Historian Tables Reference

Organize historian data in tables.

Historian Tables (Reference) define the storage configuration and archiving behavior for groups of historian tags within the FrameworX platform. Historian Tables are logical containers that:

- Group related historian tags for organized storage
- Define when and how data is archived
- Specify storage locations and retention policies
- Configure triggers and collection methods
- Optimize database schemas for performance

Each table represents a collection strategy tailored to specific data characteristics and storage requirements.

HistorianTable Configuration Properties
Save Quality
Creating Historian Tables
Archiving Methods
TimeDeadband Configuration
Schema Selection
Data Retention
SaveQuality Option
Trigger Configuration
Performance Optimization
Common Configurations
Exporting a Table to CSV
Troubleshooting
Best Practices Checklist

HistorianTable Configuration Properties

When configuring a HistorianTable, you need to specify some properties defining when and how data will be archived. The following table describes each of the properties available.

Property	Required	Description
StorageLocation	Yes	Defines the Storage Location used. By default, the TagHistorian based on SQLite is used. For more information, access the Historian Storage Locations Reference.
Name	Yes	Name of the HistorianTable object. For other databases, this name is used as a logical grouping name.
AutoCreate	No	If you select this option, the Historian module will verify if this table has already been created in your SQL database. If the table does not exist, it will create a new one.
SaveOnChange	No	If you check this option, an action event to archive Tag data is generated whenever the Tags value changes.
Trigger	No	The Trigger option enables you to choose an event that, when it happens, will trigger the archiving of all Tags associated with the current HistorianTable in the database. You can use any tag, object, or property from the object's namespaces as a trigger. Click the **Elllipses button** (...) to access all options available.
TimeDeadband	No	Specify the minimum logging interval using HH:MM:SS:MS (Hours, minutes, seconds, milliseconds) format. This parameter determines the duration the system must wait after storing one tag value before storing a new one. It prevents excessive record creation in the database, especially if you're using the SaveOnChange option. If set as zero, this setting is inactive. As an example, consider you set a five-second deadband. After a Tag is archived, the counter begins. If the counter doesn't reach five seconds, no Tag information will be saved, independent of how big its value changes. The table below exemplifies the example. **Tag's value** / **Current Time** / **Will be saved on Database?** 5 / 12:05:05:00 / Yes 20 / 12:05:07:00 / No 25 / 12:05:10:00 / Yes
LifeTime	No	Set the duration in days for retaining records on the Storage Location. The system automatically deletes data older than the specified lifetime. If you set the value to 0 or leave it blank, the system will not perform automatic data deletion, ensuring the data remains untouched.
SaveQuality	No	When checked, a column in your SQL DB stores the Quality property of the Tag along with the value. The quality level defines how reliable the Tag is, and it follows the OPC standard specification. By checking this option, only points with real data will be displayed when you access a graph. Check more about this option in the section on this page.

Normalized	No	When checked, the system uses the Normalize table schema for SQL databases or the Standard schema otherwise. For more information on SQL database schemes, refer to the Archiving Process.
ValueColumnsType	No	Select the type of value to be stored in the value columns, which defines the Tags' value type in the SQL database. You can change this property only when creating the HistorianTable.
Description	No	Description for this HistorianTable object.

Save Quality

In certain instances, the graph may exhibit lines and values even in timestamps where there is no data present. For instance, when you access a graph that you know lacks data and you see a continuous line connecting the endpoints of existing data points, despite the absence of data in between, you can infer that you are leading with unreal information. If you click on such areas, the value shown can lead you to misleading cursor values, as no data exists for those specific moments.

To address this issue, you can remove the line connecting the points of the graph where there is no data. This can be achieved by enabling the **SaveQuality** property in your HistorianTable, as depicted in the image below. If the **SaveQuality** option is not available in any column, you can enable it by right-clicking on the top of any column name and select **SaveQuality**.

Creating Historian Tables

1. Navigate to **Historian HistorianTables**
2. Click **Plus icon**
3. Configure properties:
    - Set **Name**
    - Choose **StorageLocation**
    - Select trigger method
    - Configure deadbands
4. Click **OK**

Configuration Dependencies: Some options are mutually exclusive. For example, Normalized tables cannot use Triggers and must use SaveOnChange.

Archiving Methods

SaveOnChange

Archives data when tag values change:

- Individual tag monitoring
- Respects deadband settings
- Efficient for sparse data
- Works with normalized schemas

Trigger-Based

Archives all table tags on trigger event:

- Synchronized timestamps
- Batch storage operations
- Supports time-based or event-based
- Standard schema only

Combined Approach

Using both SaveOnChange and Trigger:

- Primary: SaveOnChange for value changes
- Backup: Trigger ensures periodic storage
- Prevents data loss during steady-state

TimeDeadband Configuration

Prevents excessive database writes:

Format: HH:MM:SS:MS

Example Operation:

Tag Value	Time	TimeDeadband	Archived?
5	12:05:05:00	00:00:05:000	Yes
20	12:05:07:00	00:00:05:000	No (2s < 5s)
25	12:05:10:00	00:00:05:000	Yes

Interaction with Tag Settings:

- Tag Deviation can override deadband
- RateOfChange can force immediate storage
- DeadBand filters small changes

Schema Selection

Standard Schema

Best for:

- Synchronized data collection
- Trigger-based archiving
- Fixed tag groups
- Simple queries

Characteristics:

- One row per trigger/interval
- All tags share timestamp
- Fixed columns per tag
- Maximum ~200 tags per table

Normalized Schema

Best for:

- Asynchronous data
- SaveOnChange archiving
- Sparse data patterns
- Large tag counts

Characteristics:

- One row per tag change
- Individual timestamps
- Dynamic tag addition
- Unlimited tags per table

Data Retention

LifeTime Setting

Automatic data management:

- **0**: No automatic deletion
- **> 0**: Delete data older than N days
- Runs daily maintenance job
- Frees storage space

Retention Strategies

Data Type	Recommended LifeTime
High-frequency process	30-90 days
Production records	365 days
Quality/compliance	2555 days (7 years)

Temporary/debug	7 days

SaveQuality Option

Stores OPC quality information:

Benefits

- Identifies bad/uncertain data
- Prevents interpolation over gaps
- Improves graph accuracy
- Supports data validation

Quality Values (OPC Standard)

- **192**: Good quality
- **0**: Bad quality
- **64**: Uncertain quality

Visual Impact

- With SaveQuality: Gaps shown in trends
- Without SaveQuality: Lines connect all points

Trigger Configuration

Valid Triggers

- Tag values: `Tag.ProcessValue`
- Tag properties: `Tag.ProcessValue.Quality`
- Server objects: `Server.Minute`, `Server.Second`
- Expressions: `Tag.Value > 100`

Trigger Restrictions

- Server-side execution only
- No client tags allowed
- Single trigger per table
- Not available with Normalized

Common Triggers

Purpose	Trigger	Interval
Periodic	Server.Minute	Every minute
Batch end	Tag.BatchComplete	On batch
Shift change	Tag.ShiftNumber	On change
Event-driven	Tag.AlarmActive	On alarm

Performance Optimization

Table Design Guidelines

1. **Group by frequency** - Similar update rates
2. **Limit table size** - 50-100 tags typical
3. **Separate critical data** - Priority tables
4. **Match process dynamics** - Related variables

Deadband Recommendations

Tag Type	TimeDeadband	Notes

Fast analog	00:00:01:000	1 second
Process values	00:00:10:000	10 seconds
Calculated	00:01:00:000	1 minute
Reports	00:15:00:000	15 minutes

Common Configurations

High-Speed Process Data

```
Name: ProcessData
SaveOnChange: True
TimeDeadband: 00:00:01:000
Normalized: True
LifeTime: 30
```

Batch Records

```
Name: BatchHistory
Trigger: Tag.BatchEnd
SaveQuality: True
LifeTime: 365
```

Regulatory Compliance

```
Name: AuditData
SaveOnChange: True
SaveQuality: True
LifeTime: 2555
AutoCreate: True
```

Exporting a Table to CSV

A Historian table exports the period you select to a CSV file, for example with the `SaveToTextFile` method shown in Historian Export to CSV Example. From FrameworX 10.1.6:

- The export writes the whole period. In 10.1.5, an export of a long period stops without a warning after the first 100,000 rows of a Standard table, or after 500,000 rows per tag of a Normalized table. This applies to tables stored in the built-in Tag Historian database. Tags read from an external historian or a Tag Provider are still bound by the read limits of that system.
- The export includes every tag of the table. In 10.1.5, a table with more than 15 tags exports without the values of the 16th and later tags.
- The rows of the file are in time order when the tags of the table are stored at different times, for example in a Normalized table. In 10.1.5, they come out grouped by tag in this case.
- Each value appears under its own column header. In 10.1.5, every row of a Normalized table export carries one extra number ahead of the tag values, which pushes the values after it under the wrong headers. Values also land under the wrong headers when a Normalized export mixes tags of different value types, for example digital and analog tags, or includes a tag with no samples in the period.
- Exports of Standard tables keep the same column format.

Troubleshooting

Table not created:

- Check AutoCreate enabled
- Verify database permissions
- Review connection string
- Check table name validity

No data archived:

- Verify trigger configuration
- Check TimeDeadband setting
- Confirm tags assigned to table
- Review SaveOnChange setting

Excessive storage:

- Increase TimeDeadband
- Add tag deadbands
- Enable normalized schema
- Reduce retention period

Missing data points:

- Check SaveQuality setting
- Review trigger timing
- Verify tag quality
- Examine deadband values

Best Practices Checklist

- **Plan table structure** - Group logically related data
- **Set appropriate deadbands** - Balance detail vs storage
- **Use SaveQuality** - Track data reliability
- **Configure retention** - Manage storage growth
- **Document tables** - Use Description field
- **Test before production** - Verify archiving behavior
- **Monitor performance** - Track storage and query times
- **Regular maintenance** - Archive or purge old data

In this section...
