Ignition by inductive automation

Ignition Historian

inductive automation

Ignition Historian

Overview, Configuration, and Implementation Information

Last updated for Ignition 8.1.

Table of Contents

Overview

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, 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

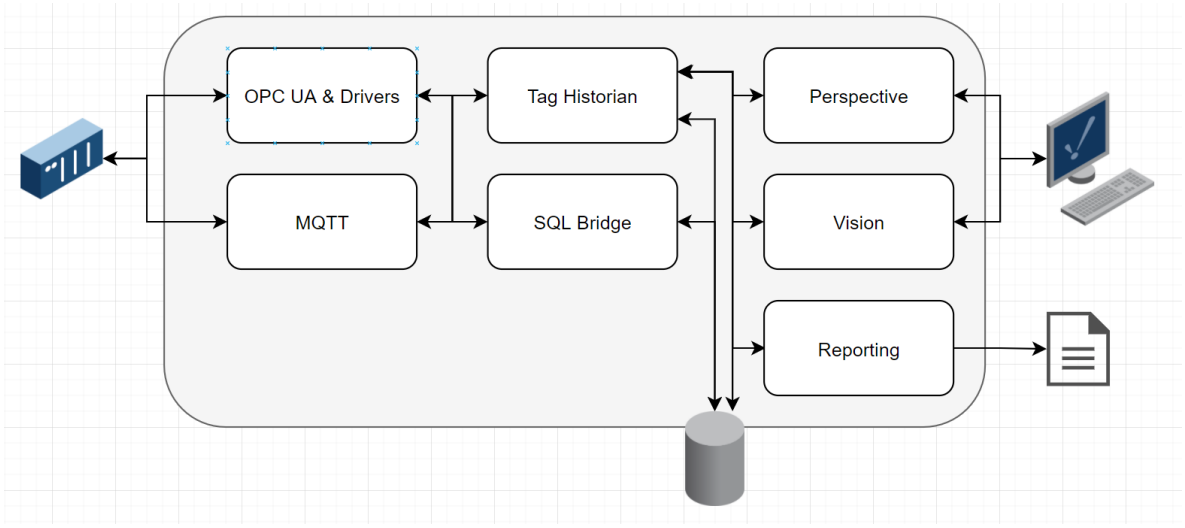


Diagram 1: One possible configuration of modules when Ignition is used as a data historian.

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. Many companies use 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 below.

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 through. It also includes event storage through DRecords, which place data in database tables which 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.

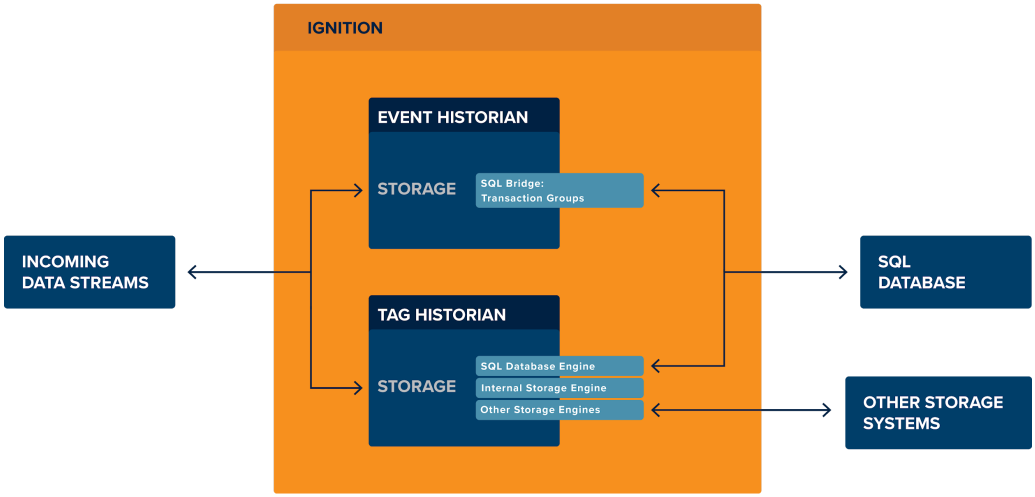
Reporting Module

Although often not included in a data historian, the Reporting Module is worth mentioning, since it allows for generation and automatic sends of PDF reports 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 that are based on equations or calculations.

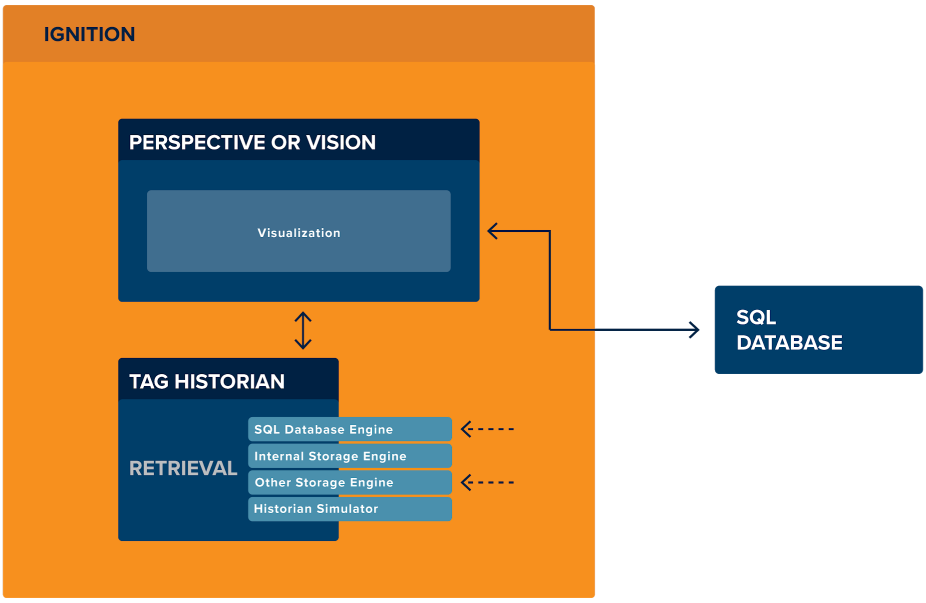
Data Storage



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 and upon what 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. Other possibilities include a small internal storage system and third-party storage engines, many of which can be found on the Module Showcase.

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 some drag-and-drop configuration in 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 anything else that SQL syntax allows.

For use with external tools

For event data, external tools can query the database directly, making 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 so any data retrieval goes through Ignition's logic that provides interpolation, aggregation, partition spanning, and any other logic needed for tag data retrieval. 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.

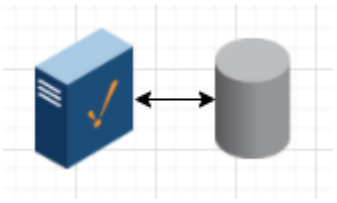
It should also be noted that Ignition's automatically-created database tables are fully and publicly documented for those users who do wish to directly query the tables instead of using the recommended queryTagHistory() function.

Architectures

These architecture diagrams are based on Ignition running with the Tag Historian Module targeting a SQL database. Note that 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.)

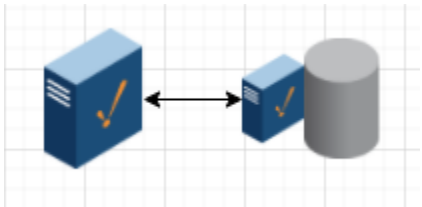
If using a SQL database for storage, these architectures can provide a sense of possible configurations. See the Benchmarks section below to get a sense of the possible throughputs for different database technologies.

Low Resilience Simple Architecture



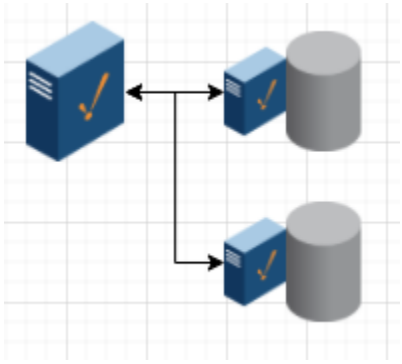
Server 1: Ignition
Server 2: SQL database

High Performance Simple Architecture



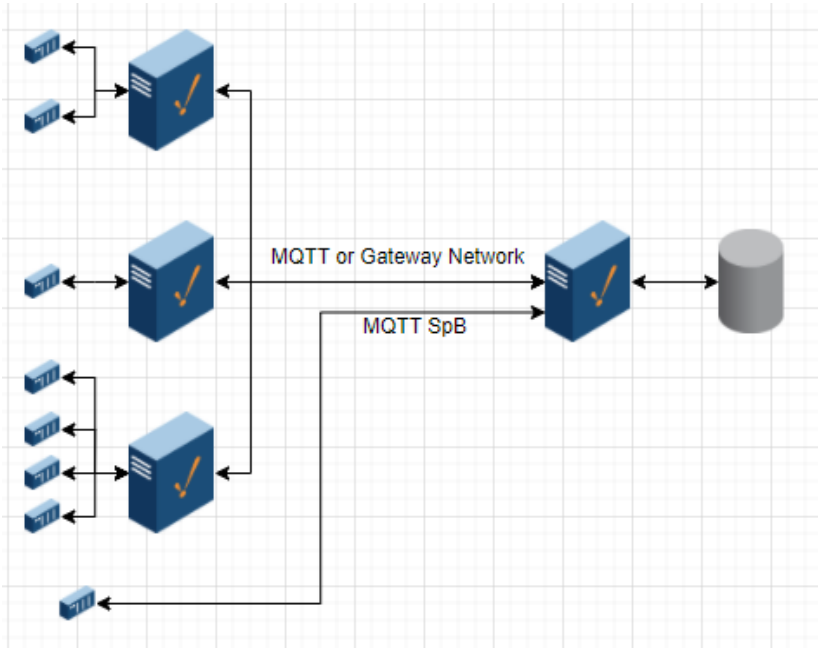
Server 1: Ignition
Server 2: Ignition with only Tag Historian Module, and SQL database installed on same server
Advantage: High speed Gateway Network communication between servers. Also weathers communication difficulties, latency, and packet loss well.

High Performance Multi-Database Architecture



Server 1: Ignition
Servers 2 & 3: Ignition with only Tag Historian Module, and SQL database installed on same server
Advantage: Allows scaling past single database throughput limits.

Data Collector Architecture



PLCs or other devices communicate with data collectors in the field. Those data collectors could be Ignition, Ignition Edge, or systems that speak MQTT Sparkplug B natively. Additionally, some devices speak MQTT Sparkplug B natively with history transfer and store & forward supported. Those devices can often communicate directly without a separate collector.

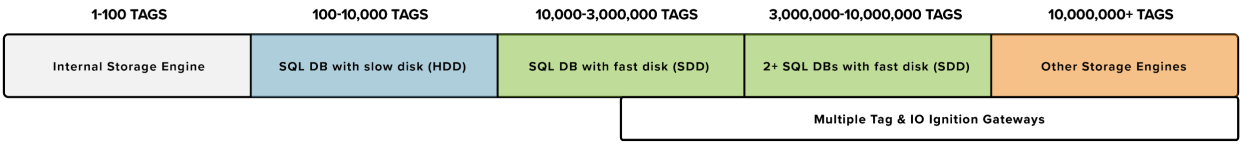
Collectors are optional. Depending on the industry, they can make sense from a bandwidth, network segmentation, and security standpoint.

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 the Benchmarks

Benchmarks are based on throughputs. For example:



At a 10s rate, with 10% of tags changing

Understanding Rates and Change Percents

Note the rate and the tag change percent listed above. Ignition's tag historian performance is best rated in tag changes per second.

At a 10s rate, with 10% of tags changing, 100 tags produce *1 tag change per second*.

Let's say a database can handle 10,000 Tag Changes Per Second. This table outlines how many tags that handles:

Example Database Tags

Storage Rate	10s	60s	1s	1s
Change Rate	10%	10%	10%	100%
Number of Tags	1 Million	6 Million	100 Thousand	10 Thousand

How many tags can a single database handle?

As you can see from the table above, that depends heavily on the Storage Rate and the Change Rate of the tags. Ignition systems are configured by default to use 10s storage rates for tags. However, some users want 1s rates or faster for some or all tags. Going to faster rates is supported and encouraged as needed. Keep in mind, however, that faster rates will equate to more changes per second going to the database.

SSD or 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 would be 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 1 week of data. 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. As 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
Tag Count * Change Rate Percent / Storage Rate (in seconds)

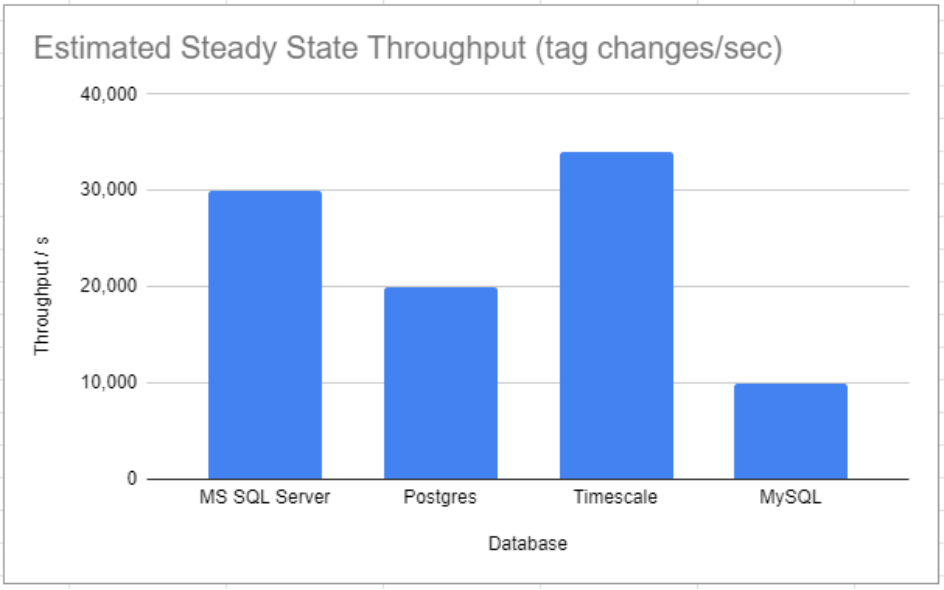
Example: 100,000 Tags * 20% changing / 5 s storage rate = 100,000 * .2 / 5 = 4,000

Inductive Automation has a storage calculator spreadsheet available that can be provided upon request to help with estimating.

SQL Database Benchmarks

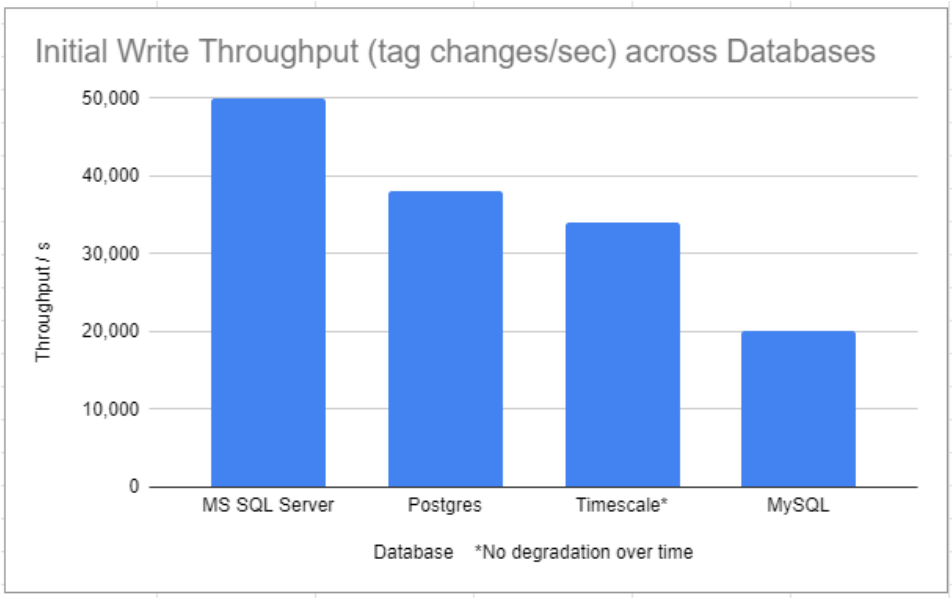
Test System

Platform	Windows
CPU	Intel Core i7-4790K (4/8 core, 4GHz)
RAM	16GB
Drive	Samsung SSD 860 EVO 2TB



Notes:

- 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.



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. The first image above captures that 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

Tag changes per second * 100 * seconds in time period

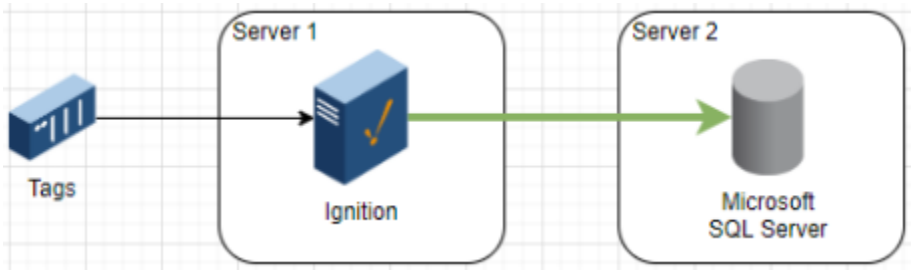
For example, if you have 1,000 tags with 10% changing and at a 10s rate, you're looking at 10 tag changes per second. Multiply that by 100 bytes and 86,400 seconds in a day, and you get around 86.4 MB per day.

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 that claims to reduce storage by up to 90% in certain circumstances. If storage space is a main constraining factor, it may be worth exploring these options.

Throughput vs Network Quality

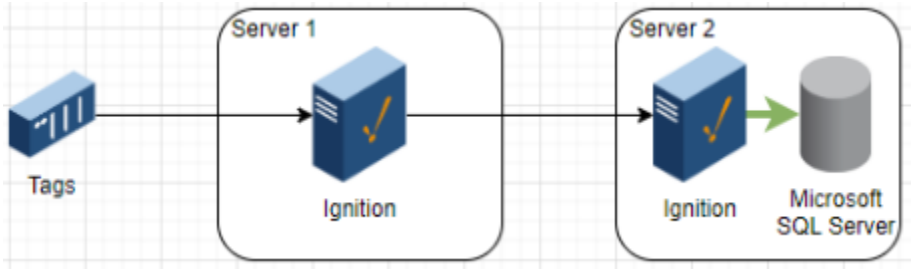
Network quality throughput testing was performed using Microsoft SQL Server. It is believed to be a reasonable expectation that other databases follow similar performance characteristics when it comes to network latency.

Scenario 1



		Latency (Server 1 to Server 2)								
		0ms	1ms	5ms	10ms	20ms	50ms	100ms	150ms	200ms
Dropped Packets (Server 1 to Server 2)	0%	30	19	17	15	12	9	6	4	3
	2%	22	4	3	3	2	2	-	-	-
	5%	10	1	1	1	1	-	-	-	-
	10%	1	-	-	-	-	-	-	-	-
	15%	-	-	-	-	-	-	-	-	-
	20%	-	-	-	-	-	-	-	-	-

Scenario 2



		Latency (Server 1 to Server 2)								
		0ms	1ms	5ms	10ms	20ms	50ms	100ms	150ms	200ms
Dropped Packets (Server 1 to Server 2)	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	-	-	-	-

All results are rated in thousands of tag changes per second.

As can be seen in the numbers, Ignition is much more resilient in Scenario 2, where a performance degradation can't even be seen until there are a significant percent of dropped packets or over 100ms latency. This testing was performed with large numbers of tag changes at a one-second rate.

Optimization

General Optimizations

These optimizations may apply to all systems.

SQL Bridge

- 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 just 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, and to ensure database table size isn't excessive for the project and the hardware it's running on.

Tag Historian using SQL databases

- 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 and is worthless. 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 1s 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 does not support or endorse 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, which can be found here: https://inductiveautomation.com/resources/article/ignition-security-hardening-guide

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 above for guidance on enabling TLS in Ignition.