4 main areas of configuration:

- Gateway Configuration
- Tags
- Images
- Projects

Each of these areas are stored differently on the server. The Gateway backup contains the configuration for all of these but some can also be handled individually. It is critical to understand the differences and how you interface with them when deploying changes.

Gateway Configuration

The Gateway configuration area includes all of the different settings, profiles, and connections you edit in the Configuration section of the Gateway webpage. This is where most of the settings that affect the whole Gateway are set up. You can add database and device connections, users and roles, adjust alarm settings, set up security, create a schedule for a Gateway backup to be taken automatically at specific times, and much more. The list of configuration options on the left menu changes based on what modules are installed on your Gateway.

All of these settings are stored inside of Ignition's internal SQLite database. They are only accessible by the Gateway user interface. Outside of a full Gateway backup, you cannot export or import these settings from Gateway to Gateway. Ignition's EAM Module currently doesn't support moving these settings from Gateway to Gateway either. This makes it tricky to work with when developing your deployment strategy since all changes must be made manually. It is also difficult to track changes since everything is inside of a SQLite database.

Gateway backups are not normally used when migrating changes because they are all-or-nothing. Different environments often have separate Gateway configuration settings for items such as devices, databases, and authentication.

NOTE

We recommend creating a process that all developers follow to track changes of these settings outside of Ignition and to document the differences in these settings within each environment (development, testing, and production).

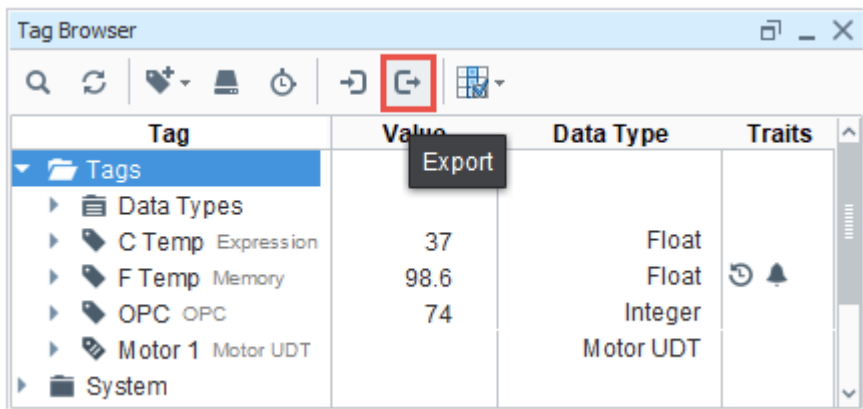
Tags

Tags are an important part of most projects. Tags are points of data and may have static or dynamic values that come from an OPC address, an expression, or a SQL query. Tags provide a consistent

data model throughout Ignition and offer the easiest way to get up and running creating real-time status and control systems. However, despite their fast initial learning curve, tags offer a great amount of power in system design and configuration. The ability to aggregate tags from a variety of installations means you can build widely distributed SCADA systems more easily than ever before with a high level of performance and relatively simple configuration.

Tags are not part of an Ignition project. Rather, projects simply reference tags. Tags are stored inside of Tag Providers. A Tag Provider is a collection of tags and can be local or remote. An Ignition server can have one or more Tag Providers associated with it and can provide logical groupings of tags. By default, Ignition is shipped with a local Tag Provider called “default”. Local Tag Providers are stored inside of Ignition’s internal SQLite database. Tags are configured inside of the Ignition Designer in the Tag Browser panel.

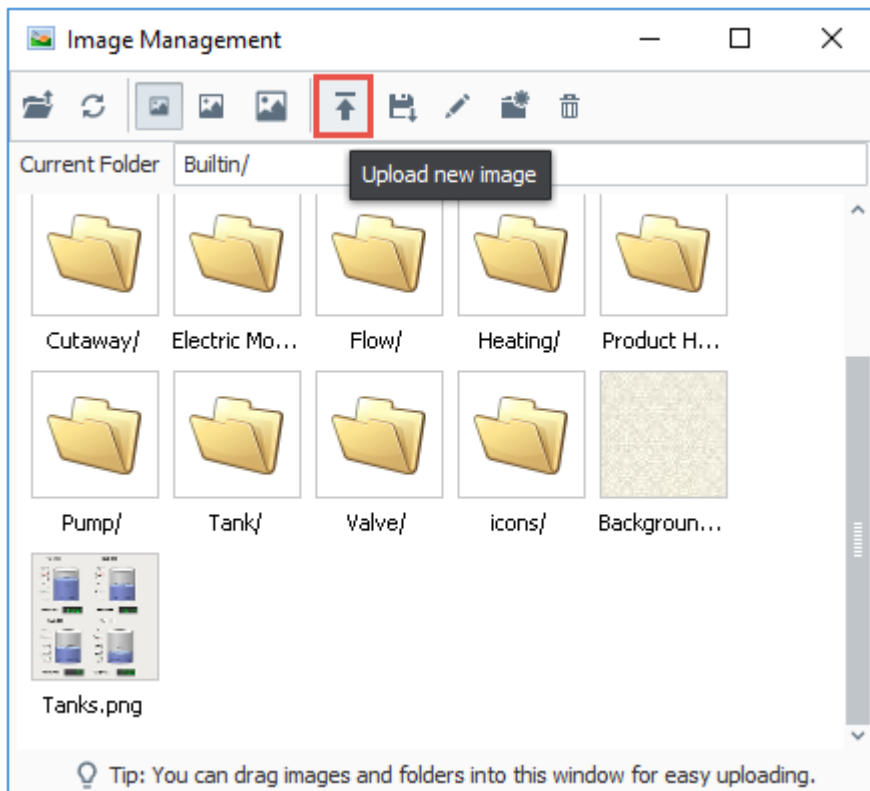
Ignition can export and import tag configurations to and from a JavaScript Object Notation (JSON) file format. You can import Extensible Markup Language (XML) or Comma Separated Value (CSV) file formats as well, but Ignition will convert them to JSON format.



Tags can also be moved through Ignition’s EAM Module from Gateway to Gateway. Tags are much easier to work with when deploying changes. However, tags are still stored inside of Ignition’s internal SQLite database and steps must be taken to track changes of tags. The process of exporting tags can be done automatically using a Python script in Ignition. You can either use Ignition’s EAM Module to deploy changes from development to testing to production, or manually export/import through the Ignition Designer.

Images

The Image Management tool, available from **Tools > Image Management** in the Ignition Designer, provides an interface to upload, download, or select images. Images such as PNGs, JPGs, GIFs, and SVGs can be uploaded in the Image Management Tool and used inside of Vision or Perspective. These images are stored inside of Ignition’s internal SQLite database and are not a part of an Ignition project. Instead, projects simply reference images.



Images cannot be moved through Ignition’s EAM Module from Gateway to Gateway. Continuously exporting images to a folder and committing the folder to a source control repository to track changes of images. This process cannot be done automatically and requires manual intervention. You have to export/import images through the Ignition Designer when deploying changes from development to testing to production.

Projects

Projects are the main unit of configuration in Ignition. Projects hold all the designed elements that do the real work. Your projects can hold both interactive elements like controls, charts, reports, entry forms, and more. Projects also hold persistent elements like historical loggers, and automated reports. An Ignition project includes but is not limited to:

- Vision Windows and Templates: Screens for user interaction.
- Perspective Views: Screens for users interaction.
- Transaction Groups: A bi-directional link between databases and PLCs.
- Reports: PDF reports for displaying and recording data.
- Scripts: Timer and event-based scripts used throughout the system.
- Alarm Notification Pipelines: Pipelines that notify users of alarm conditions.
- General settings and properties: Settings that control access, resource connections, layout, timing, and more.

These elements are created and configured in the Ignition Designer. The projects are then viewed in the runtime (Vision Clients or Perspective Sessions). You can create as many projects as you want

where users can jump between projects on the fly or open multiple projects at the same time.

Projects are stored in the file system as a series of folders and files. Some files are binary, while others are stored as plain text files. Projects are stored inside the data directory on the Ignition server.

```
/var/lib/ignition/data/projects
```

```
C:\Program Files\Inductive Automation\Ignition\data/projects
```

Since projects are stored in the file system, it is possible to use best-in-class source control tools outside of Ignition to track changes of projects, such as Git. Changes in the Ignition Designer get updated in the file system. Ignition also continuously monitors the file system for changes and automatically reads in the changes.

Projects can also be imported and exported through a file (.zip extension) and moved from Gateway to Gateway using EAM.

NOTE

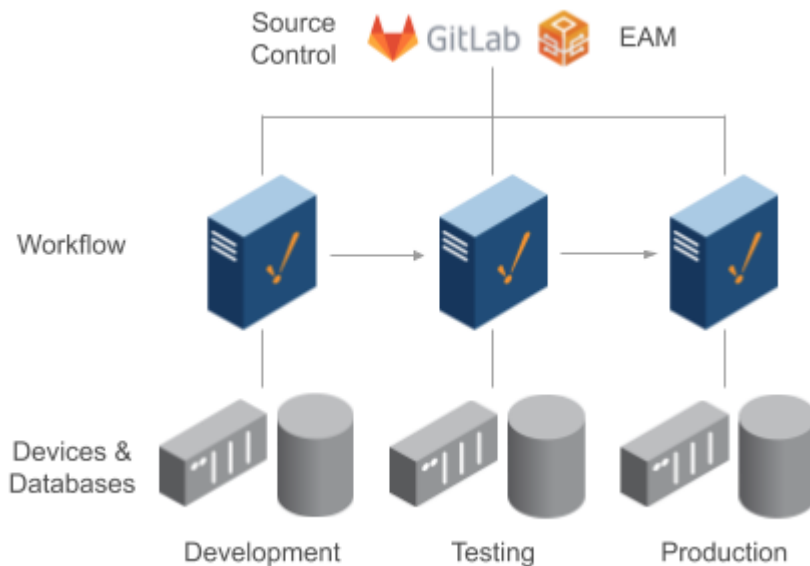
We recommend using an external source control tool to track project changes, such as Git. You can turn the project folder into a repository and use tools to deploy changes from Development to Testing to Production environments outside of Ignition.

Change Track Summary

	Gateway Configuration	Tags	Images	Projects
Change Tracking	Manual Documentation and Gateway Backup, then commit to Version Control	Tag Export, then commit to Version Control	Image Export, then commit to Version Control	No export required. Auto-commit project folder changes to Version Control
Backups	Gateway Backup	Gateway Backup	Gateway Backup	Gateway Backup

Deployment Environments

Now that you have a good idea of how Ignition's configuration is stored and works, let's take a closer look at each environment to see what are the most efficient ways to deploy to each one. In the following reference architecture you will see the Ignition Development, Testing, and Production environments, along with Ignition's EAM Module and a source control system such as GitLab.



Development Environment

The Development environment is the environment in which active Ignition development is occurring. This environment is the first step of the workflow where developers are working on bug fixes and new features before introducing them to the Production environment. It is very handy to have a separate branch in your source control system called development to represent your development environment.

There are two main camps on how to approach the Development environment:

- **Shared development environment:** A shared development environment is a central Ignition server dedicated to development that all developers work on.
- **Individual workstations as their own development environment:** An individual development environment is one where each developer has Ignition installed on their workstation with their branch of the project.

There are pros and cons to each approach. Ignition supports concurrent development and most customers tend to go with a shared development environment so nothing has to be installed on each developer's workstation. They can instead launch the Designer from the development server. If an individual development environment approach is taken, each developer's workstation will need to have its Ignition Gateway configured separately. Regardless of the approach, changes will need to be committed to the source control repository and deployed through the workflow.

Testing Environment

Once the features are implemented and considered fairly stable, they get merged into the testing branch and then deployed to the testing environment. This is when quality assurance begins. Testers go to testing servers and verify that the project works as intended.

It is very handy to have a separate branch in your source control system called testing to represent your testing environment. It will allow developers to deploy multiple branches to the same server simultaneously, simply by merging everything that needs to be deployed to the testing branch. It will also help testers understand what exactly is on testing servers at the moment, just by looking inside the testing branch.

Production Environment

Once the feature is implemented and tested, it can be deployed to production. The production environment is also known as live, particularly for servers, as it is the environment that users interact with directly.

NOTE

We recommend always deploying major releases to production at a scheduled time that the whole team is aware of.

Considerations

Consider the following factors when deciding your deployment environment approach:

- You do not need the development and testing environments to be licensed. You can simply use Ignition's trial license. Some customers elect to purchase licenses for these environments to remove having to reset every 2 hours. The choice is completely yours.
- It is extremely important to understand the differences in Ignition's configuration for each environment. They cannot be exactly the same. The production environment is communicating to live data sources such as devices and SQL databases and more. You may not want your development and testing environments connected to the same live data sources. You also may have different authentication sources or permission models on the development and testing environments. The following is a short list of common differences:
 - Devices or PLC connections: Using Ignition's native drivers.
 - OPC UA connections: Connecting to third-party such as Kepware or a PLC directly.

- Database connections: Connecting to a SQL database such as MySQL or MS SQL.
- Authentication profiles: Different authentication sources or especially a different admin or root password. Sometimes you want to lock out access to the Designer and Gateway in the production environment.
- Alarm notification profiles: Connections to email, SMS, or voice.
- Redundancy settings: Often you have redundancy on production and not development or testing.

 CAUTION

We recommend keeping the same connections on each environment, but with different IP addresses or settings behind the scenes. That way any resources dependent on knowing the name will still function from environment to environment. For example, you may have a dedicated SQL database for each environment. All three Ignition Gateways will have the same database connection by name, but they will point to their dedicated SQL database that may be local or remote. Screens or named queries using that connection will function across all environments since they are referencing the connection by name. Document these differences to ensure you don't deploy a change to production that points to a development or testing system.

- Since you want to have access to live data in all environments to understand limitations or performance issues, you'll need to find a process for simulated real device data. For example, simulated data can help you discover PLCs that don't support more than 1 connection or would get saturated with multiple Ignition servers polling the same tags multiple times. There are several ways to accomplish getting access to real data:
 - OPC UA: You can connect your development and testing environments to your production environment through OPC UA. The development environment would have the same OPC server name but it would point to production instead of pointing to itself. Be careful, as it is possible for development and testing to affect production, especially with writing to tags.
 - MQTT: This is the recommended way of getting live data to all environments. You can have your production environment publish all tag data to a central MQTT server using Ignition's MQTT Transmission Module. All 3 environments can subscribe using Ignition's MQTT Engine Module and tags are automatically discovered. Each environment can subscribe without affecting the other. Again, be careful about potentially writing to live PLCs. MQTT servers have ACLs (access control lists) you can define to make the development and testing environments read-only if needed.

- Gateway Network: You can connect your development and testing environments to your production environment through Ignition's Gateway Network and set up remote tag providers. The development environment would have the same tag provider name but it would point to production instead of being local. Again be careful about potentially writing to live PLCs. Ignition's Gateway Network has service-level security to make the development and testing environments read-only if needed. This method is not generally recommended.
- Device Simulation:
 - Programmable Device Simulator: This is perhaps the best option if you don't need live PLC data. Instead, you can use Ignition's programmable device simulator to simulate PLC data. The simulator allows you to define any tag path you want along with static or dynamic values. Your development and testing environments will use the simulator while the production environment will use the real PLC. As long as you keep the name of the device the same, the tags will be identical across all systems. See the [Programmable Device Simulator](#) page for more details.
 - SFCs: If you need more advanced device simulation, SFCs offer deep flexibility. The Programmable Device Simulator provides configurable signals, but the signals don't change based on a user's actions. SFCs, paired with memory tags, provide a great way to simulate basic PLC functionality like HOAs and valve statuses, flows, and more.
 - PLC Software Simulator: Many PLC manufacturers offer a soft PLC/emulator that can run a real PLC program. If Ignition can connect to the software via OPC, this can be a good option to have the real PLC program with all the tags running in a system that's completely disconnected from the real equipment.
- Additional PLCs: If you have a system with a small number of PLCs, it can sometimes make sense to have separate physical PLCs for the development and QA environments, especially if PLC programming changes may be rolled out at the same time as Ignition project changes.
- Each environment typically has its own dedicated SQL database. You don't want development and testing working off the same database as production. Generally, this is not a problem for Ignition's built-in history, alarming, auditing, and transaction groups. However, if you build your own custom tables and interface Ignition with those tables, you'll have to think about how to ensure those tables (structure) and possible data exist in all environments. Commonly, this is done manually by developers by running the same DDL statements in all environments. In some cases, there are tools or replication techniques that can be used by the database system to automate this process. You need to come up with a plan to ensure all three environments have the same structure and possibly data such as lookup tables.

Enterprise Administration Module

The EAM module provides a secure and intuitive way to manage many Ignition installations from one location. It is ideal for large enterprises that deploy multiple Gateways across vast geographical

distances, but even companies with a few Ignition Gateways on a single plant floor can benefit from its ability to monitor performance, and automate backup and recovery from a central location.

With the EAM module installed, an Ignition Gateway can function as the central controller Gateway that connects with, monitors, and manages any number of remote agent Gateways. Use the controller Gateway to coordinate and automate many administrative tasks for agent Gateways:

- Check health and diagnostics
- Assigning, updating, activating, and deactivating licenses
- Deploying new and updated modules
- Disaster recovery
- Remote backups
- Remote Ignition upgrade

EAM Controller Gateway and Best Practices

It's recommend to either set up a dedicated EAM controller Gateway or use the development environment as the controller. It is advised to use EAM for health and diagnostics, licenses, backups and disaster recovery, and upgrades. EAM can be used to move tags from the controller to any agent, and for any simple migration of project resources from one networked Gateway to another. For users preferring full source control of Ignition projects leveraging external version control systems, see the [Source Control System](#) section below.

Source Control System

A Source Control System is a version control system designed to track changes in source code and other text files during the development of a piece of software, such as Ignition. This allows the user to retrieve any of the previous versions of the original source code and the changes which are stored. Source control systems provide many benefits:

- Maintain multiple versions of code
- An ability to go back to any previous version
- Developers can work in parallel
- Audit traceability with a clear picture of whom, which, when, where, and what the changes are
- Synchronize the code
- Copy/Merge/Undo the changes
- Find out the difference between versions
- Provides full backup without occupying much space

- Review the history of the change
- Capable for both small and large-scale projects
- Ability to share and work on the code across the globe

There are many source control systems and we recommend using the one you are most comfortable with. GitLab and Git is a very popular open source control system.\

Source Control and Ignition

We recommend storing all Ignition resources in a source control system including projects, tag exports, images, and Gateway backups. Projects are best managed through a source control system since they are stored in the file system. It is extremely easy to push and pull projects to and from a source control system. However, it is important to define a process for storing and deploying tags, images, and Gateway configuration since they are stored in Ignition's internal SQLite database and not individual files.

It is extremely easy to get started using a source control system. We recommend installing a source control system on a dedicated server or VM, or using an existing source control system that's already set up in your company. Simply set up a project in the source control system for each unique Ignition production environment.

Project Branches

A branch is a version of a project's working tree. You create a branch for each set of related changes you make. This keeps each set of changes separate from each other, allowing changes to be made in parallel, without affecting each other.

After pushing your changes to a branch, you can:

- Create a merge request to merge changes from one branch to another
- Perform inline code review and preview changes
- Discuss your implementation with your team

Since there are three separate environments in your workflow, it is recommended to set up three separate branches for each project:

- Main: The main branch used for production
- Testing: The branch used for testing. No development occurs on this branch but changes are merged from development to testing for QA.
- Development: The branch used for active development

Now that you have your project and branches set up, let's define how to commit our changes to the repository.

Committing and Deploying Changes

Active development is occurring in the development environment. Once the features are implemented and considered fairly stable, it is time to merge them into the testing branch and then deploy them to the testing environment. We recommend storing all Ignition configuration to the repository. Let's go through each configuration item outlined above: projects, tags, images, and Gateway configuration.

Projects

Projects are stored in the file system which makes it easy to commit to a source control system. Simply make the Ignition project folder the repository. You can find the projects folder on the Ignition server in this location:

```
/var/lib/ignition/data/projects
```

```
C:\Program Files\Inductive Automation\Ignition\data/projects
```

If the repository on the source control system is empty, simply commit the existing folder to the repository. If no projects exist on the development system, simply pull the repository from the server to add the projects. Make sure to ignore the .resources folder in your repository. You don't want that directory committed to the repository since it will be unique for each system.

As changes are being made, commit and push the changes from the development environment to the repository using the proper tools (GUIs, command line tools like Git). When you are ready to deploy the changes to testing, you can merge the changes from development to testing and pull the changes from the repository to the projects folder on the testing server. See the [GitLab example](#) for more details.

Tags

Tags are stored inside of Ignition's internal SQLite database so they don't exist as files that can be committed to the repository. However, you can export tags to a file. The two methods for storing tags to the source control system are to manually export or use an Ignition Python script to automatically save the file to the repository. In the manual method, the developer can export tags to a JSON file in the Ignition Designer and save the file in the project folder when they are ready to push their changes to the repository. With this method, you can export only when needed. Conversely, in the automatic

method, you can add a Python script on the **Update** Gateway Event script that executes on every Designer save to export tags to a JSON file and save the file in the project folder. See the [GitLab example](#) for more details on the script.

Regardless of the method, it is important to commit the changes to the repository in order to track changes from version to version. We recommend storing the tags.json file to the following location:

```
<IGNITION INSTALL DIRECTORY>/data/projects/tags.json
```

The file will be ignored by Ignition. Then, when you are ready to deploy the changes to testing, you can merge the changes from development to testing and either use Ignition's EAM module to deploy tag changes or import the tags into the testing server in the Designer. If using the EAM module to deploy tag changes, see the [EAM in the Designer](#) page for more information.

Images

Images are stored inside of Ignition's internal SQLite database so they don't exist as files that can be committed to the repository. However, the developer can export images in the Image Management Tool in the Ignition Designer to a folder inside of the projects folder when they are ready to push their changes to the repository. Make sure to select all images and folders in the Image Management Tool. We recommend creating a .images directory in the projects directory, and storing the images in the newly created directory:

```
<IGNITION INSTALL DIRECTORY>/data/projects/.images
```

Generally, you would not want to create folders in the projects directory since the Gateway assumes every folder is a project. However the Gateway will ignore any hidden directories, or those that start with a "." character.

When you are ready to deploy the changes to testing, you can merge the changes from development to testing and import the images into the testing server in the Designer.

Gateway Configuration

Gateway configuration is stored inside of Ignition's internal SQLite database and doesn't have any export and import functions. With that being said, we recommend storing the latest Gateway backup (.gwbk) to the repository. Here the developer can save a Gateway backup in the Gateway configuration web page when they are ready to commit their changes to the repository.

We recommend creating another hidden directory in the projects directory to store the latest Gateway backup, similar to how images are stored in the previous section. Call this new directory

“.gateway_backups”:

```
<IGNITION INSTALL DIRECTORY>/data/projects/  
.gateway_backups/Ignition-backup-XXXXXXXX-XXXX.gwbk
```

When you are ready to deploy the changes to testing, you can merge the changes from development to testing and manually make the necessary Gateway configuration changes on the testing server.

Define Your Workflow and Process

You need to define your own workflow and process that is understood by everyone. Make sure to address all of the questions listed above. Document the process and provide each developer with a cheat sheet. We also recommend creating a checklist when deploying to ensure accuracy.

GitLab Example

This section describes how to configure version control via GitLab. The process involves two procedures.

- **Installing Git and Publishing Project** which demonstrates how to publish a project to a repository.
- **Setting Up Version Control** which demonstrates how to integrate Ignition with a version control system.

If Git is already installed, and your project already resides in a git repository, then skip the Installing Git and Publishing Project section.

Installing Git and Publishing Project

1. Install the Git Command Line Tools:
 - Linux: from terminal, type `sudo apt-get install git`
 - Windows: <https://git-scm.com/download/win>

CAUTION

The rest of this guide assumes a Linux environment. You may need to adjust commands and scripts in later steps to run on Windows

1. To identify yourself in Git, run the following commands:

```
git config --global user.name "YourUserName"
git config --global user.email "your@email.addr"
```

To publish projects to the repository:

1. Navigate to the following directory: `<IGNITION INSTALL DIRECTORY>/data/projects`
2. Add a .gitignore file to ignore the .resources directory: `nano .gitignore`
3. Add the following line to the .gitignore file: `.resources/`
4. Save the file. Now run the following commands to initialize and commit the repository: `git init`
`git remote add origin http(s)://root:TOKEN@URL/root/PROJECT.git` `git add .` `git commit -m "Initial commit"` `git push -u origin main`
5. After the initial commit, set the branch to main for future pushes. `git branch --set-upstream-to=origin/main`

You have now successfully committed your projects. Next, you need to export and commit tags, images, and Gateway configuration. For tags, export the tags from the Designer into a tags.json file. For images, use the Image Management Tool in the Designer to export all images to a folder called "images". For Gateway configuration, simply make a Gateway backup. Store all the file and folders into the projects directory. Once added, use the following commands to commit them to the repository:

```
NOW=$(date +"%m-%d-%Y %H:%M:%S") git add . git commit -m "Designer save @ $NOW"
git push origin
```

Setting Up Version Control

1. Install GitLab. Follow the [GitLab Install instructions](#) to install GitLab on a central server or VM.
2. Setup your Root Password. You will use this root user when using Git.
3. Generate a Personal Access Token. Follow the [GitLab Personal Access Token instructions](#) below to create a token for auto-publishing. Make sure to check all scopes.
4. Set up a new project (repository) from the Dashboard by creating a new project and entering a desired name. The project will be empty by default.
5. Using GitLab's web interface, create a development and testing branch, based off of the main branch (or the branch created in the [Installing Git and Publishing Project](#) procedure).
6. Navigate to the following directory to install Ignition on dedicated development and testing machines or VMs:

```
<IGNITION INSTALL DIRECTORY>/data
```

7. Remove the projects folder that already exists. It is empty since it is a fresh install. Once removed, run the following commands in that directory:

```
git clone http(s)://root:TOKEN@URL/root/PROJECT.git projects
```

8. Navigate into the projects folder now that it is created again. Run the following commands to switch to the development or testing branch:

```
git checkout development git branch --set-upstream-to=origin/development
```

i NOTE

Switch “development” with “testing” for the testing branch on the testing server.

9. Make sure to import the tags and images in the Designer. You can either use EAM to import tags or import manually in the Designer.
10. Next, manually merge in all Gateway configurations, such as database or device connections and authentication profiles.
11. Now that you have the development environment set up and the projects have been pulled from the development branch, you need to set up Ignition to automatically commit changes when a developer presses save in the Ignition Designer. Start by creating a batch/shell script in the data directory called “git-auto-commit.sh” or “git-auto-commit.bat”. For example on Linux:

```
/var/lib/ignition/data/git-auto-commit.sh
```

The file will contain the following lines:

```
cd /var/lib/ignition/data/projects NOW=$(date +"%m-%d-%Y %H:%M:%S") git add . git  
commit -m "Designer save @ $NOW" git push origin
```

i NOTE

The above script may need to be modified on Windows environments

12. Make sure the file has executable permissions.
13. Open the Ignition Designer and select **Project > Gateway Events > Update**.

14. Add the following script to call the batch/shell script and commit the changes to the repository on every save. See the [Gateway Event Scripts](#) page for more details.

```
import time
time.sleep(5)
system.util.execute(["/var/lib/ignition/data/git-auto-commit.sh"] )
```

15. Save the project and check out the repository in GitLab to see the commit.
16. Next, it can be handy to automatically export tags and save them to the repository as part of the save process. For that, simply adjust the Gateway Event Update script to the following:

```
import time
filePath = "/var/lib/ignition/data/projects/tags.json"
system.tag.exportTags(filePath=filePath, tagPaths=["[default]"])
time.sleep(5)
system.util.execute(["/var/lib/ignition/data/git-auto-commit.sh"])
```

NOTE

Adjust the tag paths to include all of your tag providers you want to export. Tags will get exported and saved to a file called tags.json inside of the projects directory on every save. The commit occurs after the export so the file will get committed to the repository.

17. Once you are done with your development you need to deploy the changes to the testing environment. First, merge changes from the development branch to the testing branch inside GitLab.
18. Open the UI and create a merge request from the development branch to the testing branch. By default, it will select the master branch so make sure to change that to the testing branch. Make sure to uncheck the “Delete source branch” checkbox so you don’t remove the development branch. See GitLab’s documentation for more details.
19. Once the merge request is created, you need to approve the request and actually merge the changes to testing. Once the changes are merged, deploy your changes to Ignition on the testing server.
- i. Deploying Projects: For projects, you need to pull the changes from the repository. Run the following command on the testing server. Once the changes are pulled down, Ignition will automatically update the project resources. Clients will update automatically. If you already had the Ignition Designer open, you will need to press **File > Update Project** to pull those changes from the server. `git pull`
 - ii. Deploying Tags: The tags.json export file will get pulled from the previous step. You can either manually import the tags into testing or use EAM to deploy tag changes.

- iii. Deploying Images: The images will get pulled from the repository from the previous step. You will have to manually import the images into the Image Management Tool in the Ignition Designer.
- iv. Deploying Gateway Configuration: You will have to manually apply any Gateway configuration changes to the testing environment. This includes changing settings, adding database connections, adding device connections, and more. This is arguably the hardest part of deploying changes from development to testing and from testing to production.

 DANGER

DO NOT restore the Gateway backup when deploying testing or production as there are differences in each environment.

Ignition Historian Guide

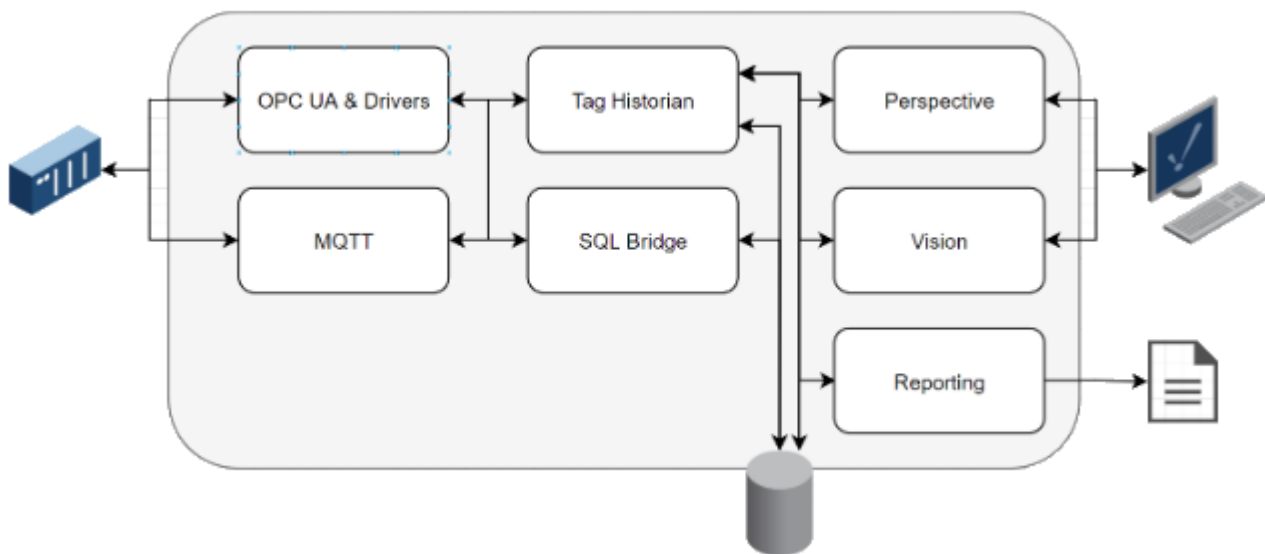
Ignition's data historian functionality is a robust set of features that are built into Ignition modules, providing data acquisition, storage, retrieval, and visualization.

As with everything else in Ignition, historian functionality is modular. Several modules are often used together to provide a complete system. Often, users combine Ignition's historian functionality with SCADA, HMI, IIoT, ERP, or MES functionality.

Although many people think of Ignition's Tag Historian Module as the "Ignition Historian," a full data historian often is described by not only storing tag data, but also storing event-based data and providing visualization. As such, this document covers the modules for those features.

Modules

One possible configuration of modules when Ignition is used as a data historian.



Ignition's modular architecture allows for the choice between using a single module or using multiple modules for a given installation. The bare minimum needed to use Ignition as a tag historian is the Tag Historian Module. However you can incorporate more modules, especially if event-based storage is needed or direct device communication will be used. The modules that are used in a typical system are outlined next.

Tag Historian Module

This provides storage and retrieval of historical data through Ignition's tag system. Incoming tag changes can be piped to data storage, and tag history can be retrieved through the module.

- **Storage:** Storage of data by the Tag Historian Module can go to several locations. Internal storage is available to store small amounts of data within Ignition itself. SQL-based storage is the normal recommendation for most systems and pipes data through to an attached SQL database. Additional storage engines are available from third-party developers, some of which can be found on the [Module Showcase](#). Regardless of the storage engine, the Tag Historian Module offers tools for data storage rates, compression, deadbands, and a number of other features.
- **Retrieval:** The Tag Historian Module can be used to retrieve history and perform calculations and aggregations on historical data. Depending on the storage engine being used, the Tag Historian Module will either perform the retrieval directly and perform any appropriate aggregates, or in the case of using third-party storage engines, the Tag Historian Module will pass the request through to the storage engine.

SQL Bridge Module

This module provides Transaction Groups, which are used for event-based logging. Data is stored into SQL database tables and options are provided for storing quality codes, timestamps, auto creating configured tables, data rates, scheduling, and more.

OPC UA & Driver Modules

These modules provide data collection. The OPC UA Module acts as the foundation for Ignition's drivers, which offer native communication to a large number of devices. It should be noted that Ignition can also connect to third-party OPC servers for data collection, either through the platform for OPC UA connections or through the OPC COM Module for OPC DA connections.

MQTT Transmission, Distributor, and Engine

These IIoT modules can be an integral part of data collection when using modern architectures. MQTT Engine plugs right into Ignition's Tag Historian Module to feed data. It also includes event storage through DRecords, which place data in database tables to create records similar to Historical Transaction Groups.

Perspective/Vision Module

Sometimes visualization is part of a requirement for data historians. Perspective and Vision are visualization systems that provide simple, easy-to-use screen building tools to create dashboards, desktop applications, web pages, and on-screen reports.

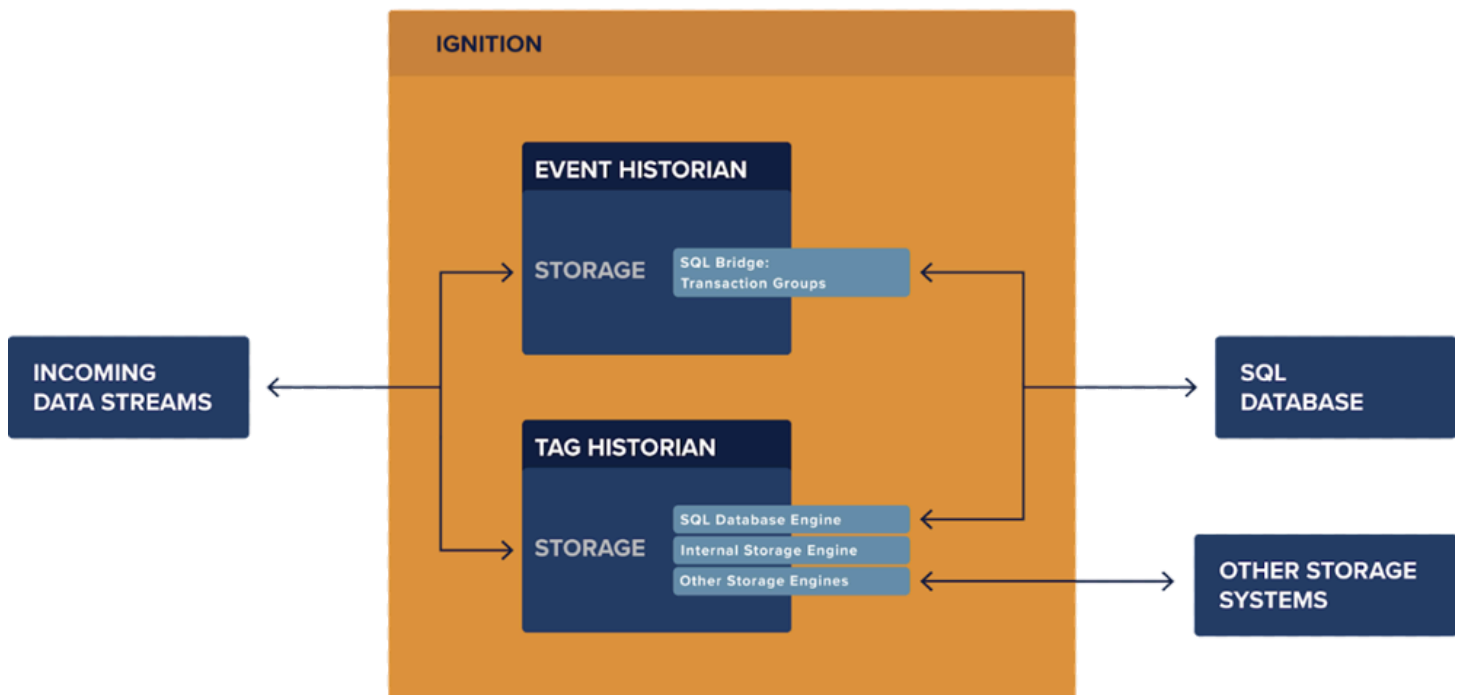
Reporting Module

Although often not included in a data historian, the Reporting Module is worth mentioning, since it allows for generation and automatic PDF reports to send directly from within Ignition.

Ignition Platform

Although this isn't a module, the platform itself is worth noting as it provides a number of features. In addition to providing the required subsystems, like the tag system, the Ignition Platform also provides scripting support to react and respond to tag changes in real time and includes an expression language that allows for tags based on equations or calculations.

Data Storage

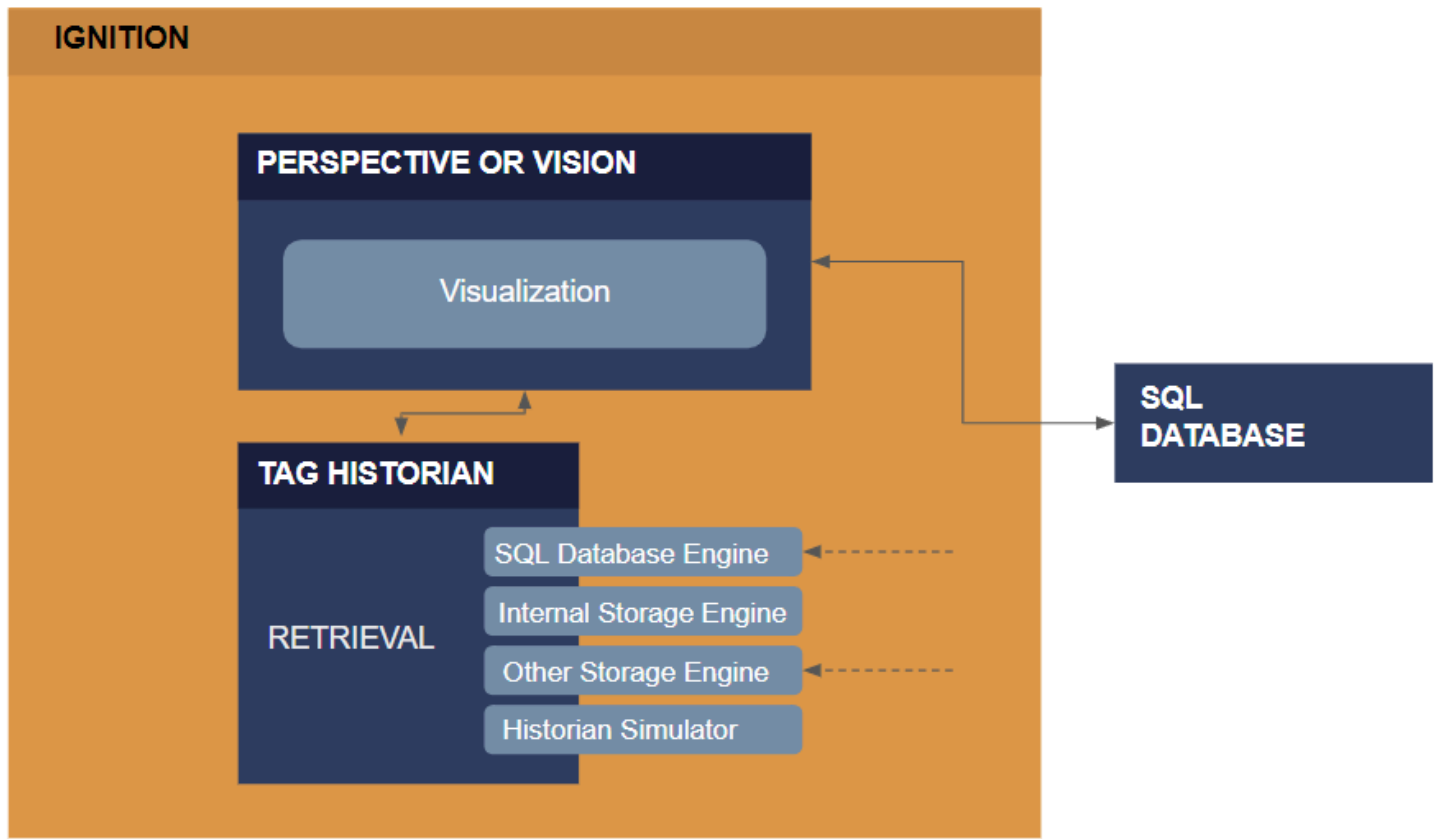


Data storage of event-based data through SQL Bridge is stored in SQL databases. SQL Bridge allows users to define table names and columns, and choose what to store depending on identified triggers. Historical Transaction Groups are a good choice for most event-based logging.

Data storage of tag-based data through Tag Historian can go to several locations. Most users target a SQL database, but other possibilities include small internal storage systems and third-party storage engines.

Data Retrieval

Using Ignition's Visualization Tools



When using Perspective or Vision, Ignition provides a nice set of built-in tools for data retrieval. For Tag History, this includes Tag History bindings, a data source type for the Reporting Module, automatic integration with some charting components, and drag-and-drop configuration within the design tools. For event data that's stored to a SQL database, Ignition has Named Queries that allow for retrieval including support for conditions, date and quality limitations, and any other options SQL syntax allows.

Using External Tools

For event data, external tools can query the database directly, which makes everything accessible to parties that have appropriate security permissions.

For tag historian data, it is recommended that external tools query data from Ignition's APIs. This is useful as any data retrieval will go through Ignition's logic that provides interpolation, aggregation, partition spanning, and any other logic needed. The API function for retrieving this data is `system.tag.queryTagHistory()`, and setting up a REST endpoint for that function is simple using the WebDev Module.

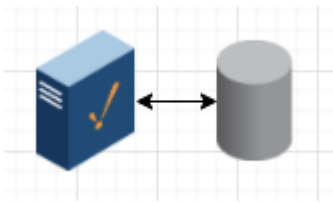
It should also be noted that Ignition's automatically-created database tables are fully and publicly documented for users who prefer to directly query the tables instead of use the recommended `queryTagHistory()` function.

Architectures

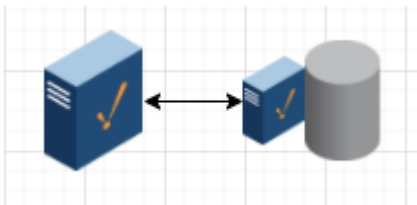
The following architectures are based on Ignition running with the Tag Historian Module targeting a SQL database. Architectures using different tag history storage could look significantly different. For example, the open source, free Azure ADX connector written by Microsoft has very high throughputs and stores to the cloud, so no SQL database is used.

If using a SQL database for storage, these architectures can provide a sense of possible configurations. See the Benchmarking section to get a sense of the possible throughputs for different database technologies.

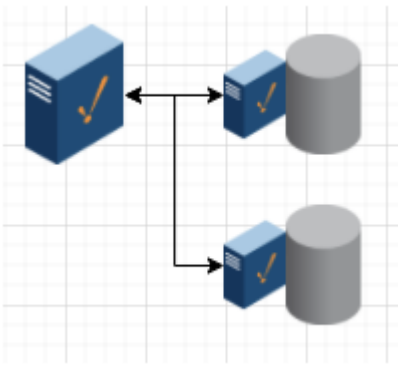
- **Standard Architecture:** The Standard architecture is as straightforward as Ignition running on one server and the SQL database running on the second server.



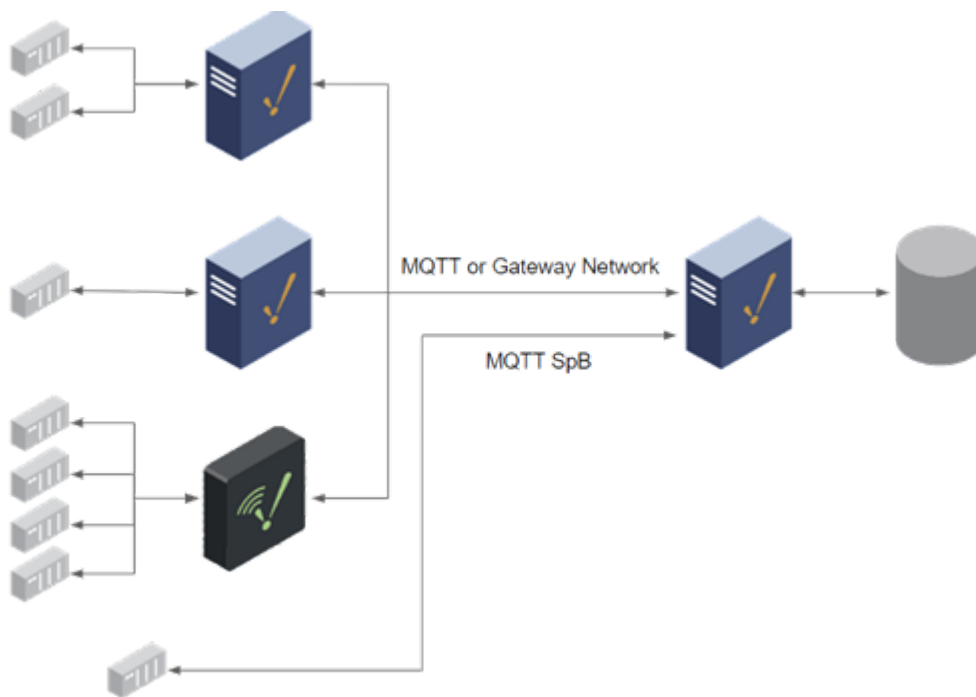
- **High Resilience Standard Architecture:** This architecture configuration adds a change to the database server by installing Ignition with only the Tag Historian Module alongside the SQL database. This offers a few advantages like high speed Gateway Network communication between servers and the ability to weather communication difficulties, latency, and packet loss well.



- **High Performance Multi-Database Architecture:** This architecture includes 3 or more servers, where the added servers replicate the database server as described in the High Resilience Standard Architecture. In addition to the communication advantages, this configuration also allows scaling past single database throughput limit.



- IIOT Architecture:** This architecture is geared towards collecting data from edge devices and centrally storing that data for further comparison and analysis. In this configuration, PLCs or other devices communicate with data collectors in the field. These data collectors could be Ignition, Ignition Edge, or systems that speak MQTT natively. Additionally, some devices speak MQTT natively with data buffering and store and forward technologies available. Since these devices use the light-weight, efficient, and secure MQTT protocol, they can often communicate directly to the central Ignition platform. Depending on the industry, collectors are optional. They do however make sense from a bandwidth, network segmentation, and security standpoint.



Benchmarking

Event-based data and tag-based data both need to have appropriate hardware and software to support the needs of the design. However, an appropriately designed system generally has thousands of times more tag-based data than event data. As such, the benchmarks here are focused on tag historian data and throughputs.

Understanding Rates and Change Percents

Ignition’s tag historian performance is best rated in tag changes per second. At a 10sec rate, with 10% of tags changing, 100 tags produce 1 tag change per second.

For example, the following table outlines the storage and change rates with the number of tags for a database that can handle 10,000 tags.

Example Database Tags

Storage Rate	10sec	60sec	1sec	1sec
Change Rate	10%	10%	10%	100%
Number of Tags	1 Million	6 Million	100 Thousand	10 Thousand

Number of Tags

The Example Database Tags table demonstrates how the amount of tags a single database can handle heavily relies on the storage rate and the change rate of the tags. Ignition systems are configured by default to use 10sec storage rates for tags. However, some users prefer 1sec rates or faster for some or all tags. Increasing rate speed is supported and encouraged as needed. Keep in mind, faster rates will equate to more changes per second going to the database.

SSD vs HDD

Inductive Automation highly recommends using an SSD both for the Ignition installation and for the database storage if medium-low, medium, or high throughputs are expected. If only low history throughputs are expected, an HDD may suffice. If choosing an HDD, testing the database’s throughput is recommended to ensure an HDD will handle the required storage speeds.

Internal Historian

The internal historian is a disk-based historian, available directly inside Ignition. This can be a great way to store a small amount of data at remote locations or locations where a SQL database isn’t available.

The data is set to be limited by default to one week. Although that limit can be expanded, Inductive Automation doesn’t recommend storing years of data in the internal historian, as the internal historian is only intended for small amounts of local history. If you calculate out your needed timeframe to log, your rates, and your change percent, we recommend keeping the total records around 10 million

rows. Since the internal historian doesn't have some of the advanced features of the SQL Database storage like automatic partitioning, anything planned for storing over 10 million records should use SQL Database storage or another storage engine.

Tag Changes Per Second

In order to provide benchmarks that are meaningful, these benchmarks are rated in Tag Changes Per Second. Please estimate your Storage Rate and Change Rate, and multiply them together to calculate your tag changes per second.

Tag Changes Per Second

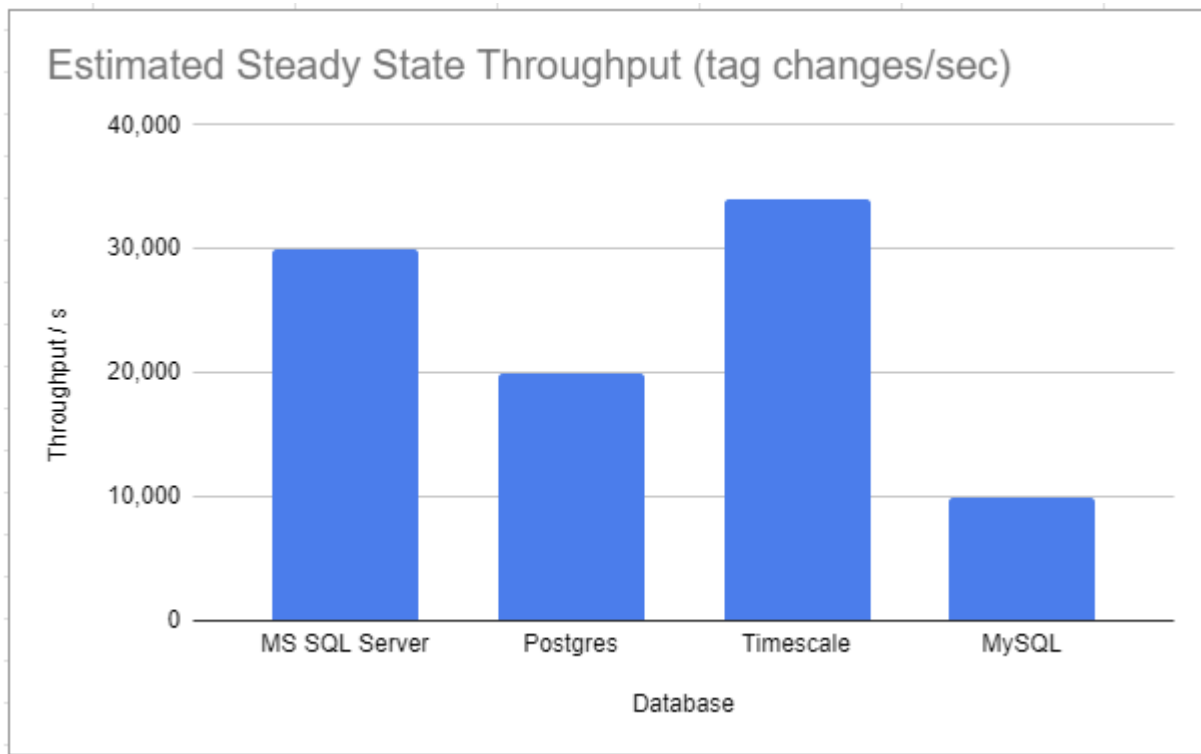
$$\text{Tag Count} * \text{Change Rate Percent} / \text{Storage Rate (in seconds)}$$

For example, 100,000 Tags * 20% changing / 5 sec storage rate = 100,000 * .2 / 5 = 4,000 tag changes per second.

SQL Database Benchmark Estimates

Category	Data
Platform	Windows
CPU	Intel Core i7-4790K (4/8 core, 4GHz)
RAM	16GB
Drive	Samsung SSD 860 EVO 2TB

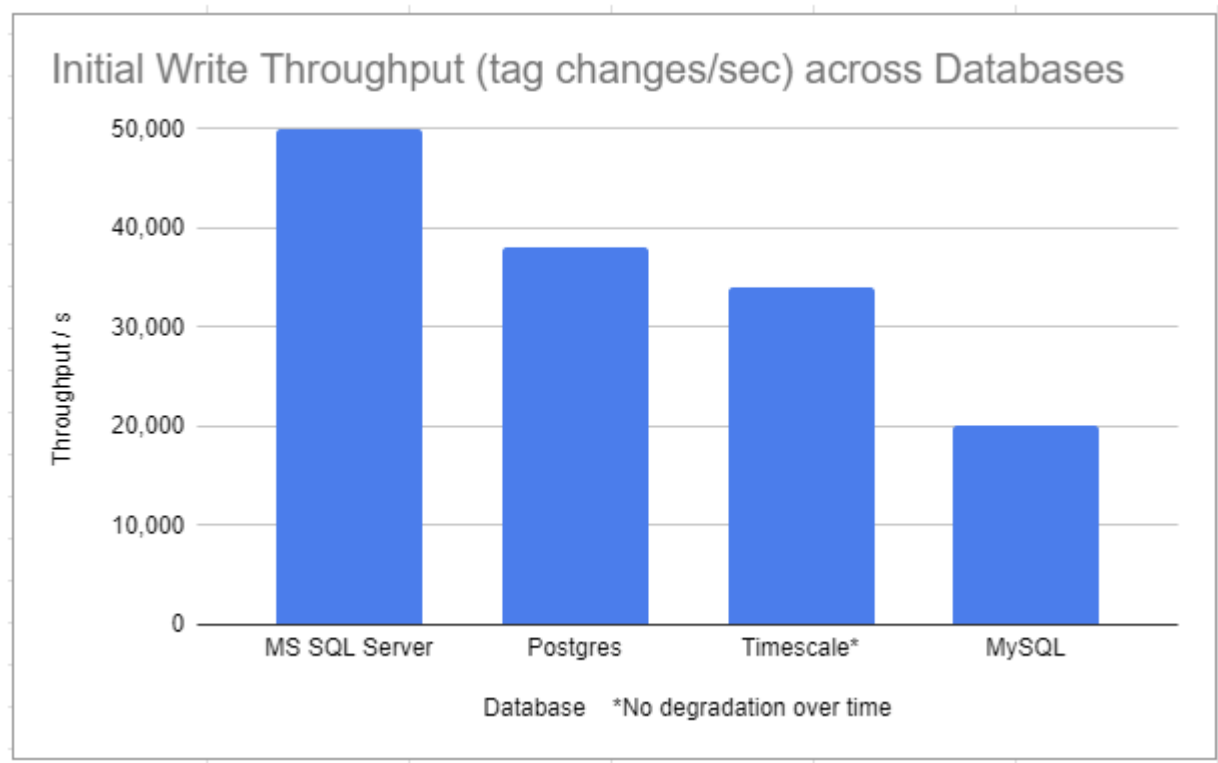
Estimated Steady State Throughput (tag changes/sec)



CAUTION

- These benchmarks are estimates only. Hardware differences and many other factors can have a significant impact on throughputs.
- These estimates are based on steady state throughputs. Having reasonable Ignition Tag Historian partition sizes configured will help to keep performance from degrading in most databases. Timescale is the exception on this list, as it doesn't experience degradation over time.
- Oracle isn't included on this list as Inductive Automation does not have an Oracle enterprise license. The lightweight Oracle Express was tested and had similar performance to MySQL, but it's expected a full version of Oracle would have significantly better performance.

Initial Write Throughput (tag changes/sec) Across Databases



These Initial Write benchmarks are included just for reference. Most databases have high performance on initial throughput and that speed then drops until they reach a steady state. If you’re running your own tests and the initial results are showing higher throughput, be sure to let the system run for a few hours or a few days to let the rates even out.

Storage Space Requirements

Uncompressed, unoptimized storage requirements in SQL databases take around 100 bytes per tag change, regardless of the database chosen.

Based on that number, it’s fairly easy to calculate storage requirements.

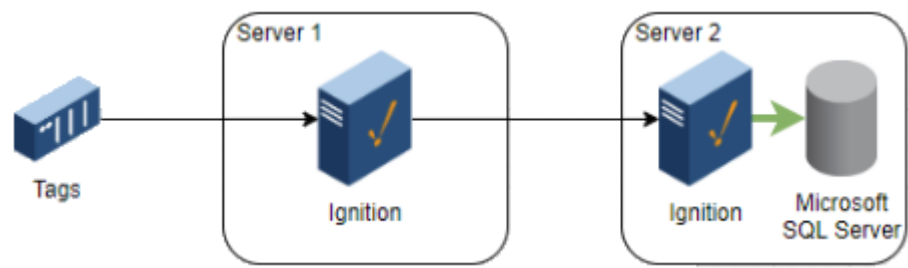
Storage Space Requirements

$$\text{Tag changes per second} * 100 * \text{seconds in time period}$$

For example, if you have 1,000 tags with 10% changing and at a 10s rate, you’re looking at 10 tag changes per second. Multiply that by 100 bytes and 86,400 seconds in a day, and you get around 86.4 MB per day.

Note that many databases have compression options, which can reduce the overall storage required by 30-50% or more, generally at the cost of CPU/throughput. Timescale, specifically, has an option

High Resilience Standard Architecture



	Latency (Server 1 to Server 2)								
Dropped Packets	0ms	1ms	5ms	10ms	20ms	50ms	100ms	150ms	200ms
0%	30	30	30	30	30	30	30	26	20
2%	30	30	30	30	30	30	19	14	12
5%	30	30	30	30	30	19	11	9	7
10%	30	22	21	20	17	10	9	7	5
15%	20	14	13	12	11	7	6	5	3
20%	5	-	-	-	6	-	-	-	-

Ignition is much more resilient in Scenario 2 using the High Resilience Standard Architecture, where a performance degradation can't be seen until there are a significant percent of dropped packets or over 100ms latency.

Optimization

SQL Bridge

These optimizations may apply to all systems.

- Configure data pruning on tables that don't need to store history forever.
- Ensure triggers are used to store events properly.
- Don't store data that's not needed. If time series storage is needed, consider using tag historian instead, and logging events through Transaction Groups.

- When writing queries against the data stored by SQL Bridge, look at the WHERE clause of the queries and create appropriate indexes in the database tables.
- Monitor the system for performance, to ensure you didn't miss adding an index or that database table size isn't excessive for the project and the hardware it's running on.

Tag Historian Using SQL Databases

These optimizations may apply to all systems.

- Set reasonable deadbands for tags. Deadbands make sure that junk data isn't logged. If a sensor has an accuracy of 0.2, there's no reason to ever log a change of 0.02, since that change is just noise. Setting reasonable deadbands can save a significant amount of storage space and throughput from logging noise.
- Set time deadbands ("Min Time Between Samples") as it makes sense, so tags don't log faster than specific rates.
- Set tag rates appropriately. Only set tags to log at 1sec rates if needed. Default to slower rates unless requirements dictate otherwise.
- If it makes sense for your system, we recommend turning off Stale Data Detection. If the system is running slower than it feels it should be, check the `sqlth_sce` table. If there are more than a few hundred rows, you may have inadvertently overloaded the system. Turning off the Stale Data Detection is normally a safe thing to do (under the advanced properties in the database connection), but read the user manual to make sure you understand the setting first. Turning that setting off will keep the `sqlth_sce` table from growing if the system is experiencing slowness.
- If using MySQL, add an extra connection parameter to the database connection. This will increase throughput from around 1,000 tag changes per second to around 10,000-20,000. `rewriteBatchedStatements=true`.
- Ensure no other applications or services are running on the systems where Ignition and the database(s) are hosted.
- Tune partitions to keep from exceeding 100m to 1b rows per partition. Keeping partition sizes reasonable makes queries over short time frames faster, and helps ensure dashboard charts and graphs render relatively quickly.
- Set up pre-processed partitions if querying over longer periods is expected. Note that doing so will require some additional storage and processing, which will impact the overall achievable storage throughput marginally.

High Throughputs

These optimizations apply if you have a system spec'd or in production where you are expecting high throughput in terms of tag changes per second.

- Option 1: Reduce throughput following the guidance above.
- Option 2: Add another database/schema to the SQL database DBMS on the same storage. Multiple databases can help split the load, and can increase throughput. Set some tags to go to one database, and other tags to go to the other database.
- Option 3: If throughputs still aren't achievable, putting multiple databases on the same DBMS, but pointed at different storage can also increase throughputs. Some databases are heavily limited by drive IO, and splitting databases to different storage can have a significant impact.
- Option 4: Pointing Ignition at multiple database servers will ensure separate throughput to each server, which effectively doubles the possible throughput that comes from a single database server.
- Option 5: Consider a distributed architecture. If there are multiple sites and a central system, consider keeping site data at each site, and sending summary data to central. Don't worry; if you're using the Gateway Network, you can always access site tag historian data from the central system, treating all the connected Ignition gateways as a large distributed system. Central reports, dashboards, and other visuals can stream data from the sites to display it centrally, as long as the site is online.
- Option 6: If very high throughputs are required, in the 500,000 to 1,000,000 tag changes per second range for example, this may exceed what is practical with a set of SQL databases. Other storage engines are available for Tag Historian from the Module Showcase and other areas that have extremely high throughputs. These options may be worth considering when very high throughputs are needed.

Reducing storage space requirements

- Database Compression (30-50%)
 - Many databases have options to compress data for tables.
- Full Drive Compression
 - Drive compression for storage drives can also provide benefits.
 - Not normally used together with Database Compression.
- If using Timescale
 - Timescale's Chunking & Compression can provide even more significant savings. Compression happens after a configurable time period. Drive space savings happen after that period has elapsed and trigger the compression. Savings are promoted as up to 90% or more. Inductive Automation has not run benchmarks, so the exact savings against the Tag Historian table data is unknown at this time.
- Using other storage engines
 - Some storage engines have significant space savings.
 - Examples:
 - Influx from Module Showcase has smaller storage requirements per record.

- Azure ADX's has very high throughput support and is supposed to have efficient storage. Microsoft has released a free, open source module as a connector, which can be found on GitHub. The Azure ADX service is a paid service from Microsoft.
- Although Inductive Automation may provide guidance to module authors from time to time, Inductive Automation is not able to provide support for Module Showcase modules. We encourage customers exploring these to contact the module authors and explore any module documentation before making a decision to use one of these modules.

Security

Security Hardening Guide

Inductive Automation has a strong focus on security and provides guidance on securing individual Ignition Gateways. Inductive Automation recommends referring to the latest [Security Hardening Guidelines](#) for the latest in securing an individual Ignition Gateway.

Encryption in-flight

Encryption in-flight can be an important consideration for any historical data. Ignition can run over secured networks, and all data transfer can run over VPN or other encrypted tunnels that support normal IP traffic. Additionally, Ignition client-to-server communication can be encrypted, Gateway-to-Gateway communication can be encrypted, and Gateway-to-Database traffic can be encrypted for all major databases. Some devices themselves don't support encryption or other security measures, so it's important to consider how to secure that device communication. Sometimes the solution is to put a small data collection device next to the insecure hardware, running Ignition Edge or another piece of software, to keep unencrypted communication off of any networks. Other solutions include VLAN encryption and securing networks for unencrypted traffic using firewall rules, IDSs, and other security measures. Inductive Automation recommends working with security experts if any questions arise on device communication security.

Encryption at-rest

Encryption at-rest can be accomplished by encrypting drives where data will be stored or encrypting database tables if using a SQL database for data storage. These encryption tools will be part of the database management tools. Encryption at-rest generally doesn't require the drive that Ignition is running on to be encrypted, since Ignition does not act as the main storage medium. Depending on the traffic, performance, and settings in Ignition, there's a chance that cached data in the Store & Forward system could be temporarily written to disk from Ignition. If this data is required to be

encrypted at-rest, even while temporarily buffered to disk, then in that case it would be appropriate to encrypt the drive Ignition runs on as well.

Client security

Client security is normally accomplished using TLS 1.2/1.3 encryption, which provides the same level of encryption as a typical banking website. Standard PKI certificates are used, and the steps for configuration are covered in the Ignition documentation. Refer to the [Security Hardening Guide](#) for guidance on enabling TLS in Ignition.