Local Historian

© 2025 PTC Inc. All Rights Reserved.

Table of Contents

Local Historian ... 1
Table of Contents ... 2
Local Historian ... 4
Overview ... 5
Applications ... 5
Architectural Summary ... 6
General Operation ... 7
Recommended System Configuration ... 9
Creating a Datastore ... 10
Datastore View ... 14
Configuring a New Tag Group ... 15
Defining Historical Tags ... 15
Historical Tag Properties ... 18
Supported Data Types ... 21
CSV File Import / Export ... 21
Viewing Historical Data ... 23
Changing the Datastore ... 26
Moving and Deleting Tags from the Datastore ... 27
Deleting the Datastore ... 28
Datastore and Server Project Synchronization ... 30
Unmapped Tags ... 31
Project Settings ... 32
Administrative Settings ... 33
Client Interfaces ... 34
Browsing ... 35
IOPCHDA_Server::GetHistorianStatus ... 36
IOPCHDA_Server::ReadAtTime ... 36
IOPCHDA_Server::ReadProcessed ... 37
IOPCHDA_SyncRead::ReadAttribute ... 38
OPC HDA 1.20 ... 39
Viewing Archive Data Using Import ... 40
Importing from a Network ... 42
Backing up Archive Files ... 44
Estimating the Datastore Size ... 45
Licensing ... 47
Troubleshooting ... 48
System Tags ... 50
Event Log Messages ... 51
Attached import file <filename>. ... 51

Collection for all items has resumed. Adequate free disk space is available. 52
Collection for all items has resumed. No free space limit is being enforced. 52
Collection for all items has stopped due to low disk space. 52
Configured historical tag count is now within the count allowed in Time-Limited Mode. 52
Configured historical tag count is now within the count specified by the installed license. 53
CSV import ignoring duplicate item <item reference>: the item was specified more than once. 53
CSV import failed: invalid or missing CSV file header. 53
CSV import of item <n> failed: <item reference> is not a valid static or dynamic item reference. 54
CSV import for item <n> failed: an item reference is required. 54
CSV import did not load any valid tags. 54
Data for historical tag <item reference> will not be persisted nor available to HDA clients because the Time-Limited Mode has been exceeded. 54
Data for historical tag <item reference> will not be persisted nor available to HDA clients because the licensed count has been exceeded. 55
Datastore creation failed in <location>, changing location to <location>. 55
Detached import file <filename>. 55
Disk access restored, collection for all items has resumed. 55
Failed to attach import file <filename> due to time span overlap. 56
Failed to attach import file <filename>. The file is invalid. 56
Failed to import item <item reference>: the item is already defined in group <group name>. 56
Failed to load datastore file <filename>. The file is invalid. 56
Failed to store <n> value(s) due to incoming buffer overflow. 57
File I/O error on <filename>, stopping data collection. 57
File <filename> was removed from the datastore. 57
Historian cannot monitor import directory <filename>. 58
Historian monitoring import directory <location>. 58
Historian Service starting. 58
Historian Service stopping. 58
A historical tag mapped to <item reference> is already defined. 59
Historical tag <item reference> was modified to synchronize with persisted data. 59
Import of item <n> failed: invalid deadband specification. 59
Missing historical tag <item reference> was generated from persisted information provided by the data-store. 60
Rejected request to store <n> value(s) due to out of order timestamp. 60
Rejecting request to roll over the active file: a rollover occurred within the last <n> seconds. 60
Rejecting request to roll over the active file; the active file contains no values. 60
Retention policy enforcement removing oldest file, which contains data from <time start> UTC to <time end> UTC. 61
Rolling over the active datastore file. 61
Update of item <n> failed: invalid deadband specification. 61
Index 62

Local Historian

Help version 1.084

CONTENTS

Overview
What is the Local Historian?

Configuring a New Datastore
How do I configure a new datastore in the Local Historian?

Local Historian Configuration
Where do I access the Local Historian in the server?

Modifying the Datastore
How can I modify the datastore?

Estimating the Size of the Datastore on Disk
How can I calculate the size of the datastore on disk?

Datastore and Server Project Synchronization
What are scenarios in which the datastore and server project may get out of sync?

Viewing Historical Data
How do I view the data collected for Historical Tags?

Client Interfaces
Which client interfaces does the Local Historian support?

System Tags
What System Tags does the Local Historian support?

Event Log Messages
What messages does the Local Historian produce?

Overview

The Local Historian is an optional feature of the server that collects, persists, manages, and serves historical process data to clients with minimal configuration. The Local Historian offers the following features:

- Ability to collect data consisting of a value, quality, and timestamp from any data source in the server (e.g. drivers, plug-ins, or system tags)
- Collection from both static and dynamic server tags
- Persistence to a volume on the local machine, which can be a fixed drive or removable media
- Access to historical data via OPC HDA 1.20
- Support for data timestamps with one millisecond resolution
- Configurable data collection scan rates, as frequent as 10 milliseconds
- Support for collection deadband
- Configurable data retention policy
- Ability to import historical data that has been backed up and removed from active use
- Built-in historical data viewer for quick troubleshooting
- Historical tag data import and export with CSV file format
- Tiered licensing for up to 10,000 tags

See Also: *Applications*, *Architectural Summary*, *General Operation*

Applications

The Local Historian is useful in the following applications:

- For customers seeking a process-improvement tool, it can be used to collect data for isolated sub-systems too difficult to integrate into a centralized historian.
- For large organizations where Enterprise Historians are managed by IT, the Local Historian can be configured quickly and used by Operations to troubleshoot a process without IT involvement. It can be used to verify that data collected by the server matches data in the Enterprise Historian.
- For regulated environments (like EPA emissions regulations), it can capture historical data local to the process, guaranteeing that data is not lost if remote connectivity goes down.
- For distributed environments, it can collect and store data local to the equipment, and allow remote clients to access both real-time and historical data on demand. This eliminates the need to manage centralized historians, decreasing maintenance costs and removing a single point of failure.

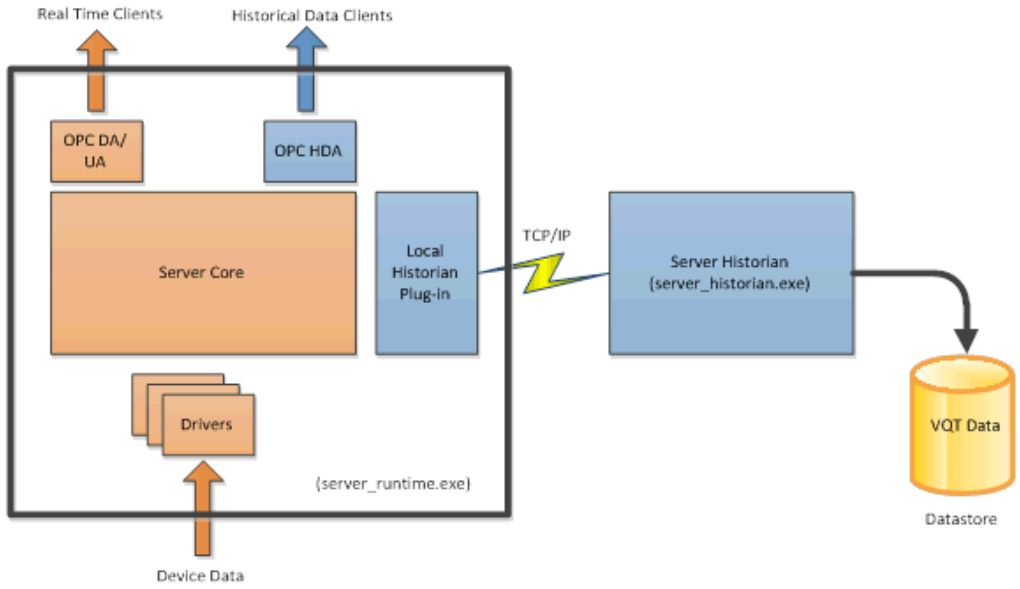
See Also:

Overview
Architectural Summary
General Operation

Architectural Summary

The Local Historian includes three main components:

- The server plug-in (local_historian.dll) is responsible for:
  - Configuration of the historian
  - Data collection from the server runtime
  - Generation and maintenance of the Historical Data Access (HDA) namespace
  - License enforcement.
- The OPC HDA Server (hdaserver.dll), which:
  - Implements the OPC HDA 1.2 interface specification
  - Provides client applications access to the HDA namespace and underlying historical data.
- The historian system service (server_historian.exe), which:
  - Manages a multi-file historical data archive, referred to as a “datastore”
  - Buffers and stores data collected by the plug-in
  - Processes read requests from the HDA server.



Note: The OPC HDA server is identified by the following Programmatic Identifier (ProgID):
• For KEPServerEX: Kepware.KEPServerEX_HDA.V6
• For ThingWorx Kepware Server: PTC.ThingWorxKepwareServer_HDA

See Also:

Overview
Applications
General Operation

General Operation

This section attempts to explain how the three components described above work together to form the basis of the Local Historian by describing some of the functions that they perform as a unit. This topic also serves as an introduction to the terminology used in the remainder of this document.

Initialization
Startup
Data Collection
Data Retrieval
Import
Shutdown

Initialization

A datastore configuration must be created using the Local Historian plug-in from within the Server Configuration user interface. Details of this are covered later in the document. When a datastore is configured and a runtime connection with the Server Configuration exists, the Server Historian Service starts as directed by the plug-in. At this time, the configured archive location is transferred from the plug-in to the service where it is created and initialized. Each datastore created is assigned a random 64-bit number to uniquely identify it from any other datastore. This number is known as the “datastore ID.” The ID is appended to the user-supplied archive location and used as the folder in which the archive files are maintained. This ID is stored within all files that are created in support of the datastore to ensure they belong to that datastore.

Two files exist in the archive location after initial datastore creation:

- <datastore id>.name
- <current time>.active

The placeholders shown are replaced with actual text that may look like:



The .name file stores the set of historical tag names and the associated metadata. The metadata for each tag includes a 64-bit unique identifier to distinguish it from all other tags, the tag’s data type, and whether the tag represents a static or device item reference in the server. Whenever a new historical tag is created, a new entry is made in the .name file. Without this file, there would be no way to associate any of the historical data in the archive with name information required to expose it via HDA, nor would the Historian Service be able to keep the datastore synchronized with the configuration stored in the server project.

The .active file is a writable file used to store incoming time series data as it is collected and forwarded by the plug-in. This file remains writeable and stores data until its size on disk exceeds a configured maximum or the time span of collected data reaches a configured maximum, whichever comes first. When that occurs, the .active file is “rolled over” to start a new file. Within approximately 60 seconds of rolling over, the file extension changes from .active to .tsd (Time Series Data) and becomes read-only. The time lag is necessary to ensure that any pending writes awaiting storage get written to the correct file. Once the .tsd extension is applied, the file has been updated to include all the tag metadata that was present in the .name file at the time of rollover and may be moved or copied out of the archive for backup purposes. The name of the file indicates the local time at which the file was created to identify the range of data it contains when viewed in a list of many .tsd files.

Startup

At system startup with a configured datastore, the Server Runtime loads its project file (e.g. default.opf). Upon detecting that a datastore is defined, the plug-in starts the Server Historian Service. The plug-in establishes a connection to the service and transmits the active datastore configuration. The service examines the existing files in the datastore location and reconstructs the archive from any .tsd files with a matching datastore ID. It then determines whether it needs to perform a rollover to create a new .active file and prepare it to receive data.

When the datastore has been validated and attached by the service, the ID is communicated back to the plug-in through a status message to indicate that it is ready to use. At this time, the plug-in synchronizes its historical tag information with new data loaded by the service. This process establishes a link between historical tags configured in the project and those in the datastore. Following this process, the HDA namespace is constructed and data

collection starts. At this time, the HDA server interface reports a “server up” status to any client that establishes a connection and requests server status information.

Data Collection

Data collection is managed by the plug-in. It creates a server item reference from each historical tag and polls for data at the configured scan rate like any other client. The updates it receives are forwarded to the Server Historian service, where they are buffered and eventually persisted to the .active file. Data buffered for storage remains in memory for no longer than 10 seconds before being flushed to disk. During a buffer flush operation, the disk file may be expanded, to accommodate new data (if necessary). It may also be rolled over if the time span or size of the file has increased beyond configured limits.

The plug-in is responsible for ensuring that, once a tag mapping between the server and the datastore has been established, the original data type assigned when the mapping was created is preserved. The data type cannot be changed from within the historical tag configuration. If the static tag definition that identifies the data source changes, the plug-in is responsible for performing type conversions to ensure the original type is maintained. This is necessary for performance and to ensure the integrity of the data. In this manner, data-type changes are managed, but not recommended because the required conversions can result in range errors that make the stored data unreliable.

Each data update persisted to the datastore consists of three elements: value, quality, and timestamp. These are collectively referred to as a VQT.

New data is recorded in the datastore when:

- an initial update is received from the server when data collection starts
- a value or (OPC DA) quality changes from the last collected VQT
- if configured, the set Interval time has elapsed since the last recorded data

The time stored with each VQT in the datastore uses Coordinated Universal Time (UTC), which is the standard time basis used by OPC HDA. This format makes it possible to record timestamps with one-millisecond resolution. Timestamps received for each successive VQT belonging to each historical tag must be monotonically increasing. If a timestamp is received and it is earlier than one already recorded, it is considered “out-of-order data” and the VQT for that update is dropped. Out-of-order data can occur if the system clock is not functioning properly or is altered while data collection is taking place.

Data Retrieval

An HDA client requesting data from the datastore must first establish an OPC HDA connection to the HDA server and obtain a “handle” to the item or items to be read. It can either browse the HDA namespace to choose a historical tag or provide the fully qualified item reference.

Read requests are handled in the HDA server in the order in which they are received. The requested parameters are evaluated at that level and turned into a generic read request that is first validated by the plug-in to ensure all required tags exist, then sent to the Server Historian. When the service receives the read request, it determines which files in the datastore it needs to open and read, and which parts within each file it needs to read. The read completes in the Server Historian whenever one of the following conditions is satisfied:

- All values within the time range specified by the client have been identified and retrieved from disk.
- The number of values read from disk reaches the limit specified by the client.
- The memory limit dedicated to a single read request is exceeded (150 MB).

The data from the read is packaged by the Server Historian and sent to the HDA server, where it is repackaged for consumption by the client in conformance with the OPC HDA specification.

Import

The Local Historian supports an import feature that allows .tsd files that have been moved out of the datastore to be re-attached. This feature does not support import of historical data from another product. It monitors a user-specified file system location for add/remove files activity. This location cannot be the archive location. When a change is detected, the file set in the import location is synchronized with the files in the archive location to produce a unified data set that can be accessed by an HDA client.

Server Historian Service notifies the plug-in when a change to import information is available and the plug-in requests an update from the Server Historian Service so that it can refresh the HDA namespace. As long as the historical tags in the imported files match those in the active datastore, there is no net change to the namespace. However, if new or conflicting name-ID pairs are detected, the plug-in auto-generates historical tags in the project under the _ImportedTags group. New tags are added to the HDA namespace, but those with conflicting names are internally marked as “conflicted imports,” represented as “Unavailable” in the Historical Tag view, and not added to the HDA namespace.

The plug-in manages the life cycle of the _ImportedTags group. When the import location does not contain files or importing is disabled, the group and all its historical tags are automatically removed from the project. There are no supported user actions at the group or tag level for imported tags. This group is not persisted when the server project is saved.

Shutdown

When the Server Runtime receives a request to shutdown, the Local Historian plug-in is responsible for stopping data collection. Whenever collection stops, a special “no value, no quality” VQT is recorded for each historical tag to indicate a gap in the data stream. There should never be two consecutive VQTs in the datastore for any historical tag with the same value and quality. After sending the final VQT updates, Local Historian uses the messaging interface to tell the Server Historian Service to detach the active datastore, which flushes all pending changes to disk and closes all open files. At this point, the HDA server interface informs any active client connections that the server is going offline by calling the shutdown sink connection point. Local Historian stops accepting new connections and reports “Down” as a server status for any client connection status requests.

See Also:

Overview
Applications
Architectural Summary

Recommended System Configuration

Beyond the standard server system requirements, the following configuration is recommended for optimal performance of Local Historian when working with a high volume of collected data:

- Windows 8.1, Windows Server 2016 or later
- Dedicated hard drive that (to contain the Data Store):
    - Has a speed of at least 7200 RPM or is flash-based
    - Is not hosting the operating system
    - Is excluded from active or passive anti-virus scanning
- Minimum of 4 GB total and 1 GB free RAM when the system is loaded

Best performance is achieved when the system has enough free RAM available to allow the operating system to provide a reasonable file system cache. A 64-bit operating system can take advantage of nearly all unused RAM for file system cache. This cache helps applications avoid lengthy waits when writing to or reading from physical disks. Disk operations satisfied by the file system cache are completed orders of magnitude faster than operations that access the physical disk.

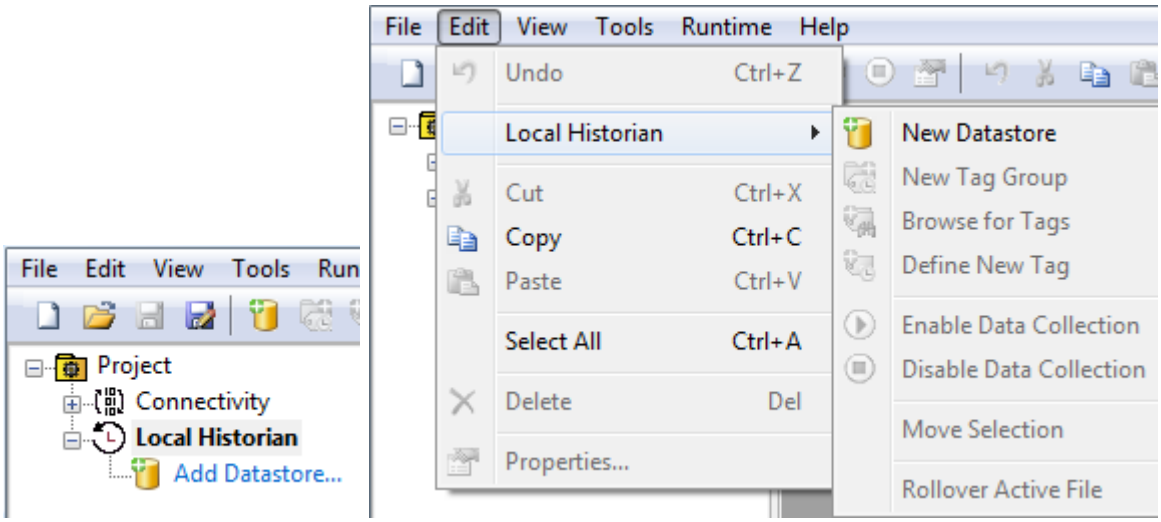
Creating a Datastore

A datastore definition is required to begin collecting and saving data. A wizard is provided to step through the process of defining and creating a datastore. Most of the settings configured in the wizard can be changed later by editing the properties of the configured datastore. The only exception is the archive location, which cannot be changed once assigned.

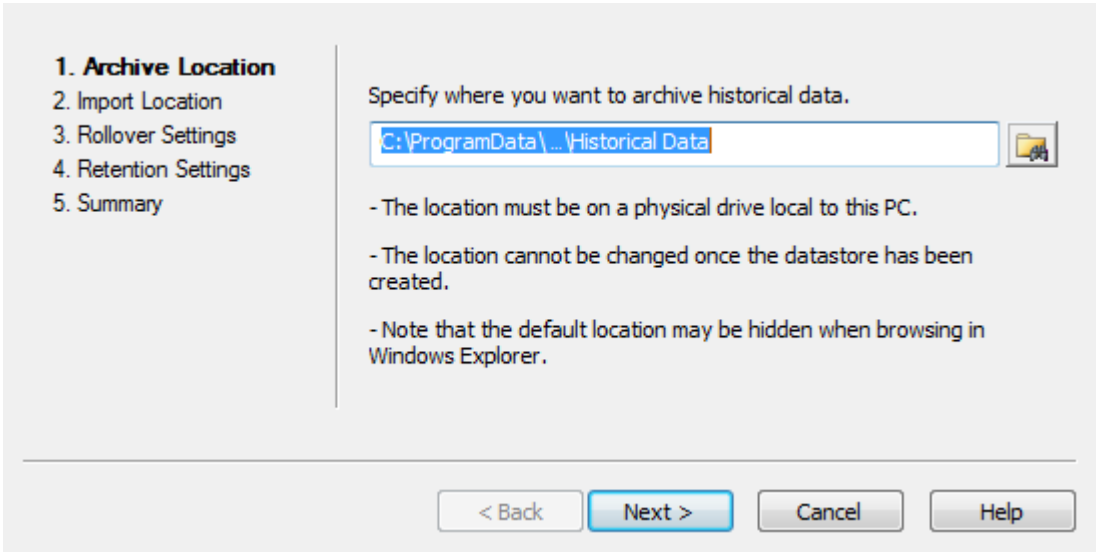
To create a new datastore:

In the server configuration interface, select Local Historian from the drop-down menu on the toolbar.

Click **Add Datastore**, the new datastore icon on the toolbar, or select **Edit | Local Historian |New Datastore** from the menu to launch the new datastore wizard.



New Datastore Wizard - Archive Location



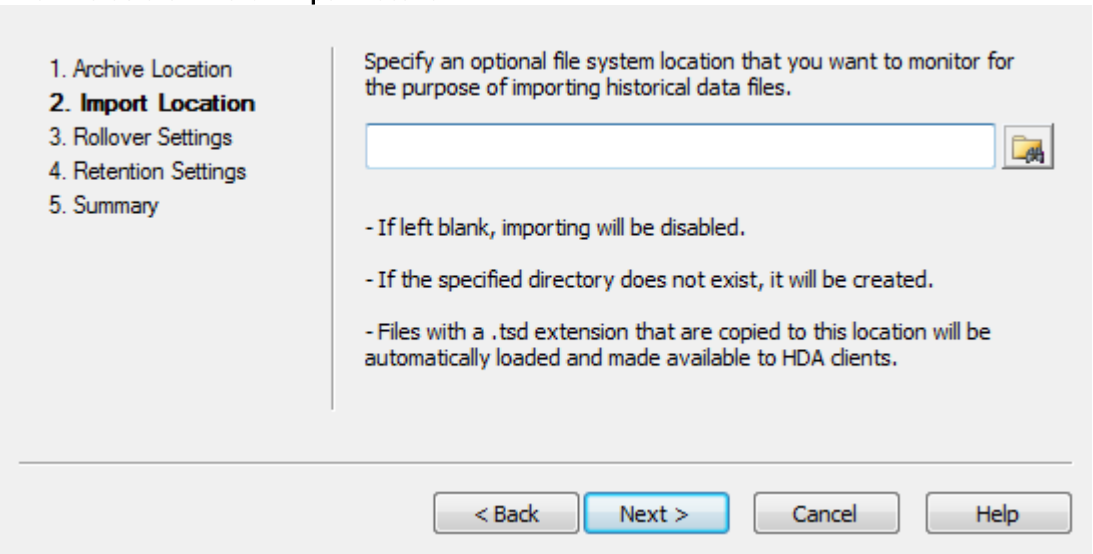
The first page of the wizard specifies the file system location to create the datastore.

Note: It must be on a physical drive that is local to the PC.

Tip: For best performance, the datastore should be on a dedicated drive or a drive that does not host the operating system.

Note: A removable drive can be the archive location, but it is not recommended for performance reasons and potential data loss in the event the drive is removed. USB flash drives can perform especially poorly due to frequent flushing performed by the operating system.

New Datastore Wizard - Import Location



The second page specifies a file system location to monitor for Local Historical datastore files. Existing .tsd files in this location are automatically attached, making their historical data available to clients through HDA.

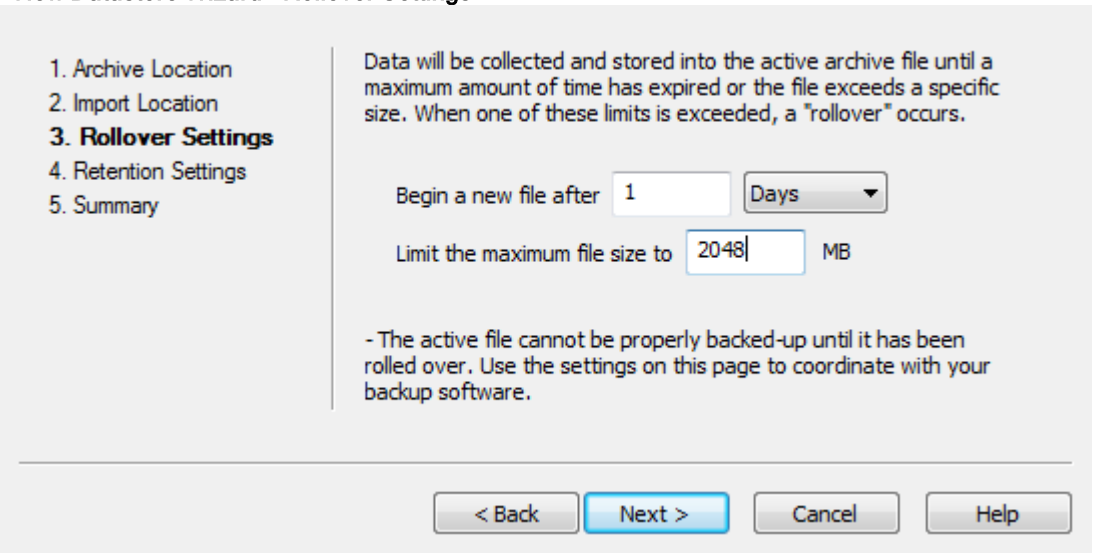
Tip: This provides visibility into older data that has been removed from the archive, a consolidated view across multiple Historian installations and datastores, and a tool for offline analysis by importing .tsd files from production systems into a separate Local Historian install (including a trial version).

Note: The import location cannot be the same as the archive location.

Caution: When using the UNC paths, ensure the path is a trusted and secure location.

Leave the field blank to disable import.

New Datastore Wizard - Rollover Settings

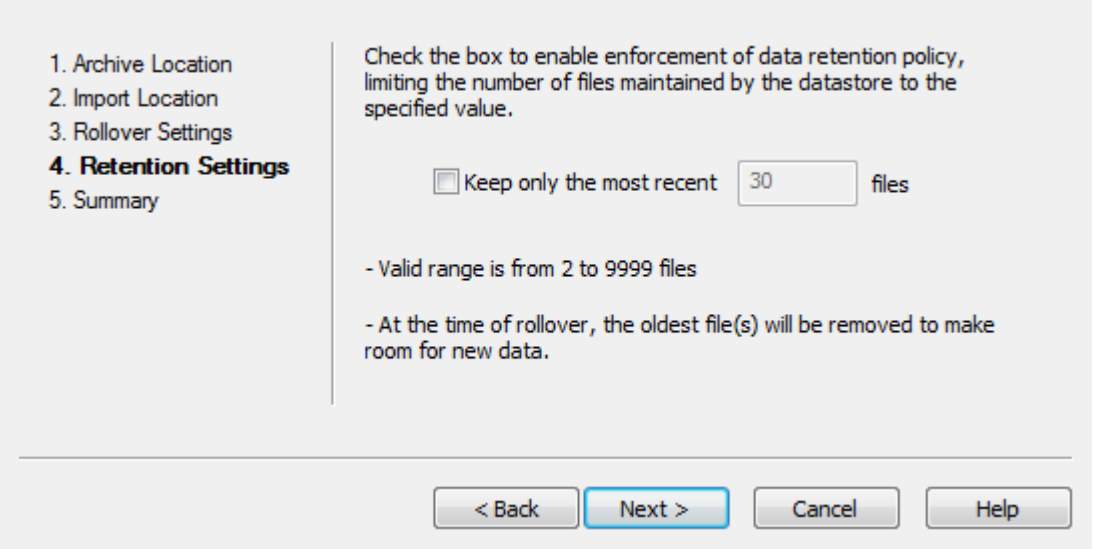


This page configures the conditions that trigger creation of a new datastore file. Datastore files are “rolled over” based on a time or size policy (whichever is reached first). Files may be configured to roll over after a fixed amount of time (from one hour to 365 days) or after reaching a maximum size (100 MB to 2 GB).

Tip: When specifying a maximum size, choose a value slightly smaller than the actual expected size to account for bursts of data that may be generated around the time of rollover. While this should not be a problem in low data volume applications, providing a size tolerance for these circumstances can prevent excessive size overrun.

Once the .active file is rolled over to create a .tsd file, it can be backed up or removed from the datastore.

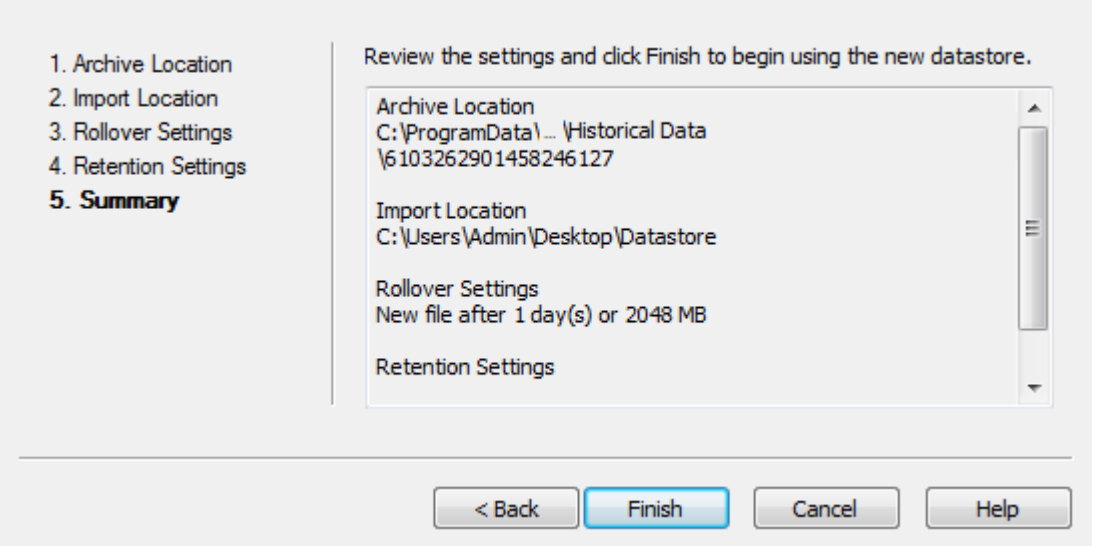
New Datastore Wizard - Retention Settings



This page is used to optionally specify the maximum number of datastore files Historian retains. When a rollover causes the file count to exceed this value, the oldest datastore file is deleted. The valid range is 2 to 9999 files.

Note: Leaving the box unchecked retains all data files created by the Local Historian indefinitely, which consumes disk space. It is recommended that files containing data older than necessary for everyday activities be backed up and removed from the archive. These can be accessed later with the import feature.

New Datastore Wizard - Summary Settings



This page provides an opportunity to review the information specified in the previous pages.

Use the **Back** button to change any settings before finalizing the new datastore configuration.

A numeric identifier is created for the datastore which is appended to the archive location path. This identifier associates the data in the archive with the configuration contained in the server project where the datastore

configuration was created. This allows different archives to be managed in a single location if multiple projects are being used for troubleshooting.

Click **Finish** and the datastore is created on disk.

Proceed with Defining Historical Tags to begin sending data to the new datastore.

Note: If the Server Configuration is disconnected from the server runtime (indicated by “Offline” in the lower right of the main window status bar), the datastore configuration is applied and the datastore created the next time it is connected and the project updated.

See Also:

Datastore Properties
Changing Datastore Settings
Defining Historical Tags

Datastore View

When the datastore is selected in the Project View, the datastore properties are shown in the Detail View. This view is a mix of configurable settings and live status information in a read-only property grid. Selecting each property or section heading updates the description area below the grid to provide information about what's selected.

Persistence	
Datastore location	C:\ProgramData\Kepware\...
Import location	Importing disabled
Available disk space	385.7 GB
Minimum required free space	500 MB
Only log Good Quality Updates	Enabled
Active File Rollover	
Maximum file size (MB)	2048
Maximum collected time span	1
Units of time span	Days
Retention Policy	
Enable retention policy enforcement	Disabled
Number of files to keep	Unlimited
Status	
Historian service status	Started
Licensed tag count	Not licensed
Configured tags	1
Collection rate (10s average)	0.0 B/s
Size on disk	20.06 KB

In addition to the configuration settings, the following status information is provided:

- **Status** includes dynamic information and the datastore, updated twice per second in normal operation. In offline mode, all fields report "Unavailable."
- **Historian Service Status** reflects the current state of the Historian Service. Possible states are:
  - **Started**: The Historian Service is fully operational and HDA Server is available.
  - **Stopped**: The Historian Service is stopped; HDA Server is not available.
  - **Starting**: The Historian Service is in the process of starting.
  - **Stopping**: The Historian Service is in the process of stopping.
  - **Service Disabled**: The Historian Service is not enabled in the Services Control Panel of the Operating System.
  - **Service not Installed**: The Historian Service is not installed or it was deleted.
- **Licensed tag count** reports one of three things, depending on the situation. It can report that no license for the Local Historian is installed. When a license is installed, it reports the number of historical tags collecting data and/or serving data to HDA clients.
- **Configured tags** reports the total count of all historical tags defined for Local Historian, including tags defined in the project as well as tags in the _UnmappedTags and _ImportedTags groups. If this number exceeds the number reported by licensed tag count, some tags are not being collected and not part of the HDA namespace. The event log messages identify tags not being collected.
- **Collection rate (10s average)** reports the input data rate to the Historian Service for the value of the VQT information being stored. The number represents an average of the most recent ten-second interval. Because this number includes only the rate of value data, it is most useful as a quick indicator that data collection is actually taking place. This same information is available at runtime from the _BytesInPerSecond system tag.
- **Size on disk** reports the total amount of disk space the datastore is consuming for all files that it contains.

See Also:

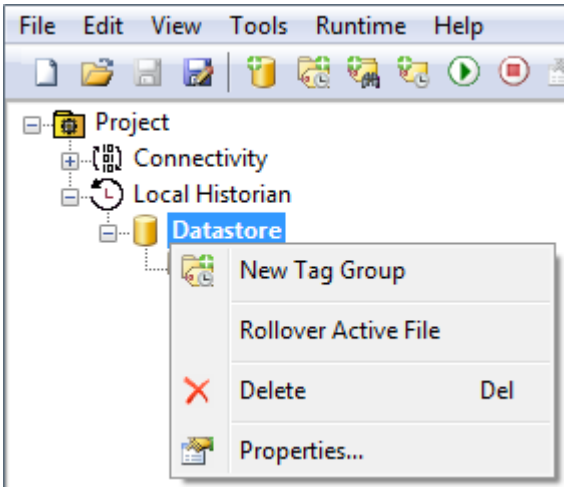
Configuring a New Datastore
Estimating the Size of the Datastore

Configuring a New Tag Group

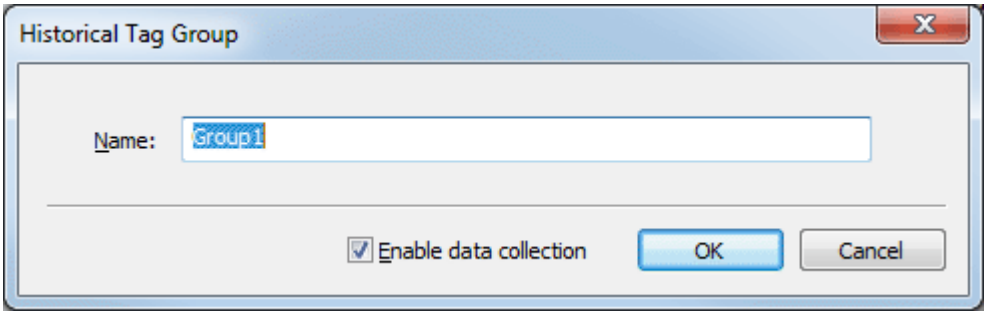
Tag groups are used to organize tags in the Local Historian. For example, Historical Tags for which users would normally enable or disable data collection at once should be placed in the same tag group. This allows users enable or disable the group rather than individual tags. The tag group “Group1” is created by default.

Note: Tag groups are not visible when browsing historical data via the client interfaces. They are only for organizational use in the Local Historian.

1. To create a new tag group, right-click on **Datastore** and then select **New Tag Group**. Alternatively, select the **New Tag Group** icon in the toolbar.



2. In the Historical Tag Group dialog, enter a name or accept the default name.
3. Confirm whether data collection should be enabled or disabled, then click **OK**. The default is enabled.



4. Once finished, click **OK**.

Defining Historical Tags

A historical tag references a server item using a static tag name or a device address. It defines the identity and collection parameters of a data source provided by the server. Historical tags are organized within tag groups. New groups can be created as necessary and tags may be moved freely across groups.

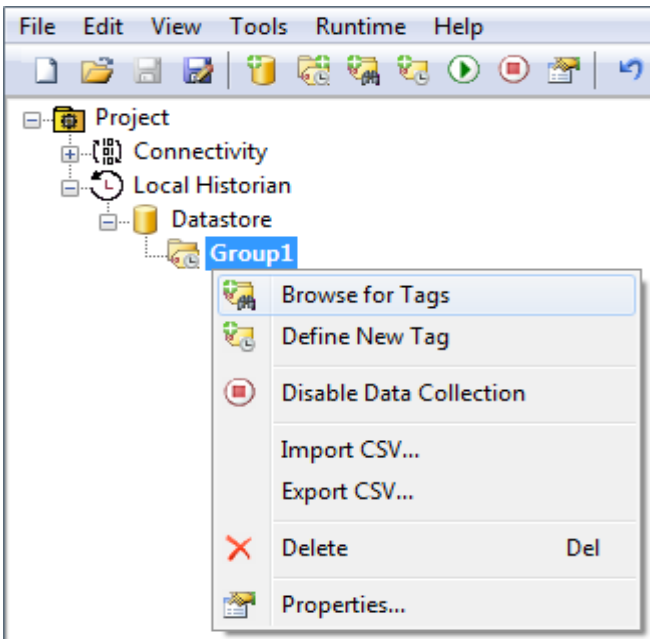
There are two ways to add historical tags to a project:

1. Browsing for and selecting one or more static tags already defined in the server.
2. Defining them manually using a device address (“dynamic” addressing).

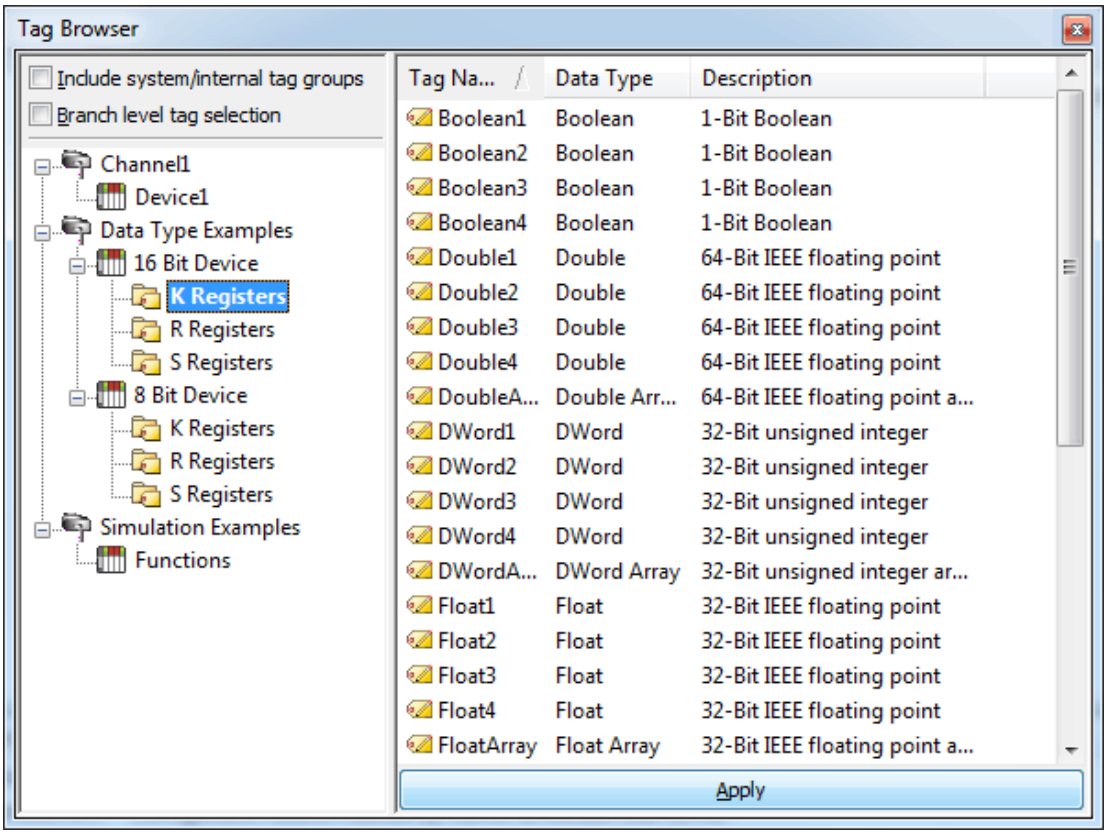
Adding Tags Using the Tag Browser

Only static tag references may be added via the tag browser. Static references identify data sources already defined in the server: user-defined static tags, system tags, or tags defined in other plug-ins.

1. To add new tags for data collection via browse, right-click on the historical tag group and select **Browse for Tags**. Alternatively, select the **Browse for Tags** icon from the toolbar.



2. Use the **Tag Browser** to select one or more tags from the server namespace.



Tip: Multiple tags can be selected. Use the Shift and Ctrl keys to multi-select with standard Windows functionality.

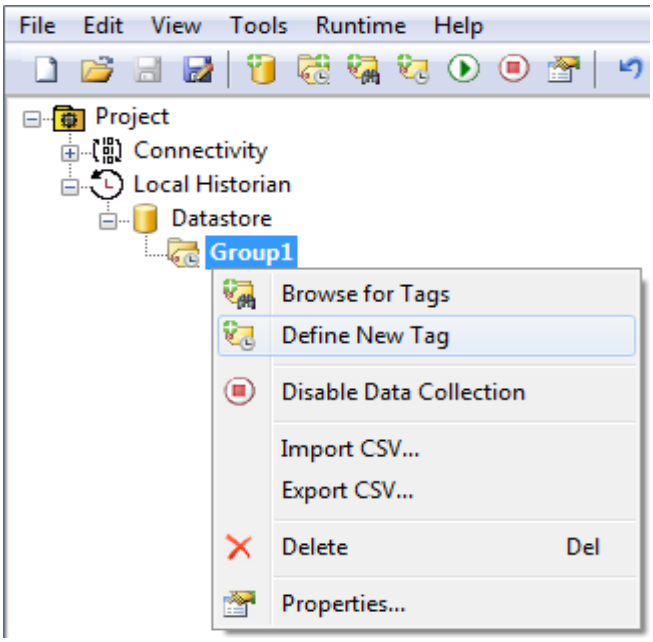
Once finished, click **Apply**.

3. Configure the collection parameters for the selected tags, as discussed in Historical Tag Properties.

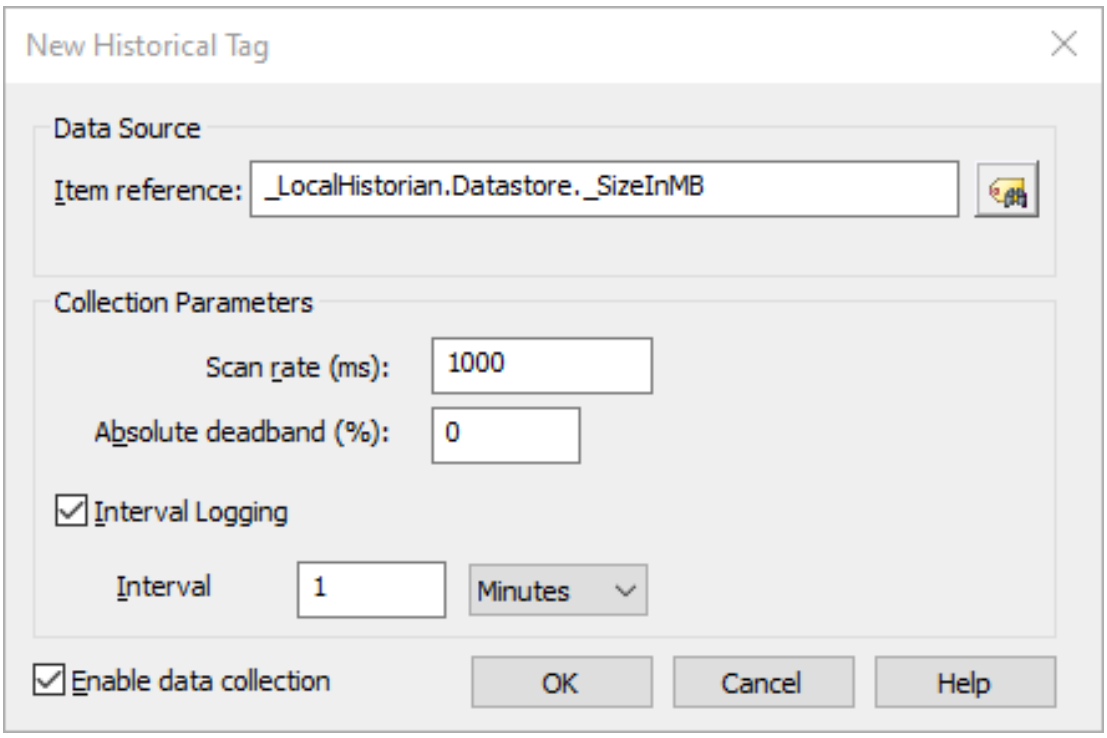
Adding Tags Manually

Historical tags that reference data obtained by direct device addressing must be added manually, one at a time.

1. To manually add a new Historical Tag for data collection, right-click on the historical tag group and select **Define New Tag** or click on the new tag icon .



2. In the New Historical Tag dialog, specify the data source and collection parameters.



3. To add a tag, enter the tag's fully-qualified address (such as "Channel.Device.Tag") or click the browse button  to the right of the Item Reference field to launch the Tag Browser.

   Tip: As an address is entered in the Item Reference field, it initially appears in red. Once the address reflects a valid tag, the text color changes to black. While the text is red, an error message appears under the Item reference field describing why the address is invalid.

   Once finished, click **OK**.

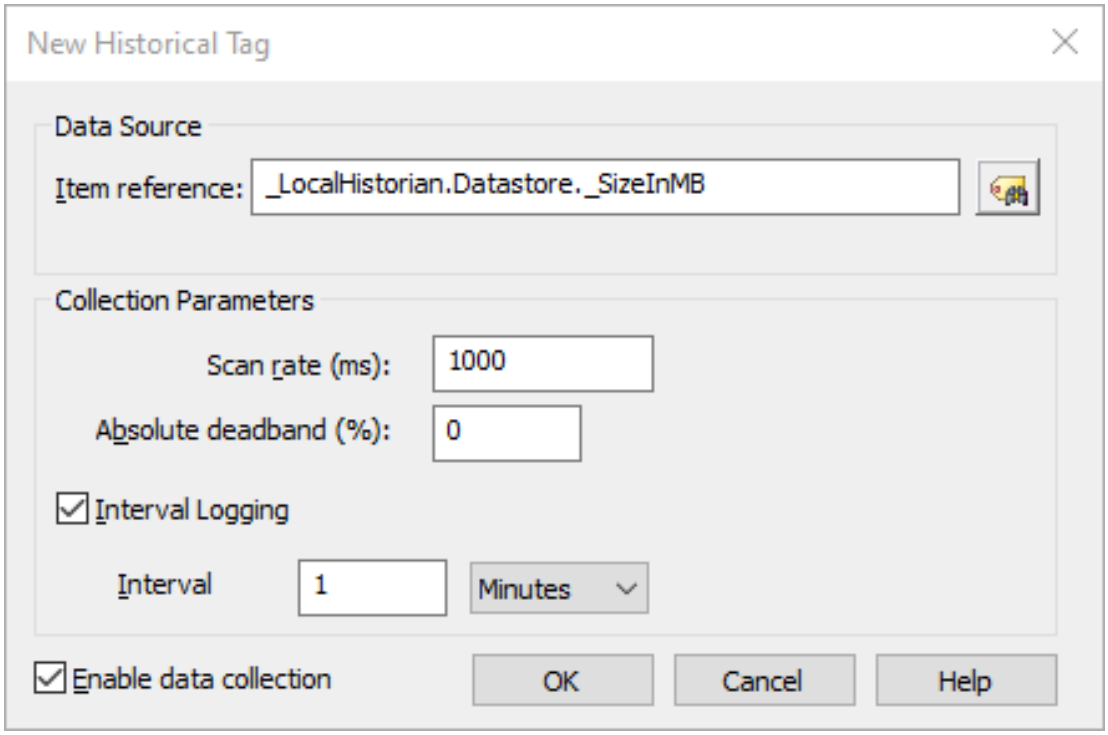
4. Configure the collection parameters for the selected tags, as discussed in Historical Tag Properties.

See Also: Historical Tag Properties

Historical Tag Properties

The collection settings of historical tags may be changed using the Historical Tag Properties dialog. Tag properties can be updated for just one tag or several. If updating several tags at the same time, changes apply to ALL the selected tags.

Select the tag(s) to update and click the properties  button or right-click and select **Properties...**.



Collections Parameters

Item reference: The fully qualified item reference given to the server when the Local Historian requests data collection. This is the name that appears in the HDA namespace a client sees when browsing for historical tags. If multiple tags are selected, the text reads “Multiple selected”; otherwise, it displays the item reference assigned to the selected tag.
Note: This field is disabled; the item reference cannot be changed once it has been assigned. The Item reference automatically updates with changes in channel, device, tag group, or tag name when changes are made in the Channels / Devices view.

Data type: The data type is determined from the data source. For a static tag reference, it is the data type assigned to the tag itself. For a device address, it is the default type associated with the address as determined by the device driver. To ensure data integrity, the data type cannot be changed once assigned to a historical tag.
See Also: *Supported data types*

Scan rate: The speed, in milliseconds, at which the data source is read. Data from the data source is sent to the Local Historian at this rate, but only changed values are written to the datastore.

Absolute Deadband Parameters

Absolute deadband: Sets the amount of change, as a percentage of total value range, to define one value as different from the previous one. Zero defines any two values as different if they are not exactly equal. Not all data types support deadband. Refer to the data type table for more information.

Interval Logging Parameters

Interval Logging: Enables logging the tag at least every interval configured. If the interval has elapsed with no data change detected, the current value is logged to the datastore. Any data changes reset the time until the next interval logs.

Interval: If Interval Logging is enabled, this configures the minimum time between logged updates. Time can be configured in seconds, minutes, hours or days. The valid range is 1 second to 31 days.
Notes:

1. Interval logging is intended to allow aggregate functions for infrequently updating historical tags to calculate over datastore rollover boundaries.
2. To preserve the order of timestamps when Interval Logging is enabled, updates are provided a server timestamp regardless if it was a quality, value, or interval expired update.

Enable data collection: Collection must be enabled at both the Historical Tag and Historical Tag Group levels to collect and store data in the datastore. Disabling collection at the tag level removes the item reference from the server so that the source of data for the tag is no longer scanned (such as during an upgrade or system configuration that would produce invalid data). Disabling collection at the group level does the same for all tags in the group. Enabling collection at the group level only starts data collection for tags that also have data collection enabled.

Supported Data Types

Data Type	Value Range	Supports Deadband
Boolean	2-byte value; 0x0000 or 0xffff	No
Byte	0 to 255	Yes
Char	-128 to 127	Yes
Word	0 to 65535	Yes
Short	-32768 to 32767	Yes
DWord	0 to 4,294,967,295	Yes
Long	-2,147,483,648 to 2,147,483,647	Yes
QWord	0 to 18,446,744,073,709,551,615 or 0 to 2^{64}-1 Unsigned 64-bit integer data	No
LLong	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 or -2^{63} to 2^{63}-1 Signed 64-bit integer data	No
Float	32-bit Real Value IEEE-754 standard definition	Yes
Double	64-bit Real Value IEEE-754 standard definition	Yes
Date	See Microsoft® Knowledge Base	No
String	Null-terminated Unicode string	No

Note: Arrays, LBCD, and BCD data types are not supported.

CSV File Import / Export

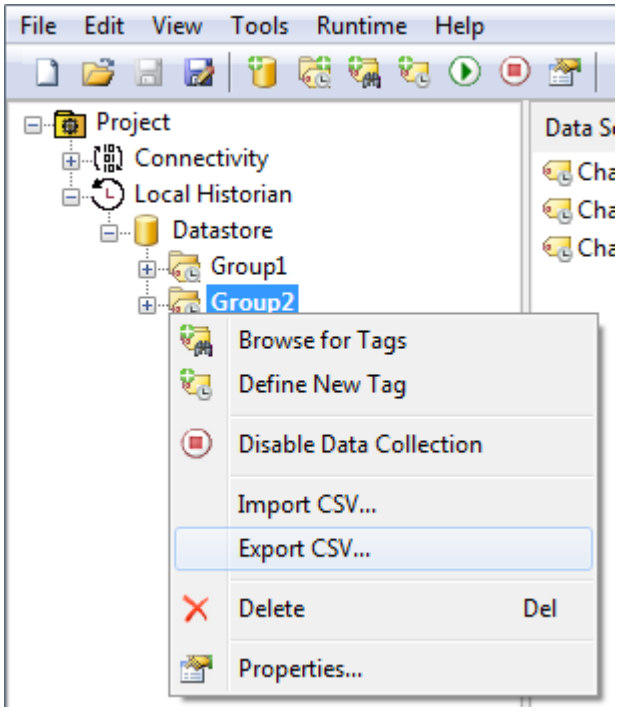
The Local Historian supports importing and exporting a historical tag group's tags using a Comma-Separated Value (CSV) file. This allows users to add historical tags to an existing historical tag group or to edit the properties of existing historical tags.

Note: It is not possible to import or export tags from the _UnmappedTags or _ImportedTags groups. CSV import does not create new historical tag groups.

Tip: One way to create an import CSV file is to export the historical tags from an existing historical tag group and use that as a template.

Creating a Template

1. Select an existing historical tag group. Right-click on the historical tag group to launch the pop-up menu, and select **Export CSV...**.
2. Browse to the directory in which to save the CSV file, and enter a filename.
3. View or edit this CSV file in text or spreadsheet software *(see CSV File Format for guidelines)*.
4. When finished, be sure to save the file in CSV format.



CSV File Format

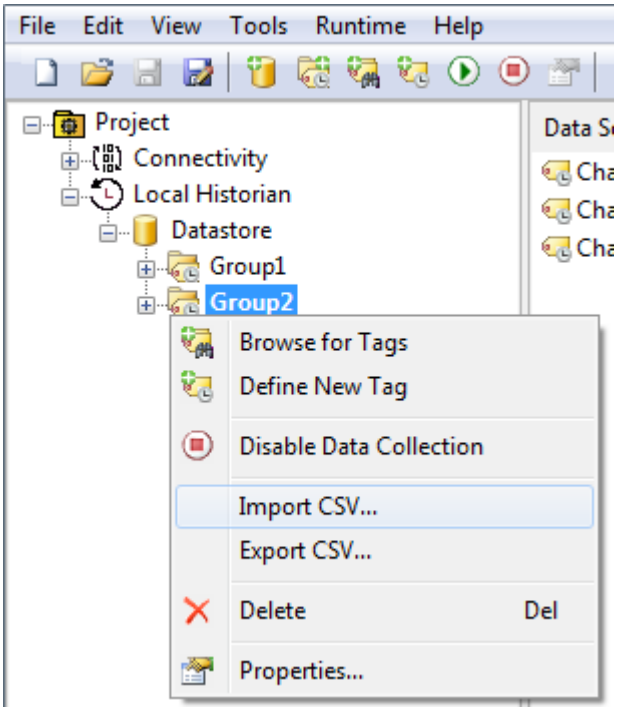
- Each record must be on its own line.
- Field titles in the header row may be in any order.
- The only required field is Item Reference. Scan Rate, Deadband, and Enabled are optional.
- Lines beginning with a semicolon are considered comments.
- Items with an optional field that is blank or missing are assigned the default property value.

Exporting a Historical Tag Group

Exporting generates a CSV file that contains a list of historical tags and the associated parameters.

Importing a CSV File into a Historical Tag Group

A CSV file can be imported into the Local Historian by right-clicking on the desired historical tag group and selecting **Import CSV...**. This adds the tags specified in the CSV file to the historical tag group. If a tag already exists in the connection, its properties are overwritten with the values from the imported CSV file.



Delimiter Characters

The Local Historian supports CSV files that use either a comma or semi-colon as the delimiter character. If using a CSV file using any other character, perform a search-and-replace on the delimiter to replace it with a comma or semicolon. For information on specifying the character to use as the server-specified delimiter, refer to Options-General in the server help file.

CSV Import Rules

Some rules must be followed when creating and importing a CSV file. The following circumstances result in a failure to import one or more items and fails to generate an error-level Event Log message:

- If the CSV file has an invalid or missing header entry, the import of the CSV file fails.
- If the CSV file has a valid header, but no valid historical tag entries, the import of the CSV file fails.
- If the import CSV file defines a historical tag that exists in a historical tag group other than the group into which you are importing, the import of that tag fails.
- If the import CSV file defines a historical tag with no item reference, the import of that item fails.
- If the import CSV file defines a historical tag with an invalid static or dynamic item reference, the import of that item fails.
- If the import CSV file defines a historical tag that specifies an invalid deadband (less than 0.0 or greater than 1.0), the import of that item fails.
- If the import CSV file defines the same historical item reference in multiple rows, the item is added (or updated) one time and the duplicate entries are ignored.

Viewing Historical Data

The Local Historian provides a rudimentary data viewer that supports basic data queries to the datastore. It can be used to verify that data is being collected as expected or to examine data collected in the past. All data stored in the datastore is time-stamped with a UTC basis. The viewer converts the stored timestamp to the local time of the viewer.

Select a historical tag in the tree of the Project View (left) to see that tag's ten most recently persisted in the Detail View (right).

Viewing Historical Tags

When historical tags are added, they appear in the Project View in the tree structure beneath their group. When that group is selected, the list of tags in the group is shown in the Detail View, where their properties are listed in sortable columns.

Valid, Enabled Historical Tag mapped to a confirmed server item reference and has data collection enabled

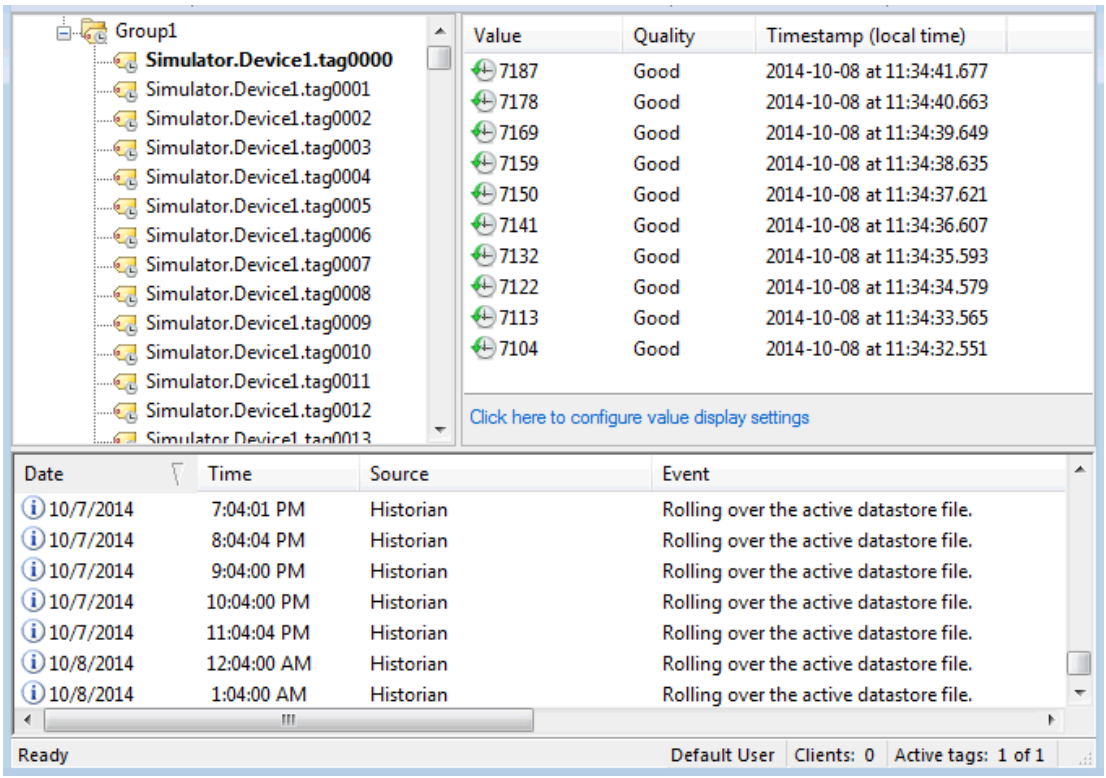
Valid, Disabled Historical Tag mapped to a confirmed server item reference with data collection disabled at the tag level

Valid, Enabled Historical Tag in a Disabled Group mapped to a confirmed server item reference with data collection enabled at the tag level, but disabled at the group level

Invalid Historical Tag mapped to a missing or invalid server item reference

Notes:

1. An invalid item reference results from changing something that affects the validity of the reference. For example, if a tag is deleted, it invalidates the historical tag mapped to it. (Data collection must be disabled or the Server Configuration in offline mode to delete a static tag providing an item reference to a historical tag.)
2. A large number of invalid references when the Server Configuration is connected to the runtime can cause performance problems when attempting to show the list view sorted by State.
3. The graying effect to indicate disabled applies to valid and invalid tags.



The Value column shows the data stored for that tag at that timestamp.

The Quality column displays the assessment of how reliable that data is from that timestamp. Options are Good, Bad, Uncertain, and No Value.

The Timestamp (local time) column provides the exact timestamp the tag data was returned.

The following icons provide a quick visual indication of the VQT for each tag record:

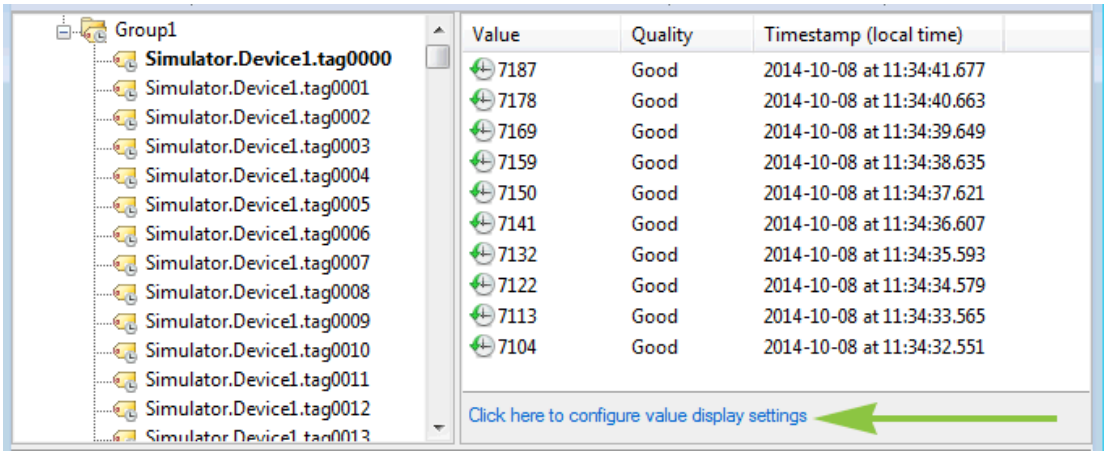
Good The data source produced the data it was asked to provide

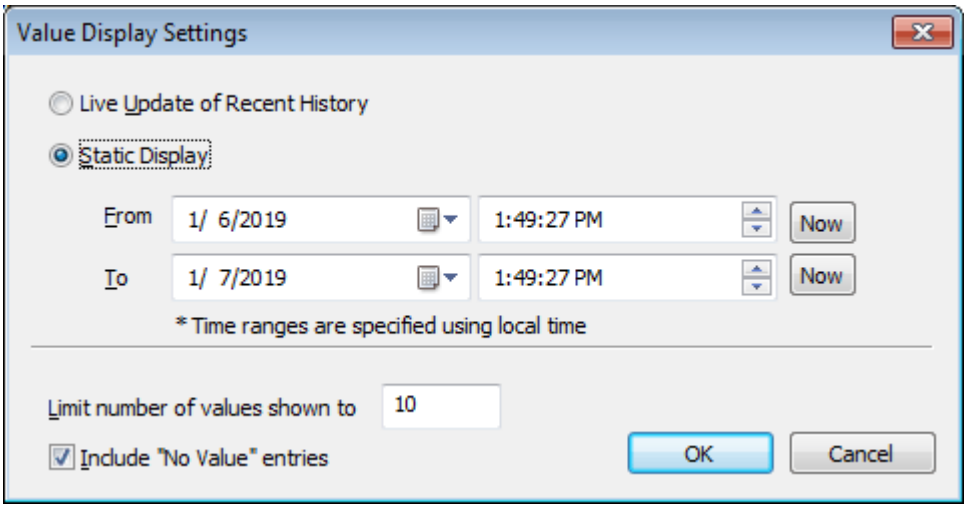
	Bad	The data source failed to respond to a request for data
	Uncertain	The data source produced an update that may include a range error in a data type conversion between the device and the historical tag
	No Value	Entry recorded when collection stopped

Note: Data collection must have been enabled for the tag and its group at some point for there to be any data for that historical tag in the datastore.

Changing the Display Settings

By default, the VQTs are ordered with the most recent data at the top of the list. The number, order, and time range of data displayed can be configured by clicking the link beneath the list to access the Display Settings dialog.

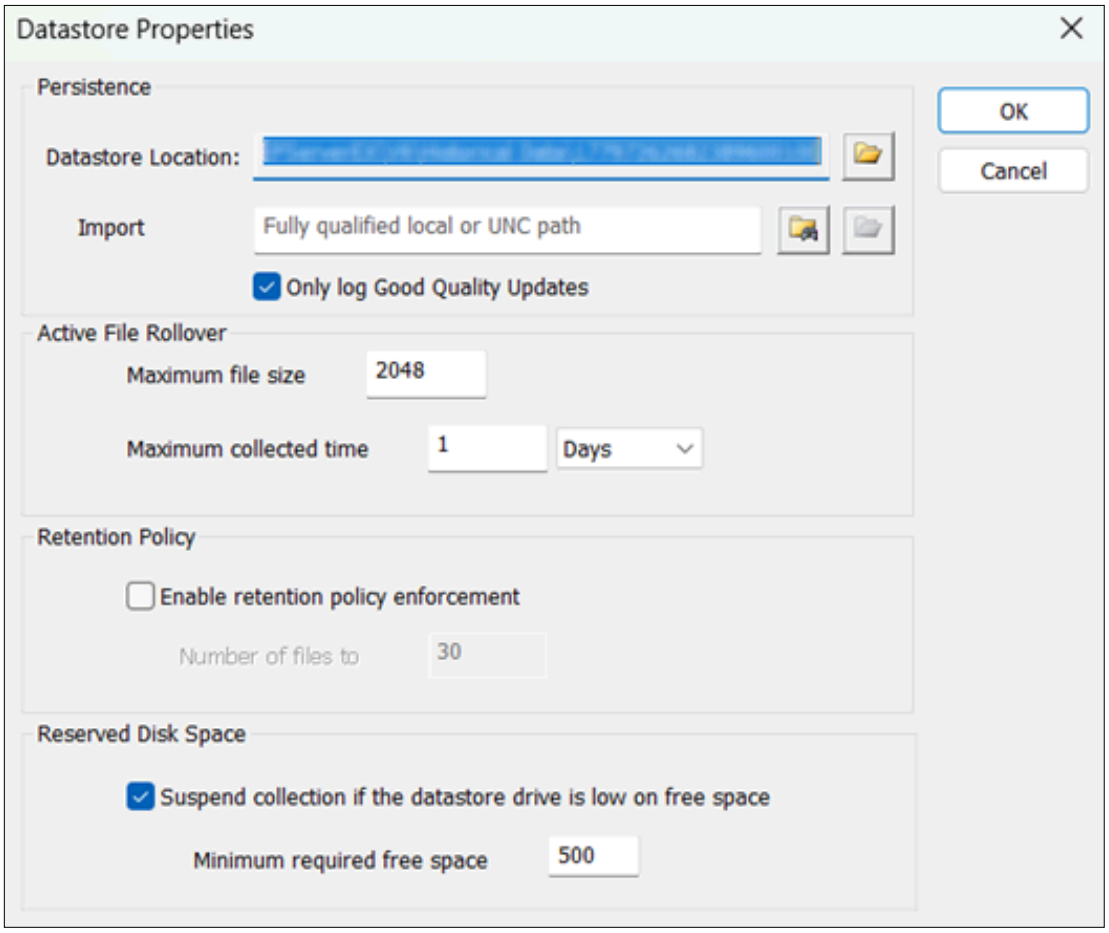




- **Live Update of Recent History** refreshes the view whenever new data is recorded in the datastore for the selected historical tag.
- **Static Display** specifies a time range for which data is requested from the datastore. Enter the From and To dates or use the calendar icon to select the range. Use the up / down arrows to define the start and end times. The **Now** button sets the date and time for the selected range (From or To).
  Note: The order of the records displayed matches the “From”/”To” order of the timestamps. If From timestamp comes before To timestamp, the list order is oldest to newest (top to bottom).
- **Limit number of values shown to** sets the maximum tag records shown. Acceptable range is 0 to 9999.
  Note: No entries are displayed when 0 is specified.
- **Include "No Value" entries** filters those tag records where collection stopped. Disable the setting to filter out those records. The default setting is enabled.

Changing the Datastore

The datastore configuration can be updated by selecting the datastore in the Project View tree and clicking the Properties button or right-clicking and choosing **Properties...**.



Persistence

Datastore Location: This field is read-only because the datastore location cannot be changed once it has been established. Use the Open Datastore location button to explore the datastore directory with Windows Explorer.

Import: This field specifies a file system path that the Historian monitors for datastore files. Existing .tsd files in this location are automatically read, making the historical data available to clients through HDA. Use the **Browse** button to locate and select the directory to be monitored. Use the **Open Import location** button to explore the import directory with Windows Explorer.
Tip: This provides visibility into older data that has been removed from the archive, a consolidated view across multiple Historian installations and datastores, and a tool for offline analysis by importing .tsd files from production systems into a separate Local Historian install (including a trial version).
Note: The import location cannot be the same as the archive location. Leave the field blank to disable import.

Only log Good Quality Updates Enabling this setting means that only data changes associated with a Good Quality from the server are logged. Quality only changes are ignored and any update passed into the Historian without Good Quality is ignored.
Note: If an item is orphaned, the quality shows No Value to indicate that state to any clients.

Active File Rollover

Maximum file size: This setting determines the maximum size to which the .active archive file can grow before it is rolled over to a read-only .tsd file. This setting is observed as closely as possible, but data waiting to be stored at the time of rollover must be written to the .active file to maintain the correct time span, so size limit is not exact.
Note: If this setting is changed when the .active file is larger than the new setting, rollover is immediately initiated.

Maximum collected time: This setting determines the range of time the active file holds before it is rolled over. When the range of time from the first stored VQT to the most recent VQT exceeds the maximum collected time span setting, the file is rolled over. The unit of time is selected through the drop-down menu.
Note: The VQT that causes the span to be exceeded is stored in the new .active file created by the rollover process.
Tip: Whenever the active file is rolled over, an informational event log message is posted (see Event Log Messages).

Retention Policy

Enable retention policy enforcement: This setting configures the Historian Service to delete the oldest .tsd file(s) once the limit set by Number of files to keep is reached. Each deletion is reported by an informational event log message identifying the time range of data removed from the archive *(see Event Log Messages)*.

Reserved Disk Space

Suspend collection if the datastore drive is low on free space: This setting configures the Historian Service to stop collecting if the datastore size reaches the threshold set in the Minimum required free space (MB). When this limit is reached, no new data is persisted until the amount of free space meets the minimum requirement. Event log messages are posted for these two events *(see Event Log Messages)*.

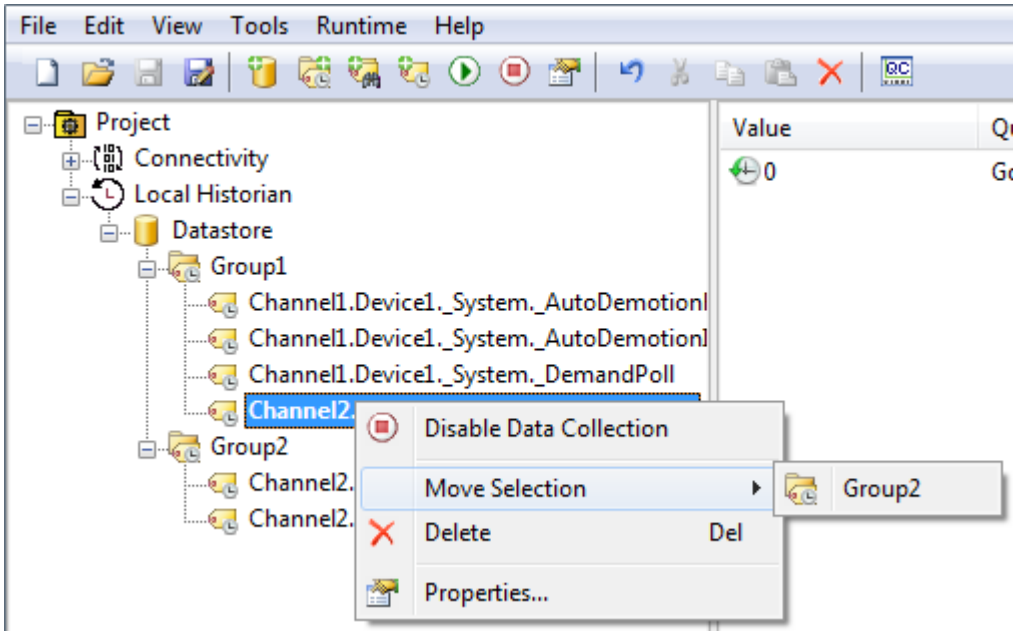
Minimum required free space Specify the megabytes of space required. The range is from 100 MB to 5 GB; default is 500 MB.

Moving and Deleting Tags from the Datastore

Moving Historical Tags to a New Tag Group

The Local Historian supports moving Historical Tags among tag groups. To be eligible for a tag move, the target group cannot be the _UnmappedTags group, _ImportedTags group, or the group in which the tag currently resides.

1. Verify that at least one user-defined tag group exists that is eligible for the tag to be moved.
2. Select the Historical Tag or collection of Historical Tags in the Project View.
3. Right-click then select **Move Selection** or choose **Edit | Local Historian | Move Selection**.
4. Choose the target group from the cascading menu.



See Also:
Datastore and Server Project Synchronization

Deleting Historical Tags

Deleting a Historical Tag removes the item reference from the HDA namespace and the tag definition from the datastore, but it does not remove any persisted VQT data. Tags can be deleted individually, several at a time by multiple selection, or all at once by deleting an entire tag group.

No new data is collected for a deleted tag and no further handles may be acquired for the tag because it no longer exists in the HDA namespace. Existing clients that acquired a handle to the data associated with a tag before it was deleted can still use that handle to access historical data.

Performing an undo after a delete adds the deleted Historical Tag(s) back to the project and re-establishes access to previously collected historical data by restoring the deleted information back to the HDA namespace. Tags restored by undo return to the state prior to deletion: if data collection was enabled, data collection begins again as soon as the tag is restored.

To delete tags:

1. Select the tag(s) or group to be deleted (multi-select is available in the Detail View).
2. Right-click and select **Delete**.

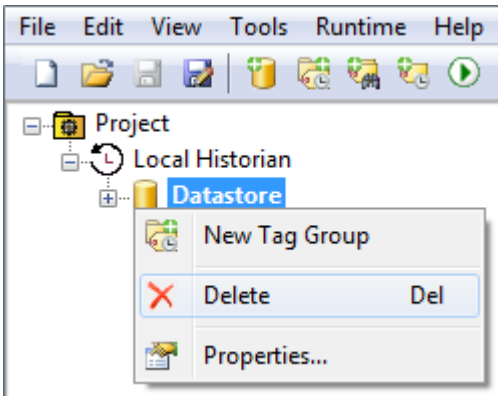
Deleting the Datastore

While it is expected to be a rare occurrence, it is possible to delete the datastore. If a datastore still contains configured historical tags when it is deleted, the delete operation first deletes those historical tags. The delete operation removes the reference to the datastore from the project, but does not delete the historical data files from disk. To delete the datastore, follow these steps:

1. Open the Server Configuration client.
2. Access the Local Historian node in the project tree view.
3. In the left pane/ Project View, select the datastore object. Then choose one of the following:
    - Right-click the selected datastore and select **Delete** from the pop-up menu.
    - Click the Delete button (red 'X') in the tool bar.

- Choose **Edit | Delete**.

4. Click **Yes** to confirm deletion.



Tip: It is possible to “undo” deletion of a datastore.

See Also:

Deleting Tags from the Datastore

Datastore and Server Project Synchronization

The following scenarios result in the datastore and server project becoming desynchronized with respect to the set of historical tag definitions:

- Deleting Historical Tags in Offline Mode
- Adding Historical Tags in Offline Mode
- Deleting and Re-adding a Tag in Offline Mode
- Unmapped Tags

Deleting Historical Tags in Offline Mode

1. A project containing Historical Tags is running.
2. Server Configuration is disconnected from the Runtime, which causes the project to enter offline mode.
3. One or more historical tags are deleted and a copy of the project is saved.
4. Server Configuration is reconnected to the Runtime.
5. The project saved in step 3 is opened and pushed to the Runtime, replacing the default project.
6. The historical tags deleted while offline come back into the project under the datastore in the _ UnmappedTags group.

When the runtime project is updated, the missing historical tags are restored from information contained in the data-store. These restored tags are placed in a special, auto-generated tag group called _UnmappedTags. Tags in this group cannot have new data collected for them. These tags can be deleted or moved into user-defined tag groups. They must be moved to collect new data.

See Also:
Unmapped Tags

Adding Historical Tags in Offline Mode

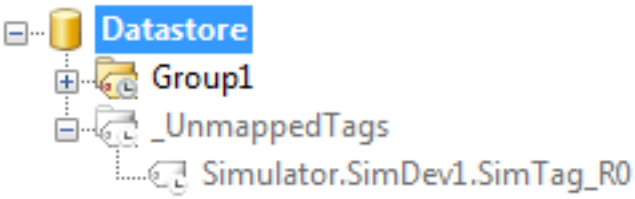
This scenario is similar to previous scenario; a new historical tag is added so the tag exists in the project, but not the datastore. When the Runtime connection is re-established, the historical tag is added to the datastore and data collection starts (if enabled for the tag and parent group).

Deleting and Re-Adding a Tag in Offline Mode

This scenario is a combination of the two described above. When the tag is re-added, it gets a different Tag ID, which is an internally generated 64-bit number used to identify each historical tag. The project is updated with information from the datastore to resynchronize the mapping.

_Unmapped Tags

The _UnmappedTags group is a special, internally generated historical tag group. It is created when there is at least one historical tag definition in the datastore that is not represented by a historical tag in the project. As in the previous scenarios, this is most likely due to offline project edits. Tags that exist in this group automatically have data collection disabled and can only be deleted or moved to another tag group.



In the example shown, the historical tag Simulator.SimDev1.SimTag_R0 was deleted in offline mode and, when the runtime project was updated, the tag still existed in the datastore so was retrieved and placed in the _ UnmappedTags group. The group and tag are shown as gray to indicate data collection is disabled.

For information on moving tags to a user-defined tag group, refer to Moving and Deleting Historical Tags.

Unmapped Tags

The Unmapped Tags group is a special group that contains tags that are in the datastore but not in the project. This can occur when Historical Tags are deleted in offline mode: the tags are added back to the project under this tag group. For more information, refer to Datastore and Server Project Synchronization. Tags can be deleted or moved to another tag group to resume data collection.

- *For information on moving tags to a user-defined tag group, refer to Modifying Tags in the Datastore.*

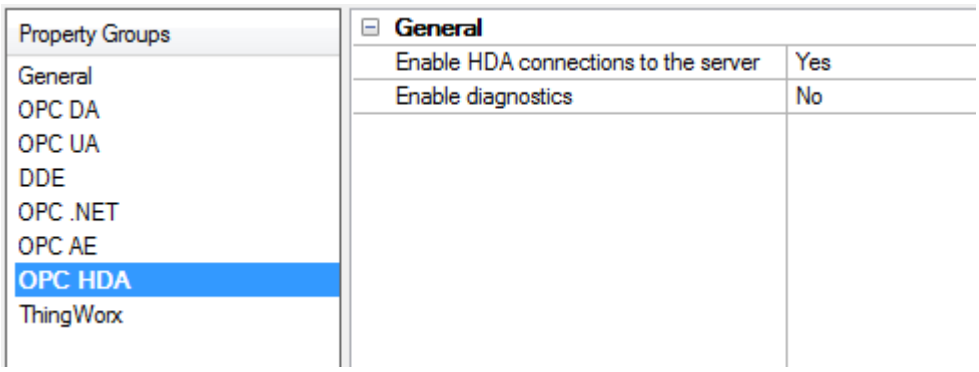
See Also:

Deleting Tags from the Datastore

Project Settings

The Local Historian includes an OPC HDA server interface with a small set of project-level configuration settings. These settings are located in the Project Properties dialog on the OPC HDA tab, along with the OPC DA server settings.

Select **File | Project Properties...** to open the properties dialog:



By default, the server is configured to accept connections from OPC HDA clients. Unless there is an actual datastore configuration present, the client sees a “down” status.

Enable HDA connections to the server
Disabling this option causes the HDA server interface to stop accepting new connections. If there are active connections, the runtime disconnects the clients and disables the interface on reinitialization.

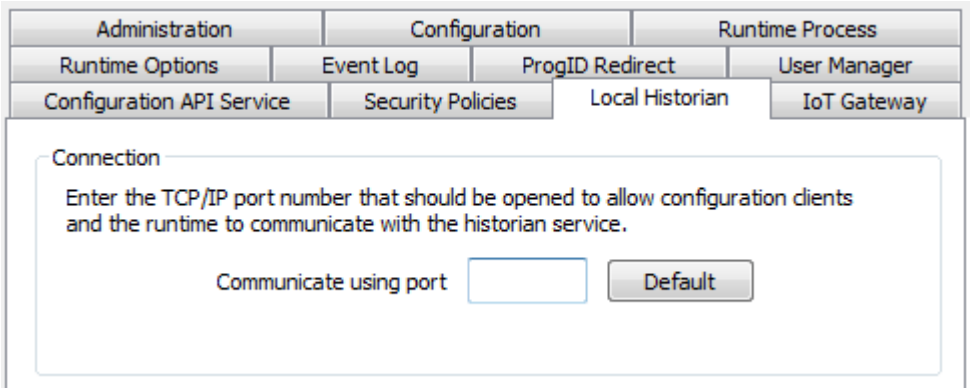
Enable Diagnostics
The HDA server interface supports diagnostic reporting similar to OPC DA. Diagnostics generation is disabled by default. **Enable Diagnostics**, allows information to be reported from the OPC Diagnostics viewer. To open the viewer, select **View | OPC Diagnostics** from the main menu.

See Also:

Troubleshooting

Administrative Settings

The Local Historian administrative settings are automatically configured on installation. If the settings must be updated, access the Local Historian system settings by right-clicking on the Administration icon located in the system tray and selecting **Settings | Local Historian**.



Communicate using port: This parameter specifies the TCP/IP port that the server Runtime and Configuration use to communicate with the Historian Service. The valid range is 1024 to 65535. The default is configured by the server.

The **Default** button populates the field with the default port number.

Tips:

1. The default port is recommended unless there is a conflict with another server application using that port.
2. The Historian Service does not accept remote connections, so there should be no firewall implications associated with this port assignment.

Client Interfaces

The OPC HDA is the Historical Data Access standard created by the OPC Foundation and the server supports the synchronous portion of the OPC HDA 1.20 interface specification. This provides access to raw and interpolated (such as, average) time-series data. Similar to OPC Data Access (DA), OPC HDA relies on Microsoft® COM/DCOM technology to exchange data between the client and server. For information on DCOM configuration, refer to Settings - Runtime Options in the server help file.

Clients attempting to connect to the server should use the following PROGID:
"<vendor>.<server>_HDA.V5"

The Local Historian implements the following interfaces and functions:

- IOPCHDA_Server
  - GetAggregates
  - GetHistorianStatus
  - GetItemAttributes
  - GetItemHandles
  - ReleaseItemHandles
  - ValidateItemIDs
  - CreateBrowse (does not support filtering)
- IOPCHDA_SyncRead
  - ReadAttribute
  - ReadRaw
  - ReadProcessed
  - ReadAtTime
- IOPCHDA_Browser
  - GetEnum
  - ChangeBrowsePosition
  - GetItemID
  - GetBranchPosition

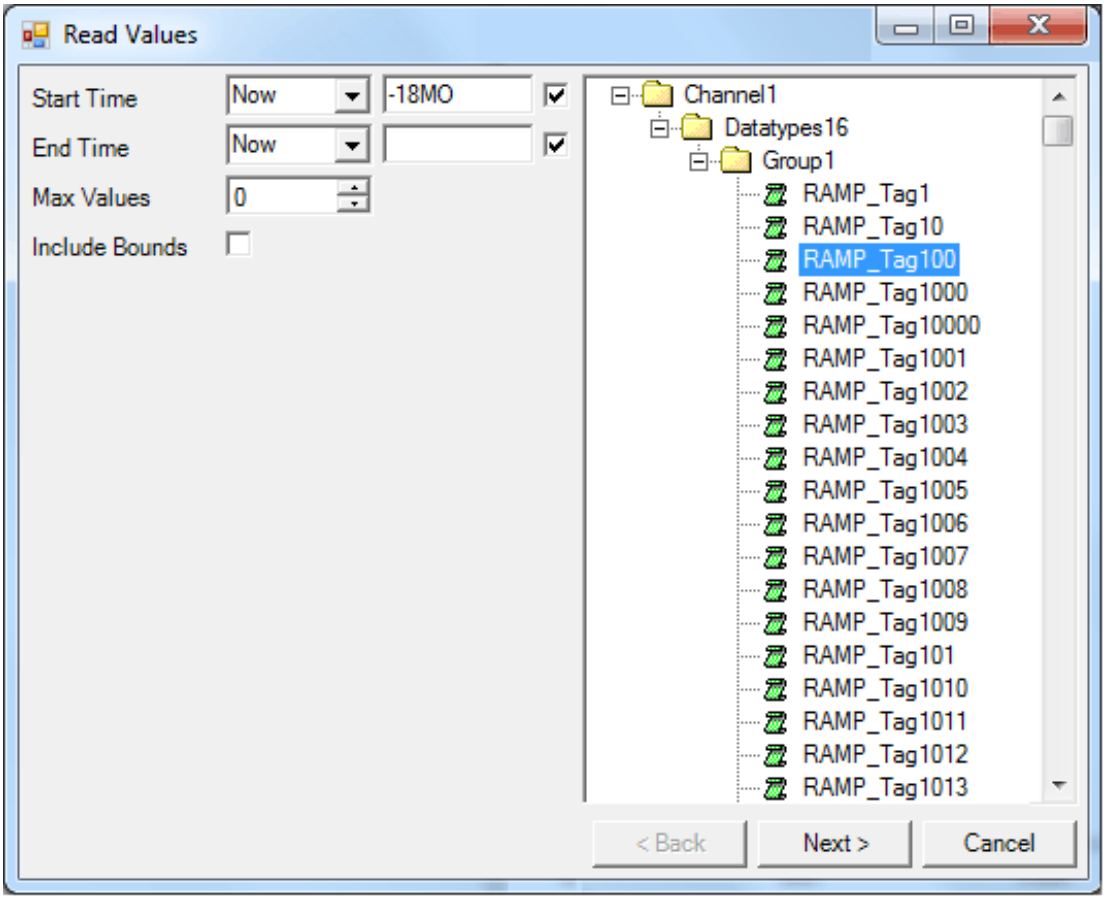
See Also:

IOPCHDA_Server::GetHistorianStatus
IOPCHDA_SyncRead::ReadAttribute
IOPCHDA_SyncRead::ReadAtTime
IOPCHDA_SyncRead::ReadProcessed
OPC HDA 1.20

Browsing

The HDA browse space is represented in a similar manner to the server's OPC DA browse space. Tags are displayed in a hierarchical structure that utilizes the concept of "branches" (the main nodes) and "leaves" (the sub-nodes). When browsing for tags from an HDA client, all available Historical Tags are shown. An example of the HDA browse interface (taken from a sample HDA client) is shown below.

Note: Browse filtering is not supported.



IOPCHDA_Server::GetHistorianStatus

This interface function returns status information about the HDA server.

Parameter Name	Expected Content on Return
pwStatus	One of the OPCHDA_SERVERSTATUS enumeration constants: • **OPCHDA_UP:** This indicates that the OPC HDA server is initialized and ready to be queried for data. It requires that the Historian Service (server_historian.exe) be up and running. • **OPCHDA_DOWN:** This indicates that the HDA Server is in the process of shutting down. • **OPCHDA_INDETERMINATE:** This indicates that the server project has no datastore configuration or the HDA server is shut down.
ppszStatusString	The string equivalent to the enumeration constant returned in pwStatus: • **"Up"** for OPCHDA_UP • **"Down"** for OPCHDA_DOWN • **"No Configuration"** for OPCHDA_INDETERMINATE
ppszVendorInfo	Company name
pwMajorVersion, pwMinorVersion, pwBuildNumber	Product version number components (i.e. 5, 16, 555, respectively).
pftStartTime	FILETIME that specifies the most recent time that the HDA server was initialized.
pftCurrentTime	FILETIME that specifies the current time.
pdwMaxReturnValues	Always zero, indicating no specific limit to the amount of data that may be returned in a single read request. There is, however, a 150-MB limit on the amount of memory allocated to service a read request at the datastore level, which translates to well over 1 million VQTs.

See Also:

Client Interfaces
IOPCHDA_SyncRead::ReadAttribute
OPC HDA 1.20

IOPCHDA_Server::ReadAtTime

This interface function reads attribute information from the specified item. Attribute values are not historized, so only the current values are available for any item.

This interface function retrieves the raw and good quality value found at the timestamp provided in the request. If no such value is available, the HDA server searches for a good quality start and end to perform a linear interpolation (or extrapolate using stepped interpolation when a good quality end bound cannot be found), in accordance with the interpolative aggregate described by IOPCHDA_SyncRead::ReadProcessed.
It is possible to unintentionally provide parameters that cause the ReadAtTime implementation to process for an excessive amount of time. For that reason, the server times out any outstanding ReadAtTime call after roughly five (5) minutes of processing. The entire request is terminated and a result of E_FAIL is returned to the client. ReadAtTime is designed to timeout immediately when the server is reinitialized or shut down.

See Also:

Client Interfaces
IOPCHDA_Server::ReadProcessed
OPC HDA 1.20

IOPCHDA_Server::ReadProcessed

This interface function performs standardized calculations, known as aggregates, on historical data retrieved from the datastore at a user-defined resample rate.
The client provides the following inputs:

- A start and end time for the request. Reverse reads, where the end of the request is earlier than the start of the request, are allowed.
- A resample interval. A unique aggregate value, quality, and timestamp are provided for each interval. The interval is specified in 100 ns ticks.
- An array of server item handles (and the size of that array). Each item in the array is aggregated at the provided resample interval.
- An array of aggregate types, which specifies the calculation to be performed on data retrieved for each item. The size of this array must be equal to the size of the item array.

It is possible to provide an exceptionally large request domain and/or an exceptionally small resample interval. For that reason, the server times out any outstanding ReadProcessed call after roughly five (5) minutes of processing. The entire request is terminated and a result of E_FAIL is returned to the client. ReadProcessed is designed to timeout immediately when the server is reinitialized or shut down.

Aggregate types are specified by DWORD, not by an enumeration. This allows server vendors to specify custom aggregate types. This server does not support any custom aggregate types and set the item ppError to OPC_E_NOT_AVAIL for each unsupported aggregate type specified.

Based on the user-defined inputs, the server retrieves values from the datastore, performs the appropriate aggregate calculation, and returns an aggregate VQT for each item at each resample interval in accordance with the HDA specification.

An error array (ppErrors) is also returned in accordance with the specification. The error array provides an indication of the success or failure of the aggregate calculations for each item. When success is indicated, it is the client's responsibility to free the server allocated memory for the aggregate VQT for each interval in accordance with COM guidelines.

All aggregate return values are of type double (VT_R8). Some precision loss is possible based on the data type of the historical values aggregated. Doubles (VT_R8), 64-bit integer types (VT_I8 and VT_UI8), and numerically convertible strings (VT_BSTR) are especially prone to precision loss.

The server attempts to convert strings to numeric values for each interval in accordance with the OPC HDA specification. If all strings in the interval fail conversion, the quality of the aggregated value is bad, OPCHDA_CONVERSION.

This server explicitly rejects both bad and uncertain qualities from all aggregate calculations. The quality of the aggregate itself is uncertain/subnormal when a non-good quality value is skipped.

Each aggregate has unique processing requirements and output behaviors. These are summarized below.

Interpolative

This aggregate returns the raw and good quality value found at the start time of the interval, if present. If no such value is available, the HDA server searches for a good quality start and end bound in accordance with the HDA specification. In this case, a linear interpolation is performed to deduce the value that would be present at the start of the interval using the following formula:

V = Va + (Vb - Va) * (T - Ta) / (Tb - Ta)

where:

V is the unknown interpolated value.
T is the interpolation time.
Ta is the start bound time.
Tb is the end bound time.
Va is the start bound value.
Vb is the end bound value.

When the end bound of an interpolative aggregate request cannot be found, the server extrapolates the end bound using stepped interpolation. Practically, this means that the start bound value is returned as the interpolated aggregate.

If there is no beginning bound, the aggregate quality is OPCHDA_NODATA. The server does not extrapolate backward in time (for forward or reverse interpolation reads).

Time Average
This aggregate uses the interpolative algorithm to find valid start and end bounds. A line is then plotted between each raw value in the interval. The area under the line is divided by the length of the interval to provide the time-weighted average. Non-good values (bad or uncertain) are not included in the area calculations.

Total
This aggregate is the result of the Time-Weighted Average multiplied by the length of the interval, normalized to seconds. It is the area under the curve formed by the start and end bound of the interval, plus the raw data points in between.

Average
This aggregate performs a running average for each value retrieved in the current interval. A running average is used to limit overflow errors in the returned double precision value. The returned quality is uncertain/subnormal when any non-good values are ignored in the computation.

Minimum Actual Time
This aggregate retrieves the minimum good raw value within the interval and returns the timestamp at which that value occurs. If the same minimum value exists at more than one timestamp, the oldest one is retrieved. This is true for reverse reads as well.

Maximum Actual Time
This aggregate retrieves the maximum good raw value within the interval and returns the timestamp at which that value occurs. If the same maximum value exists at more than one timestamp, the oldest one is retrieved. This is true for reverse reads as well.

See Also:
Client Interfaces
IOPCHDA_Server::ReadAtTime
OPC HDA 1.20

IOPCHDA_SyncRead::ReadAttribute
This interface function reads attribute information from the specified item. There are no historical attribute values, so only the current values are available for any item.

HDA Attribute ID	Returned Value
OPCHDA_DATA_TYPE	This is the data type of the historical item in the Local Historian. The data type is always the canonical data type of the tag in the server.
OPCHDA_DESCRIPTION	User-defined description that may be present for the server item reference identified by the historical tag if it was created from a static or system tag; empty otherwise.
OPCHDA_ARCHIVING	Indicates the status of data collection: true = enabled; False = disabled.
OPCHDA_HIGH_ENTRY_LIMIT	This is the maximum scaled value when scaling is enabled for the item. If scaling is disabled, this is the maximum data type value. Scaling is configurable as a tag property in the server. For more information, refer to **Tag Properties – Scaling** in the server help file.
OPCHDA_LOW_ENTRY_LIMIT	If scaling is enabled for the item, this is the minimum scaled value. If scaling is disabled for the item, this is the minimum data type value.
OPCHDA_EXCEPTION_DEV	This is the deadband of the historical item, reported as a decimal. For example, 25% is reported as 0.25.

See Also:
Client Interfaces
IOPCHDA_Server::GetHistorianStatus

OPC HDA 1.20

OPC HDA 1.20

OPC HDA is the Historical Data Access standard created by the OPC Foundation. This interface provides access to raw and interpolated (such as, average) time series data. Similar to OPC Data Access (DA), OPC HDA relies on Microsoft COM/DCOM technology to exchange data between the client and server.

See Also:

Client Interfaces
OPC HDA Specification

Viewing Archive Data Using Import

The Local Historian supports viewing data collected from another Local Historian instance or archived to long term storage. Archive .tsd files copied or moved into the import directory are detected, read, and attached to make the contained data available to HDA clients.

See Also:

General Operation

Initialization

When an import location is configured, the Historian Service creates the directory (if necessary) and begins monitoring it for the addition or removal of .tsd files. The import location can be a local system drive or a UNC network path.

See Importing from a Network for notes on configuring service permissions.

File Validation

When a new .tsd file is detected in the directory, the Historian Service verifies that it is a correctly formatted file. Next, the service confirms that the file doesn't contain data for a time range that overlaps any file currently in the datastore or any other attached import file. If the file is invalid or specifies an overlapping time range, an event log message is logged *(see Event Log Messages)*.

Namespace Resolution

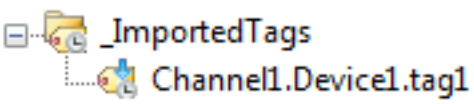
Once an import file is validated, the Historian Service reads the names and metadata for each of the historical tags in the file and provides it to the plug-in for namespace resolution.

The plug-in integrates the set of imported tags into the historical tags currently defined in the project. During this process, each imported tag is identified as a byproduct of one of the following cases:

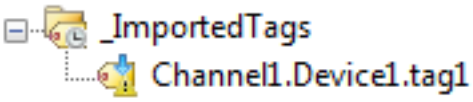
1. Imported tags with an ID matching the ID of a tag currently configured in the project.
2. Imported tags with an ID that does not match any tag in the current project, and with a name that does not match any tag in the project.
3. Imported tags with an ID that does not match any tag in the current project, but with a name that does match a tag in the project.

Tags in the first case are historical tags that existed when the import file was originally backed-up and still exist in the current project. Their data is made available in the user interface and to HDA clients through the existing historical tag that is still in the project. No additional license is consumed by this class of imported tag nor is any change required to the project or HDA namespace.

Tags in the second case are either historical tags that have been deleted from the project since the import file was backed-up or are the result of importing a .tsd file that originates from another datastore. These new imported tags are displayed in the special tag group "_ImportedTags", as they do not correspond to any historical tag that exists in the project. These tags are added to the HDA namespace and their data is available to HDA clients. Because they represent an addition to the HDA namespace, each of these tags consumes a license while they are loaded. They appear as follows when viewed in the user interface:



The third case represents a special subset of tags in the second group -tags not present in the current project, but they cannot be added to the HDA namespace because the name matches that of a historical tag in the project (OPC HDA does not allow for duplicate item references). These "conflicted imports" are also added to the _ImportedTags group with a status of Unavailable. These historical tags are unavailable to HDA clients and do not consume a license. They are shown as follows when viewed in the user interface:



Notes:

1. If the same historical tag (same ID) exists in more than one import file with different names, the name from the most recent import file is used.
2. To make a conflicted import available to HDA clients, delete or rename the historical tag in the project that conflicts with the imported tag. When its name no longer conflicts, it is available as a tag in the second group above.

File Removal

When the Historian Service detects that a .tsd file previously imported has been removed from the import location, the service recalculates the net collection of the remaining imported files (if any) and provides that to the plug-in. The plug-in then repeats the process of namespace resolution based on the remaining files. If imported tags from that file were consuming a license and those tags are not present in any other imported files, those licenses are released.

Note: The Historian Service does not delete any files from the import location.

Deactivation

Import is “turned off” by clearing the Import location path in the datastore properties. If this occurs, the Historian Service detaches any currently attached .tsd files and informs the plug-in of the empty imported historical tag set. All licenses consumed by imported tags are released.

Note: The Historian Service does not delete any files from the import location.

Key Points

- Import files to which the historian attaches are accessed from the import location.
  - Imported data is not copied into the datastore.
- Import files are always read-only.
  - The Local Historian does not modify or delete import files.
  - Data cannot be collected for imported tags.
- Importing tags that aren’t in the project consumes licenses.
  - Disabling import or removing all import files releases licenses.

Attaching Imported Files

A common use of import is to view historical data backed-up from a production machine on another system for off-line analysis or data validation. This does not require the project file from the machine that originally collected the data; it only requires the .tsd files. This data is viewable in the Local Historian and served to HDA clients.

Tip: The data can be viewed and served to HDA clients on an instance of the server running under Time-Limited Mode; no second license is required. Time-Limited Mode timer limitations apply *(see Licensing)*.

To configure a system to import and view (and provide HDA access) to data stored in .tsd files:

1. In Server Configuration, choose **File | New** to create a blank project (you may be prompted to save any existing project files).
2. Access **Local Historian**and select **Click here to create a new datastore…**.
3. In the wizard, accept default values for Archive Location and click **Next >**.
4. On the Import Location page, choose or enter an import directory. This can either be a new directory into which the files are copied or an existing directory that already contains those files. Click **Next >**.
5. Complete the other steps of the wizard, accepting default values.

6. Copy or move the .tsd files to be viewed into the import location.
7. Verify the Local Historian has attached to all valid .tsd files in the import location by selecting the "_ImportedTags" historical tag group. The Detail View should contain all historical tags found in those. Data for these tags is available to HDA clients.

See Also:
Importing from a Network

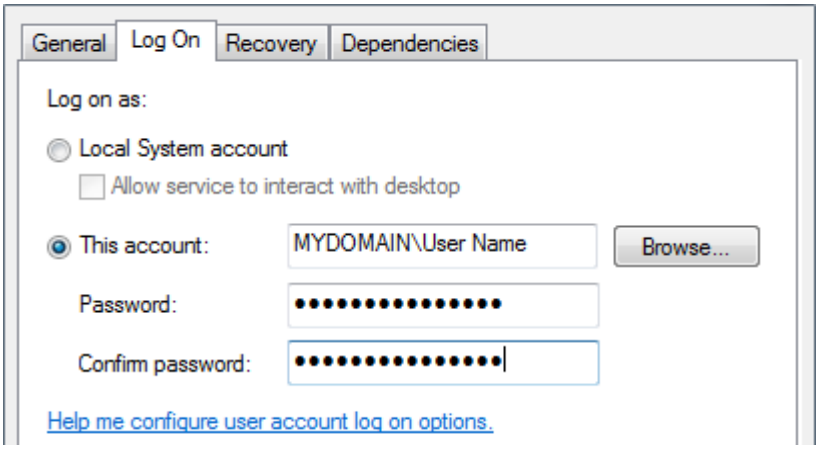
Importing from a Network

If the import location is a UNC network path, the Historian Service must have credentials with permissions to access the directory. By default, the service runs in the SYSTEM account, which does not have any permission outside the local PC.

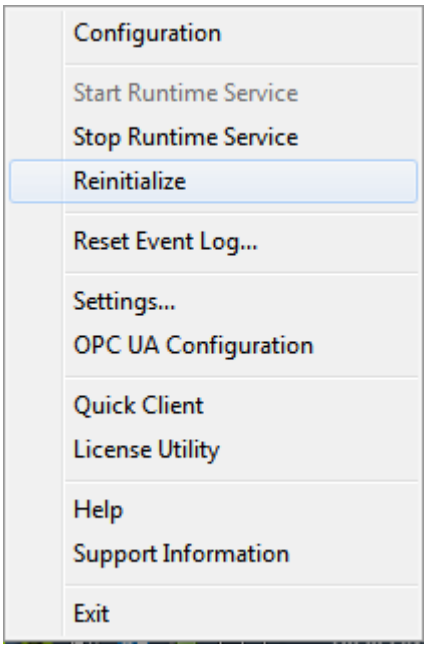
Follow these steps to change the user account the service runs to access the network.

Note: Administrator privilege is required to execute these steps.

1. Start the Windows Services configuration:
   - Locate and launch the Control Panel.
   - From the Control Panel, select **Administrative Tools**, then **Services**.
2. Within Services, find and right-click the Historian Service, and select **Properties**.
3. Navigate to the **Log On** tab.



4. Select **This account:** and enter the name and password of the account whose credentials the Historian Service should use to access the import directory.
5. Click **OK** to accept the changes.
6. If the Historian Service is running, it must be restarted to apply the changes. Reinitialize the Server Runtime by right-clicking on the Server Administration taskbar icon and selecting **Reinitialize**.



7. Observe the Event Log for a message confirming that the Local Historian is able to monitor the import location.

See Also:

Viewing Archive Data Using Import

Backing up Archive Files

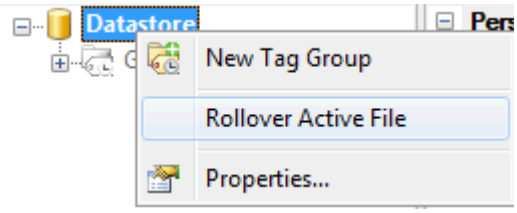
This section discusses aspects of Local Historian operation related to backup of the files that it creates. Backup in this context refers to any action that attempts to copy a file from the archive location.

Runtime Restrictions

While the Historian Service is running, the only files that may be removed or copied from the datastore are those with a .tsd extension. These have been rolled over with name information appended so they can be imported or returned to the datastore at a later time. If desired, the .active file can be rolled over manually to make it eligible for backup.

Manual Rollover

Manual rollover is initiated from the main menu under Edit | Local Historian | Rollover Active File or by right-clicking on the Datastore in the tree:



Note: It takes approximately 60 seconds after rollover to convert an .active file into a .tsd file.

Restoring Files to the Datastore

Files can be restored to the datastore location as part of a recovery operation, but is not recommended as a means to re-gain access to old data. If data access is the primary reason for the restore, use the Import feature instead. To bring a .tsd file back into the datastore, the Server Runtime must be re-initialized after copying the file for the Historian Service to re-attach it to the archive.

Coordinating with Automatic Backup Software

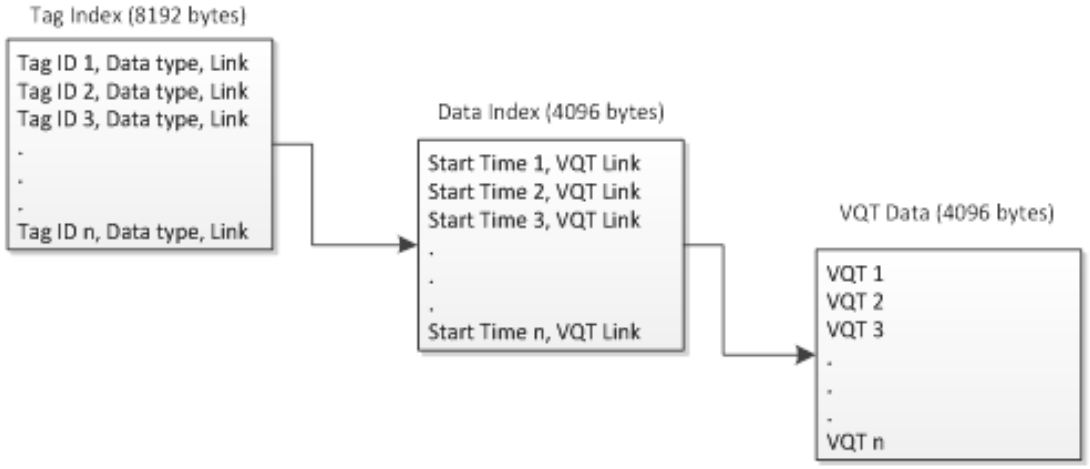
If retention policy enforcement is enabled, the oldest .tsd file is automatically deleted when the total number of files exceeds the configured limit. Keep the backup ahead of this by including a buffer in the configured limit to allow the backup software to copy them. The buffer is application dependent and should be determined by observation and tuning the file count and rollover settings appropriately.
If retention policy enforcement is disabled, no files are automatically be deleted from the datastore. A system backup can be run at any time and capture all historical data up to, but not including, the start time of the .active file. A third-party mechanism can be used to manage the file set within the archive location to remove .tsd deemed old enough to be moved out of daily use.

Backing up the .name File

To completely recover a datastore from backup , the .name file must be restored so new data can be collected. To back up this file, the Server Runtime must be stopped to remove the sharing lock placed on it by the Historian Service.

Estimating the Datastore Size

A .tsd file consists of an internally indexed set of fixed-size data blocks. A two-level index is employed: tag index to data index to VQT data. The blocks are broken into individual fixed-size entries. To get to the data for any historical tag, the ID is used to look up the link (or links) to the data index corresponding to the historical tag. From the data index, the time range is used to lookup the VQT data block(s) containing the data.



Tag Index

Each tag index block is 8192 bytes in length and contains entries defined as follows:

Entry	Description	Size (Bytes)
Tag ID	Unique identifier of the historical tag.	8
Data Type	VARTYPE enumeration corresponding the to the data type assigned to the historical tag (see MS Knowledge base).	2
Link	Offset within the file where the tag data index is stored.	4

Each unique historical tag in the project consumes a single entry in the tag index. At 14 bytes per entry with an 8 K block size, this allows for 585 entries per block. For a 10000-tag project, 18 tag index blocks are needed, for a total of 147,456 bytes.

Data Index

Each tag index block is 8192 bytes in length and contains entries defined as follows:

Entry	Description	Size (Bytes)
Start Time	Time associated with the first VQT in the VQT data block indexed by the entry.	8
Link	Offset within the file to the start of a VQT data block.	4

The data index gets a new entry whenever the time range of data associated with an entry goes beyond 36 hours or the VQT block is fulland a new one is needed. For slow data collection rates and a rollover time of less than three days, there should only be one data index block and one VQT data block per tag.

VQT Data

Each VQT data block is 4096 bytes in length and contains entries defined as follows:

Entry	Description	Size (Bytes)
VQT	VQT information stored in packed format.	4 + size of (Data Type)
Data Type	VARTYPE enumeration corresponding the to the data type assigned to the historical tag (see MS Knowledge base).	2
Link	Offset within the file where the tag data index is stored.	4

A new entry is added to a VQT data block each time a VQT is stored. The capacity of the block depends on both data type and time range. The compacted format allows for, at most, 36 hours of data to be stored in one block. The following table shows how many entries can be stored based on the type and assuming no time range restriction.

Data Type	VARTYPE	Entry Size (Bytes)	Max. Entries
Boolean	VT_BOOL	6	682
Byte	VT_UI1	5	819
Char	VT_I1	5	819
Word	VT_UI2	6	682
Short	VT_I2	6	682
DWord	VT_UI4	8	512
Long	VT_I4	8	512
Float	VT_R4	8	512
Double	VT_R8	10	409
String(1)	VT_BSTR	8	512

Note: For strings, the VQT is 8 bytes and the data itself is appended to the end of the file in non-blocked fashion. Stings are stored with either UTF-8 or UTF-16 encoding, whichever consumes less storage space. The space consumed on disk is 4 bytes + length multiplied by the size of encoding, where UTF-16 encoding uses 2 bytes per character and UTF-8 uses 1 byte per character.

Estimating the overall datastore size encompasses tag count, data types, data change rate of each tag (a new entry is recorded only if a data change occurs), rollover rate, and retention policy.

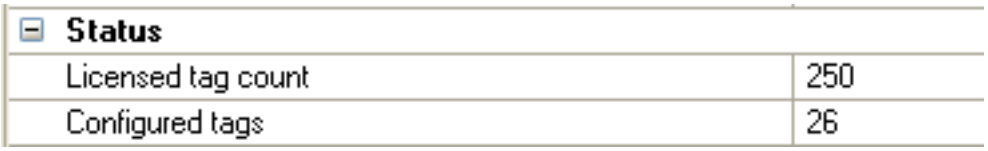
To reserve adequate disk space for the application, first estimate the overall file size that needed to store the amount of data for some unit of time that matches a reporting interval for the application. If daily data is collected and reported, try rollover once a day. If the amount of data collected each day is roughly the same, multiply file count by file size to calculate the necessary space. The file count should consider the retention policy, which can be used to limit the total file count by deleting the oldest file when the count reaches the configured limit. This assumes a backup strategy moves .tsd files out of the active archive at some interval and stores them off the local PC.

This information is provided as a guide. In practice, observations of the system will yield more accurate results.

Licensing

The Local Historian uses a tiered, count-based licensing model. A license can be purchased that enables the product to run for an unlimited amount time for a fixed maximum number of historical tags. The license limit does not prevent the addition of new tags beyond the tag count, nor does it signal the product to enter Time-Limited Mode, but it does prevent data collection and HDA data access for any tags added beyond that count.

The licensed and configured tags counts are reported in the datastore Detail View in the Status section, as shown in the below example with a 250-tag license installed and 26 historical tags in the project:



Exceeding the Limit

An event log message is posted each time a new tag is created in excess of the license limit. Those messages are the only way to identify those tags. If the license limit is exceeded, existing tags can be deleted to make license counts available for the new ones. When the historical tag count is back to the license limit, an event log message is posted to indicate the product is within licensed limits.

- When the license limit is exceeded, Local Historian processes the licensed number of tags only; the exact tags can vary based on the project loading order. The Event Log messages indicate the exact tags omitted.

Unlicensed Operation

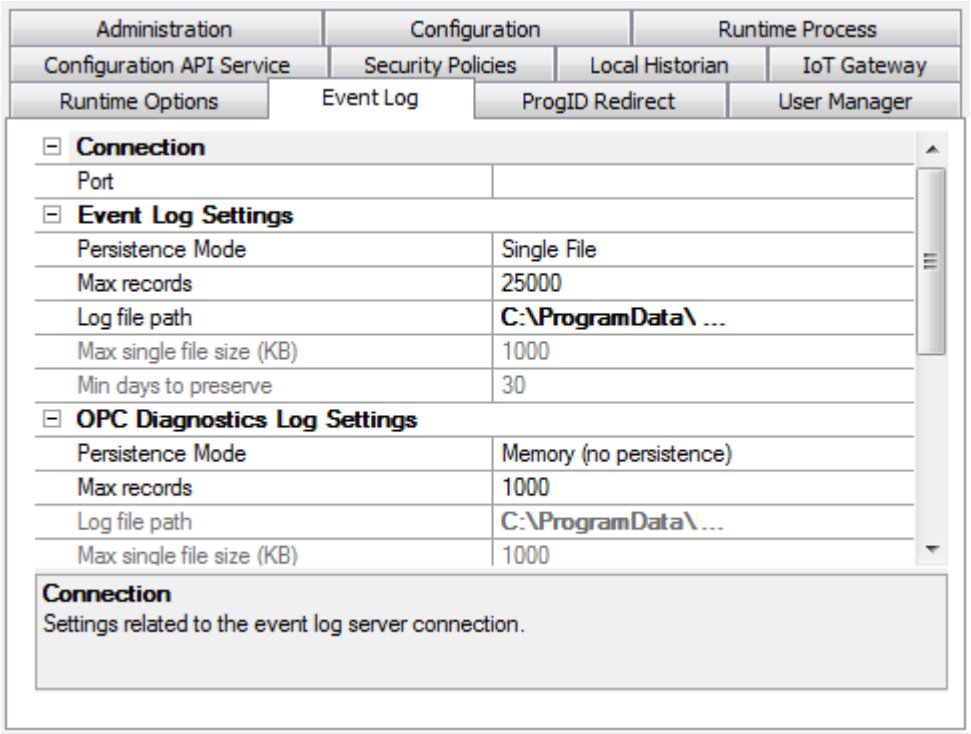
When no license is installed, the product enters Time-Limited Mode, which allows a time-based 10,000-tag trial that stops working when the Time-Limited Mode expires.

Troubleshooting

OPC Diagnostics

The HDA server component of the Local Historian produces diagnostic information when enabled in the project-level settings. This option should only be enabled for troubleshooting a problem as it can generate disk I/O that competes with the Historian Service when the event log is configured to store data using the same disk as the archive location.

OPC diagnostic logging is configured through the server Administration menu. . Right-click the Administration icon in the task bar, then select Settings and choose the Event Log tab.

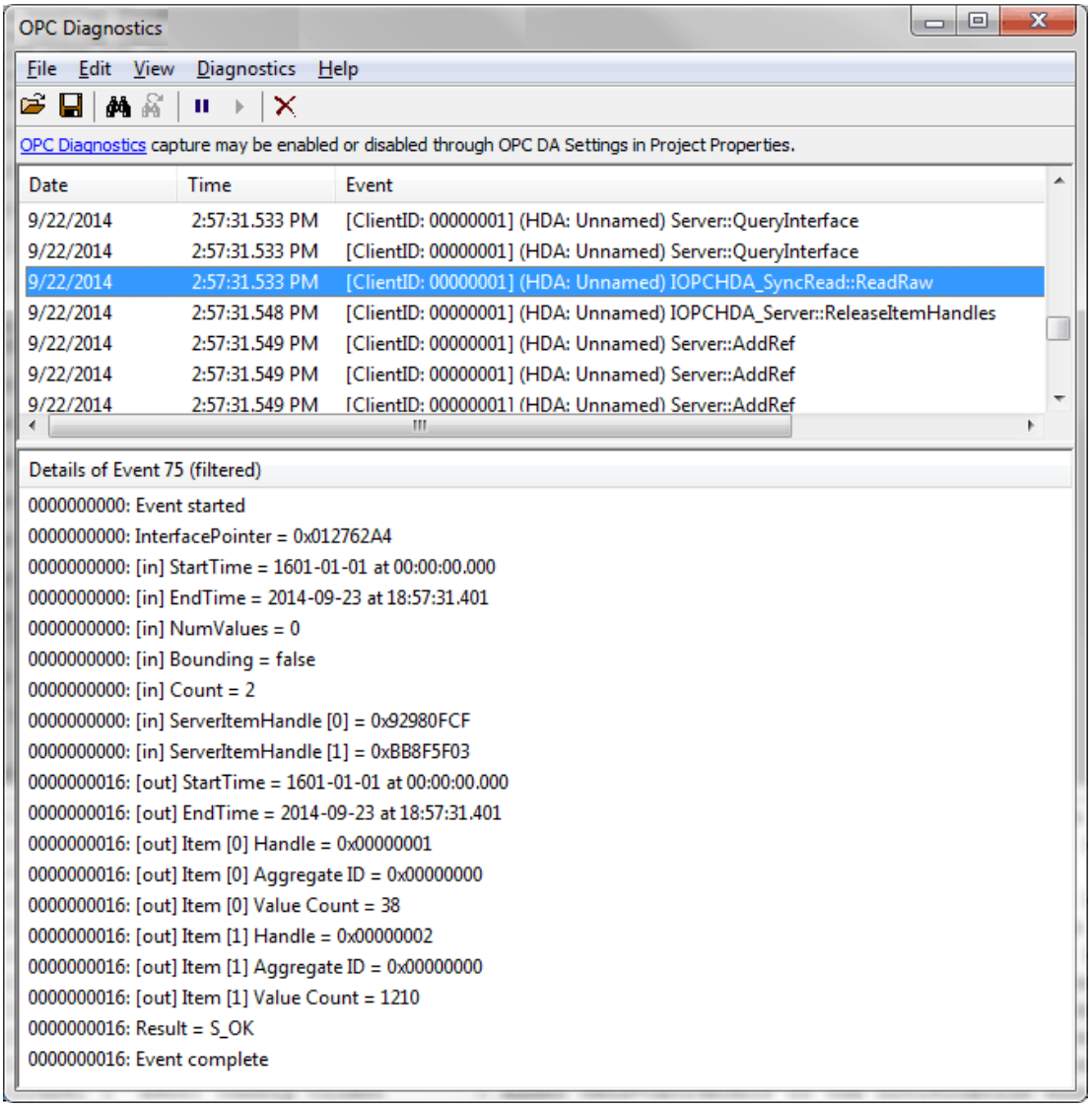


To avoid disk contention between the Historian Service and the Event Log Service, the OPC diagnostics log settings should be configured with Persistence Mode set to “Memory (No persistence)” or, if disk persistence is necessary, change the Log file path to a drive other than the archive location.

To see this information, select **View | OPC Diagnostics** from the server Configuration main menu.

Tip: The OPC Diagnostics viewer can be used to display diagnostics information for services other than HDA. If only HDA diagnostics information is required, it may be helpful to turn off diagnostics for other services. An example of this is OPC DA diagnostics. The OPC diagnostics can be disabled by clearing the check box labeled “Enable diagnostics capture” under **File | Project Properties | OPC DA Settings.**

Once enabled, diagnostics information is recorded for all supported interface functions. Output is function-specific and client-specific; each line of output refers to a function, the server-assigned client ID, the client-assigned client name that called the function, and the client-specific handle(s) associated with the historical tag(s) referenced in the function call. The diagnostics provided for the IOPCHDA_Server::GetItemHandles function can be used to determine which Historical Tags are being referenced when the client calls subsequent functions. The following image depicts the OPC Diagnostics window with a “ReadRaw” event selected:



The lower half of the Detail View presents information about the input parameters the client passed to the server and data the server returned. All times presented use the UTC basis. VQT data is not included by default for performance reasons. To see VQT information, create the DWORD value “Verbose Diagnostics Enabled” under the registry key HKLM/[Wow6432/]<company>/<product>/V<version>/Historian/HDA and set the value to 1.

There is no custom filtering support provided for HDA-specific output. If diagnostics are enabled, all HDA output is shown.

Data Loss

While rare under normal circumstances, errors reported in the event log can indicate dropped data. This can be due to one of the following conditions:

- Continuous incoming data rate is too fast (>100,000 VQTs per second).
- Target disk is too slow or is competing with other applications for access time.

There is maximum incoming buffer capacity of 2,097,152 VQTs in the Historian Service available across all historical tags. This buffer is flushed on a per-tag basis whenever there is data present that is >10 seconds old or the tag has accumulated 10 VQTs since the most recent flush. It is recommended that data be scanned no faster than is actually necessary to avoid flooding the Historian Service with data.

Note: The flush operation must be able to write data to the .active file in a timely manner to keep up with the incoming data. If the target disk hosting the datastore is a slow USB drive or is hosting the operating system, this

can severely limit the rate at which data can be written to the .active file. Slow or heavily utilized drives should not be used with a large number of historical tags collecting at a high scan rate.

Interacting with Anti-Virus Software

Anti-Virus (A/V) software consumes disk I/O bandwidth. Under some circumstances, interactions with A/V software can result in performance problems caused by contention for the disk. When collecting and storing historical data on a system with A/V software installed, follow these recommendations:

- Use a dedicated drive for the archive location and exclude that drive from A/V protection.
- If a dedicated drive cannot be used, configure the A/V software to exclude the archive location.

System Tags

The Local Historian exposes a limited subset of its status information though datastore-level system tags. These are addressable as _LocalHistorian.Datastore.<tag name>. The following table describes the system tags that are available:

Name	Date Type	Client Access	Description
_BytesInPerSecond	Float	Read Only	Data input rate to the Historian Service calculated as an average over the most recent 10-second interval. The same value is displayed in the Detail View, Status, “Collection rate (10s average).”
_SizeInMB	Float	Read Only	Total size of the datastore archive on disk in MB (1024 x 1024 bytes). The same value is displayed in the Detail View, Status, “Size on disk.”

Event Log Messages

The following messages may be generated. Click on the link for a description of the message.

Attached import file <filename>.
Collection for all items has resumed. Adequate free disk space is available.
Collection for all items has resumed. No free space limit is being enforced.
Collection for all items has stopped due to low disk space.
Configured historical tag count is now within the count specified by the installed license.
Configured historical tag count is now within the count allowed in Time-Limited Mode.
CSV import failed: invalid or missing CSV file header.
CSV import did not load any valid tags.
CSV import of item <n> failed: <item reference> is not a valid static or dynamic item reference.
CSV import for item <n> failed: an item reference is required.
CSV import ignoring duplicate item <item reference>: the item was specified more than once.
Data for historical tag <item reference> will not be persisted nor available to HDA clients because the Time-Limited Mode has been exceeded.
Data for historical tag <item reference> will not be persisted nor available to HDA clients because the licensed count has been exceeded.
Datastore creation failed in <location>, changing location to <location>.
Detached import file <filename>.
Disk access restored, collection for all items has resumed.
Failed to attach import file <filename> due to time span overlap.
Failed to attach import file <filename>. The file is invalid.
Failed to import item <item reference>: the item is already defined in group <group name>.
Failed to load datastore file <filename>. The file is invalid.
Failed to store <n> value(s) due to incoming buffer overflow.
File I/O error on <filename>, stopping data collection.
File <filename> was removed from the datastore.
Historian cannot monitor import directory <filename>.
Historian monitoring import directory <location>.
Historian Service starting
Historian Service stopping
Historical tag mapped to <item reference> is already defined.
Historical tag <item reference> was modified to synchronize with persisted data.
Import of item <n> failed: invalid deadband specification.
Missing historical tag <item reference> was generated from persisted information provided by the datastore.
Rejected request to store <n> value(s) due to out of order timestamp.
Rejecting request to roll over the active file: a rollover occurred within the last <n> seconds.
Rejecting request to roll over the active file: the active file contains no values.
Retention policy enforcement removing oldest file, which contains data from <time start> UTC to <time end> UTC.
Rolling over the active datastore file.
Update of item <n> failed: invalid deadband specification.

Attached import file <filename>.

Error Type:

Information

Possible Cause:

A file copied or moved to the import location was successfully processed and made available for HDA access.

Solution:
N/A

See Also:
Troubleshooting

Collection for all items has resumed. Adequate free disk space is available.

Error Type:
Information

Possible Cause:
Available disk space has returned to a level above the configured limit as defined in the datastore properties.

Solution:
N/A

See Also:
Troubleshooting

Collection for all items has resumed. No free space limit is being enforced.

Error Type:
Information

Possible Cause:
The configured disk space reserve has been disabled in the datastore properties after free space has fallen below and currently is below the previously configured limit.

Solution:
N/A

See Also:
Troubleshooting

Collection for all items has stopped due to low disk space.

Error Type:
Error

Possible Cause:
Available disk space is below the limit specified in the datastore configuration settings.

Solution:
Free up disk space by removing archive files.
Decrease the minimum free space setting.

See Also:
Troubleshooting

Configured historical tag count is now within the count allowed in Time-Limited Mode.

Error Type:

Information

Possible Cause:
A historical tag was deleted that caused the count to be at or below the Time-Limited Mode limit.

Solution:
N/A

See Also:
Troubleshooting

Configured historical tag count is now within the count specified by the installed license.

Error Type:
Information

Possible Cause:
A historical tag was deleted that caused the count to be at or below the licensed limit.

Solution:
N/A

See Also:
Troubleshooting

CSV import ignoring duplicate item <item reference>: the item was specified more than once.

Error Type:
Error

Possible Cause:
The CSV file specified the same item reference more than once.

Solution:
Correct the CSV file so that each item reference is unique.

See Also:
CSV Import / Export

CSV import failed: invalid or missing CSV file header.

Error Type:
Error

Possible Cause:
The CSV file specified contains an invalid header row.

Solution:
Specify a CSV file with a valid header or correct header row in the file being imported.

See Also:
CSV Import / Export

CSV import of item <n> failed: <item reference> is not a valid static or dynamic item reference.

Error Type:
Error

Possible Cause:
The CSV file specified an invalid item reference.

Solution:
Add a static tag to the project that matches the expected item reference name or select a device for which the specified reference is valid.

See Also:
CSV Import / Export

CSV import for item <n> failed: an item reference is required.

Error Type:
Error

Possible Cause:
The CSV file specified an item with no item reference.

Solution:
Add an item reference to the historical tag definition in the CSV file.

See Also:
CSV Import / Export

CSV import did not load any valid tags.

Error Type:
Error

Possible Cause:
The CSV file specified contains no valid historical tag definitions.

Solution:
Specify a CSV file with at least one valid historical tag definition or add one to the file being imported.

See Also:
CSV Import / Export

Data for historical tag <item reference> will not be persisted nor available to HDA clients because the Time-Limited Mode has been exceeded.

Error Type:
Warning

Possible Cause:
A historical tag was added beyond the unlicensed 10,000 tag limit available in Time-Limited Mode.

Solution:
Turn off Import or remove files containing imported tags.

See Also:
Troubleshooting

Data for historical tag <item reference> will not be persisted nor available to HDA clients because the licensed count has been exceeded.

Error Type:
Warning

Possible Cause:
A historical tag was added beyond the installed license tag limit.

Solution:
Turn off Import or remove files containing imported tags. Delete historical tags from the project.

See Also:
Troubleshooting

Datastore creation failed in <location>, changing location to <location>.

Error Type:
Warning

Possible Cause:
At startup or re-initialization, access to or creation of the configured archive location reported an error. The default archive location may have been overridden in the datastore settings.

Solution:
Verify that the archive location is valid and accessible, then load the project file again.

See Also:
Troubleshooting

Detached import file <filename>.

Error Type:
Information

Possible Cause:
A file in the import location was moved or deleted.

Solution:
N/A

See Also:
Troubleshooting

Disk access restored, collection for all items has resumed.

Error Type:
Information

Possible Cause:
A file system error that caused data collection to stop, such as media removal, has been resolved.

Solution:
N/A

See Also:
Troubleshooting

Failed to attach import file <filename> due to time span overlap.

Error Type:
Warning

Possible Cause:
An attempt was made to import a file that overlaps in time with another file. The file was not originally part of the active datastore.

Solution:
Remove any conflicting import or .tsd files to resolve the time overlap.

See Also:
Troubleshooting

Failed to attach import file <filename>. The file is invalid.

Error Type:
Warning

Possible Cause:
A file in the import location is corrupt or was not created by the Local Historian.

Solution:
N/A

See Also:
Troubleshooting

Failed to import item <item reference>: the item is already defined in group <group name>.

Error Type:
Error

Possible Cause:
The CSV file specified an item reference already defined by another historical tag group.

Solution:
Identify the tag group where the item reference should be associated and either update the existing historical tag manually or delete it from the group so that it can be imported via CSV.

See Also:
CSV Import / Export

Failed to load datastore file <filename>. The file is invalid.

Error Type:

Warning

Possible Cause:
A file in the archive location is corrupt or was not created by the Local Historian.

Solution:
N/A

See Also:
Troubleshooting

Failed to store <n> value(s) due to incoming buffer overflow.

Error Type:
Error

Possible Cause:
A data loss resulted from failure to flush all buffered data to disk.

Solution:
1. Collect data for fewer historical tags.
2. Configure tags with a slower scan rate.
3. Disable other applications competing for the disk.

See Also:
Troubleshooting

File I/O error on <filename>, stopping data collection.

Error Type:
Error

Possible Cause:
A file system error occurred attempting to read from or write to the indicated file. The disk may be full (in the case of a write) or access lost to removable media.

Solution:
Ensure the disk has free space and is available.

See Also:
Troubleshooting

File <filename> was removed from the datastore.

Error Type:
Information

Possible Cause:
An archive file with a .tsd extension was deleted or moved out of the archive location after it was loaded at startup or created and rolled over.

Solution:
N/A

See Also:
Troubleshooting

Historian cannot monitor import directory <filename>.

Error Type:
Warning

Possible Cause:
The specified import location no longer exists or cannot be accessed.

Solution:
Verify that the configured import location is accessible. If necessary, configure a new location.

See Also:
Troubleshooting

Historian monitoring import directory <location>.

Error Type:
Information

Possible Cause:
The configured import location is ready to receive files.

Solution:
N/A

See Also:
Troubleshooting

Historian Service starting.

Error Type:
Information

Possible Cause:
The Historian Service was started by the Server Runtime or Windows Service Control Manager.
This typically occurs when:

1. A new datastore is created.
2. The Server Runtime is restarted or re-initialized.

Solution:

See Also:
Troubleshooting

Historian Service stopping.

Error Type:
Information

Possible Cause:
The Historian Service was stopped by the Server Runtime or Windows Service Control Manager.
This typically occurs when:

1. The datastore configuration is deleted from the project.
2. The Server Runtime is restarted or stopped.

Solution:

See Also:
Troubleshooting

A historical tag mapped to <item reference> is already defined.

Error Type:
Warning

Possible Cause:
An attempt was made to create a second historical tag with the same server item reference as one that already exists.

Solution:
N/A

See Also:
Troubleshooting

Historical tag <item reference> was modified to synchronize with persisted data.

Error Type:
Warning

Possible Cause:
The definition of the item reference has changed in either data type or reference type since being added to the data-store. This is typically reported when a new project is loaded after being edited offline.

Solution:
N/A

See Also:
Troubleshooting

Import of item <n> failed: invalid deadband specification.

Error Type:
Error

Possible Cause:
The CSV file specified an item with a deadband specified at less than 0.0 or greater than 1.0.

Solution:
Correct the deadband value specified by the CSV file to be within the required range.

See Also:
CSV Import / Export

Missing historical tag <item reference> was generated from persisted information provided by the datastore.

Error Type:
Warning

Possible Cause:
A historical tag was deleted from the project while offline and the Server Runtime was updated with this modification by re-connecting.

Solution:
Find the tag in the _UnmappedTags tag group and delete it or move it to an existing tag group for continued use.

See Also:
Troubleshooting

Rejected request to store <n> value(s) due to out of order timestamp.

Error Type:
Error

Possible Cause:
Data received by the Local Historian could not be persisted to the datastore because the timestamp on the data is older than data already stored for one or more historical tags.

Solution:
Verify that the system clock is functioning properly.
Verify that no time-sync software is configured to jump the system clock backwards in time.

See Also:
Troubleshooting

Rejecting request to roll over the active file: a rollover occurred within the last <n> seconds.

Error Type:
Information

Possible Cause:
A manual rollover was received, but cannot be processed because it occurred too soon after a recently processed request.

Solution:
Try again once the specified time has elapsed.

See Also:
Troubleshooting

Rejecting request to roll over the active file; the active file contains no values.

Error Type:
Information

Possible Cause:
A manual rollover request was received, but cannot be processed because no data is stored in the active file.

Solution:
Add historical items or enable collection for existing historical items before trying a manual rollover.

See Also:
Troubleshooting

Retention policy enforcement removing oldest file, which contains data from <time start> UTC to <time end> UTC.

Error Type:
Information

Possible Cause:
Retention policy enforcement is enabled for the datastore and the number of files has exceeded the configured limit.

Solution:
Reduce the number of files or increase the limit.

See Also:
Troubleshooting

Rolling over the active datastore file.

Error Type:
Information

Possible Cause:
A new archive file has been created because: the active file has been filled to a configured limit the user requested a new file from the user interface file-system error condition has been resolved.

Solution:
N/A

See Also:
Troubleshooting

Update of item <n> failed: invalid deadband specification.

Error Type:
Error

Possible Cause:
The CSV file specified an item with a deadband specified at less than 0.0 or greater than 1.0.

Solution:
Correct the deadband value specified by the CSV file to be within the required range.

See Also:
CSV Import / Export

Index

A

A historical tag mapped to item reference is already defined. 59
Administrative Settings 33
aggregates 37
Applications 5
Architectural Summary 6
Attached import file. 51
Average 38

B

Backing up Archive Files 44
Browsing 35

C

Changing the Datastore 26
Client Interfaces 34
Collection for all items has resumed. Adequate free disk space is available. 52
Collection for all items has resumed. No free space limit is being enforced. 52
Collection for all items has stopped due to low disk space. 52
Configured historical tag count is now within the count allowed in Time-Limited Mode. 52
Configured historical tag count is now within the count specified by the installed license. 53
Configuring a New Tag Group 15
Creating a Datastore 10
CSV File Import / Export 21
CSV import did not load any valid tags. 54
CSV import failed - invalid or missing CSV file header. 53
CSV import for item <n> failed - an item reference is required. 54
CSV import ignoring duplicate item <item reference> - the item was specified more than once. 53
CSV import of item <n> failed - <item reference> is not a valid static or dynamic item reference. 54

D

Data 8
Data for historical tag will not be persisted nor available to HDA clients because the licensed count has been exceeded. 55
Data for historical tag will not be persisted nor available to HDA clients because the Time-Limited Mode has been exceeded. 54
Data Retrieval 8

Datastore and Server Project Synchronization 30
Datastore creation failed, changing location. 55
Datastore View 14
Defining Historical Tags 15
Deleting the Datastore 28
Detached import file. 55
Disk access restored, collection for all items has resumed. 55

E

Estimating the Datastore Size 45
Event Log Messages 51

F

Failed to attach import file due to time span overlap. 56
Failed to attach import file. The file is invalid. 56
Failed to import item <item reference> - the item is already defined in group <group name>. 56
Failed to load datastore file. The file is invalid. 56
Failed to store n value(s) due to incoming buffer overflow. 57
File I/O error on filename, stopping data collection. 57
File was removed from the datastore. 57

G

General Operation 7

H

Help Contents 4
Historian cannot monitor import directory. 58
Historian monitoring import directory. 58
Historian Service starting. 58
Historian Service stopping. 58
Historical Tag Properties 18
Historical tag was modified to synchronize with persisted data. 59

I

Import 8
Import of item <n> failed - invalid deadband specification. 59
Importing from a Network 42

Initialization 7
Interpolative 37
IOPCHDA_Server_GetHistorianStatus 36
IOPCHDA_Server_ReadAtTime 36
IOPCHDA_Server_ReadProcessed 37
IOPCHDA_SyncRead_ReadAttribute 38

L

Licensing 47

M

Maximum Actual Time 38
Minimum Actual Time 38
Missing historical tag was generated from persisted information provided by the datastore. 60
Moving and Deleting Tags from the Datastore 27

O

OPC HDA 1.20 39
Overview 4-5

P

Project Settings 32

R

Recommended System Configuration 9
Rejected request to store value(s) due to out of order timestamp. 60
Rejecting request to roll over the active file- the active file contains no values. 60
Rejecting request to roll over the active file - a rollover occurred within the last few seconds. 60
Retention policy enforcement removing oldest file, which contains data from time start UTC to time end UTC. 61
Rolling over the active datastore file. 61

S

Shutdown 9
Startup 7
Supported Data Types 21

System Tags 50

T

Time Average 38
Total 38
Troubleshooting 48

U

Unmapped Tags 31
Update of item <n> failed - invalid deadband specification. 61

V

Viewing Archive Data Using Import 40
Viewing Historical Data 23