Tag Historian and Data Log

The Tag Historian module performs automated data logging of selected tags to SQL databases, and others. You can also use the Datasets module to store this data in SQL, but the Tag Historian module has a simplified configuration, with the database tables automatically created and a built-in connection with the trend charts.

The system can also use the information from the historian for .NET scripting, get past values for tags or graphic displays or export the data.

You can select any SQL database, such as Microsoft SQL Server, Oracle, MySQL or any OLEDB or ODBC compliant database to store the data. By default, FactoryStudio uses an embedded SQL database (TatsoftDB) that has a maximum capacity of 10 GB.

When using OSIsoft(tm) PI System, there is no need to do any Historian configuration to access the data stored in the PI System.

On this page:

Selecting a Database

The database used to store the Tag Historian is defined in Edit > Datasets > DBs by the database connection object named TagHistorian.

By default, when a new project is created, the TagHistorian is defined to use the Tatsoft built-in embedded SQLite database.

> The SQLite database may be used for databases up to 10GB. If the amount of tags and save interval is expected to create more than 10GB, you should define another SQL database for the Tag Historian.

To define another database to store the Tag Historian, you just need to create a new database connection, and name it TagHistorian. The system does not allow duplicate names, so to create a DB connection named TagHistorian you must first rename or delete the existing row using that name.

The Tag Historian Tables

Configuration

By default, the tag historian database has one table configured. The table settings provide rules for saving each tag. You can set a trigger that determine when tags will be saved, a time deadband that defines the minimum interval between saves, and a lifetime value that determines how long saved tag values will be retained. The table configuration is independent of the database selected to store the information.
If you want some tags to have different settings, such as different time deadbands, you should configure additional tables with the settings you need and assign tags to the appropriate table. You can also edit the settings of the default table.
In general, you should not store more data than you need. Storing a large amount of data slows the recovery of that data. You should use triggers and deadbands that are as large as possible, to ensure you have the information you need, without overloading the system.

To configure a tag historian table:

- Go to **Edit > Tags > Historian**.
- Do one of the following:
- To edit an existing table, select it from the Historian Tables drop-down list and click **Config**.
- To create a new table, click **New**.

Enter or select information, as needed.

Option	Description
Database	Display-only name of the current tag historian database.
Table Name	Enter a name for the table in the database.
Auto Create	Select to have the system automatically create the table in the database.
Save on Change	Select to store data in the table (add a row) every time a tag associated with the table changes.
Trigger	Use to store data in the table every time a tag or tag property changes. When used with the Save on Change option, the system stores data in the table when either the tag value changes or the Trigger value changes.

Time Deadband (Log TimeSpan)	Enter the minimum logging interval, that is, how long the system must wait after storing the value of a tag before storing a new value. Use with the Save on Change option to avoid creating too many records in the database.
Life Time	Number of days to retain the historian data. After that time, the older rows are automatically deleted from the database. To never delete data, leave this field blank or enter 0 (zero).
Save Quality	Enable or disable the quality of the tag to be stored on the historian table
Normalized	Enable or disable to configure the schema to be used on the historian table
GetSamplesMethod	Use to configure a Script > Class method to customize the data retrieved from the historian table
Description	Brief description of the historian table

Using the OSIsoft™ PI System

The system can seamlessly use the OSIsoft PI System as the historian provider. In this scenario it is not necessary to do any Historian configuration at all. If the tag is mapped to a OSIsoft PI point, the system will automatically call the PI Server to get data when plotting trend charts or any script or display methods requiring historian information.

The system can work with both the PI System and the built-in historian at the same time. When historical information is requested for a point, either from trend charts or scripts, the system will first look for that information on the built-in historian module. If it is not available, it will try to find the data on the PI server. For information on connection to PI Systems, see "Import Wizards".

The Tag Historian Tables Schemas

There are two different schemas for tag historian tables, based on the configuration of the tables: the default schema and the normalized schema. Each schema is described below.

The Default Schema

The default historian tables contain the following columns:

- ID - PK, BigInt (8 Bytes) - The primary key of the table used as reference by the system.
- UTCTimeStamp_Ticks - BigInt (8 Bytes) - Date and time in Universal Time for that row in 64-bit .NET ticks. The value of this property is the number of 100-nanosecond (1/10th of a microsecond) intervals that have elapsed since 12:00 A.M., January 1, 0001. This is a new date/time standard used by the Microsoft .NET framework.
- LogType - TinyInt (1 byte) - Auxiliary column to show when the row was inserted: 0=on startup, 1=normal logging, 2=on shutdown.
- NotSync - Int (4 Bytes) - Auxiliary column to show if the data has been synchronized or not when the Redundancy option is enabled. See "Deploying Redundant Systems".
- TagName - Float (8 Bytes) - Column automatically created using the name of each tag as the column title. It stores the data value using double precision.
- _TagName_Q - Float (8 Bytes) - Column automatically created for the quality of the data for each tag, using the OPC quality specification.

Typically you can associate up to 200 tags with each historian table, but that number is dependent on how many columns your target Database allows. The tags should be defined in the same table when they have similar storing rates and process dynamic, since when you need to save one tag in the table, you need to save the entire row.

The Normalized Schema

The Normalized tables have the following schema:

Table: TagsDictionary

- ID - PK, BigInt (8 Bytes) - The primary key of the table used as reference by the system.
- TagName - NVarchar - The name of all the tags configured to "normalized" databases on Historian
- NotSync - Int (4 Bytes) - Not used for this release. Created expecting future changes and new features.

The system will automatically create four more tables as follows:

- TableName_BIT
- TableName_FLOAT
- TableName_NTEXT
- TableName_REAL

The schema for these table is:

- ID - PK, BigInt (8 Bytes) - The primary key of the table used as reference by the system.
- UTCTimeStamp_Ticks - BigInt (8 Bytes) - Date and time in Universal Time for each row in 64-bit .NET ticks. The value of this property is the number of 100-nanosecond (1/10th of a microsecond) intervals that have elapsed since 12:00 A.M., January 1, 0001. This is a new date/time standard used by the Microsoft .NET framework.
- ObjIndex - FK, Integer (4 Bytes) - Reference to the column ID on table TagsDictionary
- ObjValue - can be Bit, Float, NText or Real, depending on which table it is - Represents the value of the tag on the specified timestamp
- ObjQuality - TinyInt (1 Byte) - Represents the quality of the tag on the specified time, using the OPC quality specification.
- NotSync - Int (4 Bytes) - Not used for this release. Created expecting future changes and new features.

It's important to remember that the normalized database cannot be synchronized through the Redundancy option.

Adding Tags for Data Logging

After you configure your tables for the tag historian, you can configure the tags that will be logged.

To configure the tags for the tag historian:

- Go to **Edit > Tags > Historian**.
- You can copy and paste tags from the **Objects** tab.
- Enter or select information, as needed.

Column	Description
TagName	Enter a tag name or click ... to select a tag.
DeadBand	When using the Save on Change option in the historian table, the DeadBand is how much the value must change (in Units) for the system to store the value in the historian.
Deviation	When using the Time DeadBand option in the historian table, the Deviation is how much the value must change (in Units) for the system to store the value in the historian. This value overrides the Time Deadband logging interval.
RateOfChange	When using the Time DeadBand option in the historian table, the RateOfChange is how much the value must change (in Units) by second, for the system to store the value in the historian. This value overrides the Time Deadband logging interval
HistorianTable	Select the table that has the settings you want to use for when to save and how long to retain this tag's value.
[Other columns]	For definitions of other columns that are available in many tables, see "Common Column Definitions".

- Continue adding as many tags as you need.

Other settings:

Use Binary Cache: Select this option to store the tag historian data in binary blob columns instead of directly accessible values.

> ⚠ Changing this setting changes the storage format for future saves. Data in the previous format is not automatically converted.

Visualizing Trend Charts

In order to visualize trend charts with historical information, you can use the built-in trend object on the displays or reports. See "Configuring the Trend Window" for more information.

Customizing Getting Samples

The trend chart object calls the Historian server to get the data to plot the charts. In some scenarios, you may want to override that configuration and define a .NET code to provide the values. This is used, for instance, to plot recipe calculated data, future data, data from other SQL tables or any custom scenario.

The custom GetSamples method should be defined at any Script Class, and on Edit-Tags-Historian that method must be defined at the column GetSamplesMethod.

> ⓘ The prototype of the method is:
> DataTable GetSamples(string[] tagNames, object startRange, object endRange)

When creating tables used on time charts the StartRange and EndRange are of type DateTimeOffset. When getting data to X-Y charts, the Range arguments are double variables.

The Returned DataTable for time Charts shall have the columns:

- DateTime: Date time of the sample

- TagName: The name of the tag is used as FieldName to the column with Double values of the tag
- _TagName_Q: Optional column with the quality of the data

The Returned DataTable for XY Charts shall have the columns:

- X: The name X is used to the column with Double values
- TagName: The name of the tag is used as FieldName to the column with Double values of the tag
- _TagName_Q: Optional column with the quality of the data

The Historian Namespace

The **Historian** namespace has the properties and current status of the Historian server. The **Historian.Table** object has the list of Historian tables defined and their properties.

The following tag property is enabled if there is data logging for each tag.

Tag.tagname.Historian