DIGITAL

PROFICY HISTORIAN FOR CLOUD - AZURE

User Guide

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:
doc@ge.com

Cloud Historian

Contents

Chapter 1. Release Notes

What's New in Proficy Historian for Azure Cloud 2023

Proficy Historian is a best-in-class historian software solution that collects industrial time series data. It is secure, fast, and highly efficient.

Proficy Historian for Cloud allows you to deploy the Historian server and its components on cloud destinations. It supports all the components/tools in the on-premises counterpart of Proficy Historian (such as collectors, Excel Addin). And it offers all the advantages of cloud technologies.

Proficy Historian for Cloud offers plenty of new features and enhancements to help you connect your data with other analysis tools, migrate your data from Proficy Historian on-premises, scale up Data Archiver, and many more:

- **Azure Marketplace:** Proficy Historian for Cloud is now available in Azure marketplace.
- **Integration with Workbook:** Using Workbook, you can access logs, which help in troubleshooting issues.
- **File Storage in Azure:** Files are stored in Azure File Share, thus achieving scalability and security. It also offers backup and restore options.
    - **Locally redundant storage (LRS)** copies your data synchronously three times within a single physical location in the primary region. LRS is the least expensive replication option but isn't recommended for applications requiring high availability or durability.
    - **Zone-redundant storage (ZRS)** copies your data synchronously across three Azure availability zones in the primary region. For applications requiring high availability, Microsoft recommends using ZRS in the primary region, and replicating to a secondary region.
- **Deployment in Azure Kubernetes Cluster (AKS):** Proficy Historian for Azure Cloud is deployed in AKS, thus achieving high availability.
- **Secure and highly available:** Proficy Historian for Azure is deployed on Azure Virtual Network (VNet). It uses TLS encryption, thus making it secure.

  In addition, Proficy Historian for Azure Cloud is deployed on multiple availability zones. If Data Archiver goes down, a new one is created in less than 10 seconds, thus making it highly available.

Known Issues and Limitations

Limitations

The following features available in Proficy Historian (on-premises) are *not* available in Proficy Historian for Cloud:

- Messages and alerts.
- Configuration Hub.
- Enumerated data sets, user-defined data types (UDTs), and array tags.
- Collector redundancy.
- Alarms and events archiver.
- Diagnostics Manager.
- Historian as an OPC HDA Server and OPC UA DA Server.
- The File collector, the HAB collector, the Cygnet collector, the Windows Performance Collector, and the Alarm and Events collectors.
- .NET and CA API. These do not work with Cloud Historian as Cloud Historian is Linux based.
- LDAP or SAML connections for security configuration for Cloud Historian are not supported.

In general, any features/components for which information is not available in the Proficy Historian for Cloud documentation are not supported.

In addition:

- The performance of Historian Administrator when connected to Proficy Historian for Cloud is not good. We recommend using the Web Admin console *(on page 90)*, Excel Add-in for Historian *(on page 183)*, or the Historian REST APIs *(on page 105)* instead of Historian Administrator.
- You cannot back up and restore data using any of the clients. This is because the backup and restore functions are performed using Azure File Shares (AFS). For more information how to set this up, see: https://learn.microsoft.com/en-us/azure/backup/backup-azure-files?tabs=backup-center.
- Although you can install collectors on-premises or on an Azure Virtual Machine in a VNet, you cannot install the on-premises and cloud versions of collectors on the same machine. You can choose a different machine or uninstall the existing version of collectors.
- Proficy Historian for Azure Cloud can be deployed in the regions: South Africa North,East Asia, Southeast Asia, Australia East, Australia Southeast, Canada Central,Canada East, North Europe, West Europe, France Central, Central India, South India, Japan East, Japan West, Korea Central, Norway East,Sweden Central, Switzerland North, UAE North, UK South, UK West, East US, East US2, North Central US, South Central US, West US, West US2, and West US3.

Known Issues

Description	Tracking ID
Unable to view tags which are in the sub folders of the hierarchical structure for OPC collector when browsed from the Web Admin. **Workaround:** Use the Historian Admin client for adding tags which are in the hierarchical structure.	DE196693
The Cloud Collector installer does not create instance of IFIX collector automatically when IFIX is already present in the system. **Workaround:** Manually create the instance of the iFIX collector.	DE191526
The Configuration Utility does not have the facility to provide user name and password to connect to iFIX when started in secured mode. **Workaround:** Start the service from the service manager with the proper user credentials.	DE193502
The uaa_config_tool utility displays NullReferenceException until you enter a command and begin using it.	DE182557

Chapter 2. Overview

About Proficy Historian for Azure Cloud

Proficy Historian is a best-in-class historian software solution that collects industrial timeseries data. It is secure, fast, and highly efficient.

Proficy Historian for Cloud allows you to deploy the Historian server and its components on cloud destinations. Specifically, Proficy Historian for Azure Cloud allows you to deploy the Historian server on Azure Cloud.

It supports all the clients in the on-premises counterpart of Proficy Historian (such as collectors, Excel Addin). You can install them on-premises or on an Azure Virtual Machine (either on the same or a different VNet), and then connect to Data Archiver deployed on Azure Cloud.

Advantages of using Proficy Historian for Azure Cloud:

- **Azure Marketplace:** Proficy Historian for Cloud is now available in Azure marketplace.
- **Integration with Workbook:** Using Workbook, you can access logs, which help in troubleshooting issues.
- **File Storage in Azure:** Files are stored in Azure, thus achieving scalability and security. It also offers backup and restore options.
    - **Locally redundant storage (LRS)** copies your data synchronously three times within a single physical location in the primary region. LRS is the least expensive replication option but isn't recommended for applications requiring high availability or durability.
    - **Zone-redundant storage (ZRS)** copies your data synchronously across three Azure availability zones in the primary region. For applications requiring high availability, Microsoft recommends using ZRS in the primary region, and replicating to a secondary region.
- **Deployment in Azure Kubernetes Cluster (AKS):** Proficy Historian for Azure Cloud is deployed in AKS, thus achieving high availability.
- **Secure and highly available:** Proficy Historian for Azure Cloud is deployed on an Azure Virtual Network (VNet). It uses TLS encryption, thus making it secure.

  In addition, Proficy Historian for Azure Cloud is deployed on multiple availability zones. If Data Archiver goes down, a new one is created in less than 10 seconds, thus making it highly available.

Components of Proficy Historian for Azure Cloud

Proficy Historian for Azure Cloud contains the following components:

- **Collectors:** Collect tag data from various data sources. You can install collectors *(on page 29)* on multiple Windows machines. These machines can be on-premises or on virtual network (VNet). For information on the various types of collectors, refer to Choosing a Collector *(on page 44)*.
- **Clients:** Include the following applications:
- Web Admin console *(on page 90)*
- Excel Addin *(on page 183)*
- Excel Addin for Operations Hub
- The OLEDB Provider *(on page 234)*
- Historian Administrator *(on page 383)*
- The Extract, Transform, and Load (ETL) tools

You can use them to retrieve and analyze the data stored in Historian. You can install them on multiple Windows machines. These machines can be on-premises or on virtual network (VNet).

Compatibility with Other GE Products

Several GE products work with Proficy Historian for Azure Cloud. The following is a general set of required versions to work with Proficy Historian for Azure Cloud.

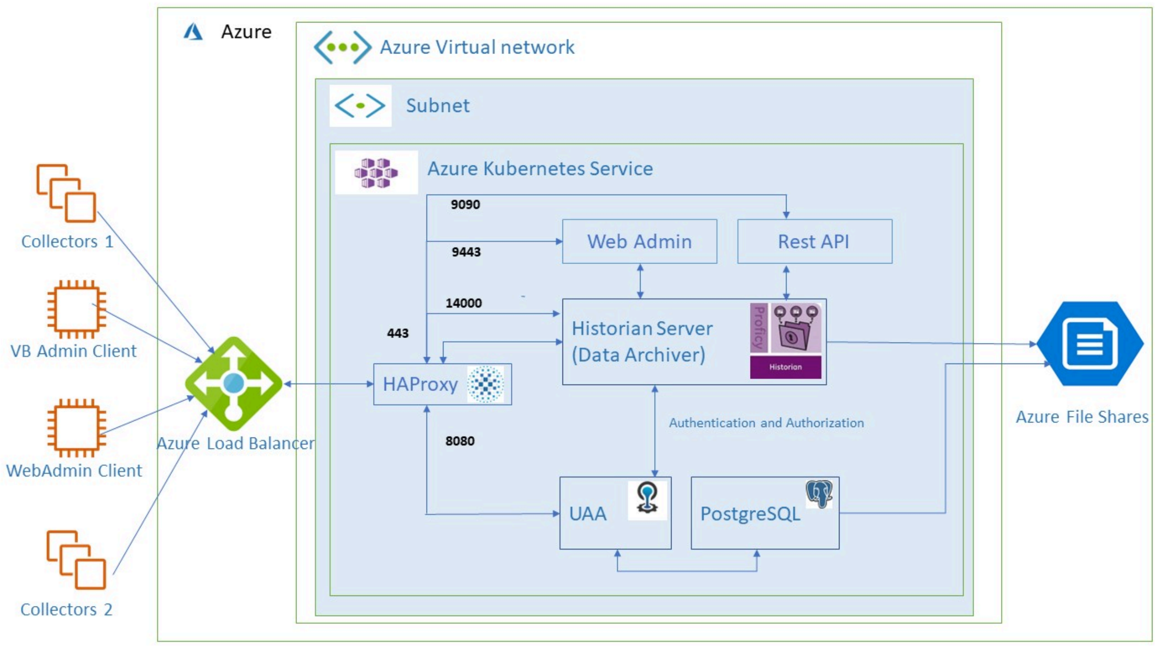
Product	Supported Version
Proficy Historian (on premises)	2023
Operations Hub	2022, 2023

Chapter 3. Deployment

Deployment Architecture

The following diagram shows the deployment architecture of Proficy Historian for Azure Cloud. In this diagram:

- Data Archiver, UAA, PostgreSQL, WebAdmin, RESTAPIs, and HAProxy containers are deployed in Azure VM (virtual machine) as part of Azure Kubernetes Service (AKS).
- Azure File Shares is mounted onto Data Archiver, PostgreSQL, WebAdmin, RESTAPI containers.
- Collector 1 and Collector 2 are collector instances created on an on-premises Windows machine. Similarly, Historian Administrator is installed on an on-premises client machine.



Prerequisites

Hardware Requirements

Proficy Historian for Azure Cloud

Proficy Historian for Azure Cloud is deployed in an Azure Virtual Machine instance in Azure Kubernetes Service (AKS) inside an Azure Virtual Network (VNet). During deployment choose an instance type based on your requirement. Our benchmarking results suggest that Standard_F8s_v2 supports up to 3 million samples per minute. You can choose an instance of lower or higher capacity based on the rate of collection.

Collectors

You can install collectors on an on-premises Windows machine. Or, you can install them on a Virtual Machine in Azure.

The following table provides the hardware requirements for installing collectors on an on-premises Windows machine.

Hardware Component	Recommendation
RAM	8 GB
Disk size	80 GB

Sizing Recommendations

We recommend the following configuration of Azure Virtual Machines depending on the type of environment or the volume of data.

Environment or Volume of Data	Recommendation
Production Environment with write rate <6MM samples/min	Standard_F8s_v2
Test environment	Standard_F8s_v2

Software Requirements

Collectors

Install any of the following operating systems:

- Microsoft® Windows® Server 2022 (64-bit)
- Microsoft® Windows® Server 2019 (64-bit)
- Microsoft® Windows® Server 2016 (64-bit)
- Microsoft® Windows® 11 (64-bit)
- Microsoft® Windows® 10 (64-bit), Professional ,or Enterprise Edition.
- Microsoft® Windows® 10 IoT Enterprise with LTSC enabled.

The other requirements, such as Microsoft®.NET Framework, are installed automatically.

Note:
If your machine is Firewall/proxy-enabled, Microsoft .NET Framework may not be installed automatically. In that case, before installing Historian, you must install Microsoft .NET Framework manually (if it is not available).

Excel Add-in for Historian

You can install Excel Add-in for Historian on an on-premises machine or on an Azure Virtual Machine in a VNet. To use Excel Add-in for Historian, install any of the following versions of Microsoft® Excel®:

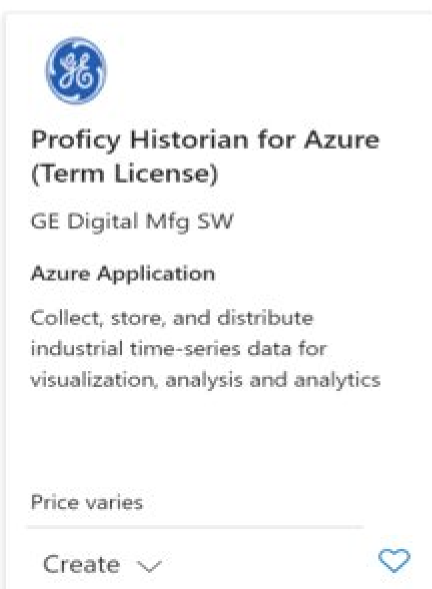
- Microsoft® Excel® 2021 (32 & 64 bit)
- Microsoft® Excel® 2019 (32 & 64 bit)
- Microsoft® Excel® 2016 (32 & 64 bit)
- Microsoft® Excel® 2013 (32 & 64 bit)

About Deploying Proficy Historian for Azure Cloud

Historian is an Azure Application Solution Template offer type.

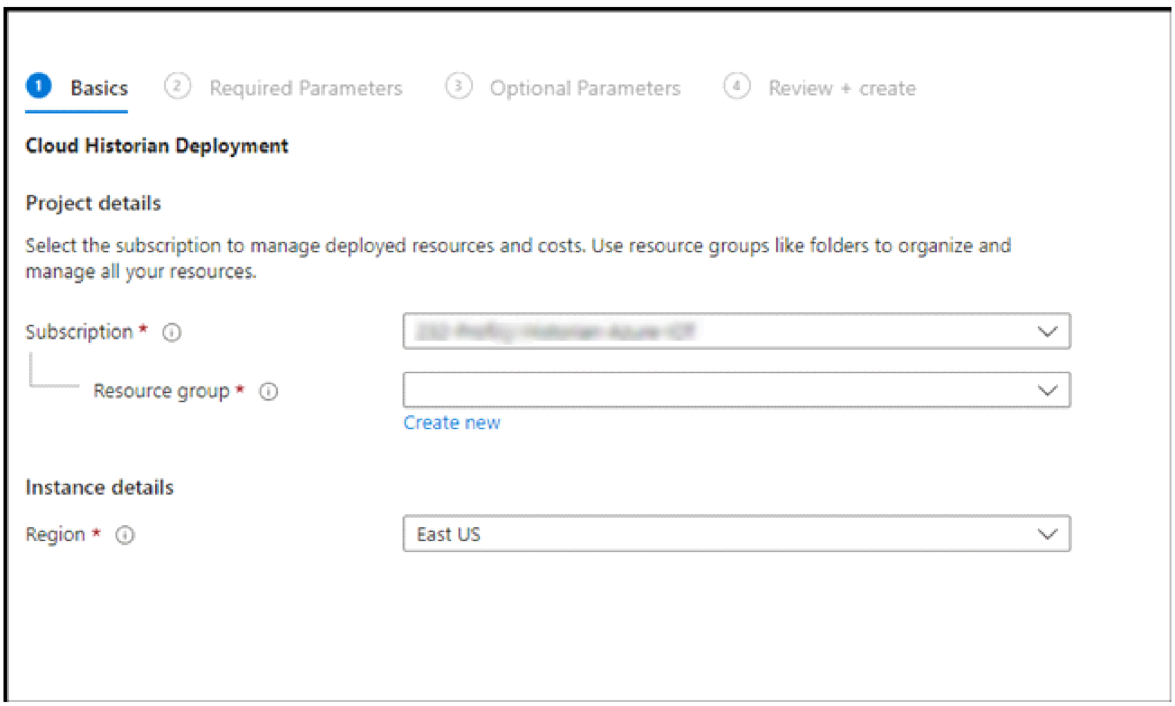
Deployment Steps

1. Login to Azure Marketplace.
2. Search for Proficy Historian.
3. In the search results, select **Proficy Historian for Azure (Term License)**.



4. Click **Create**.

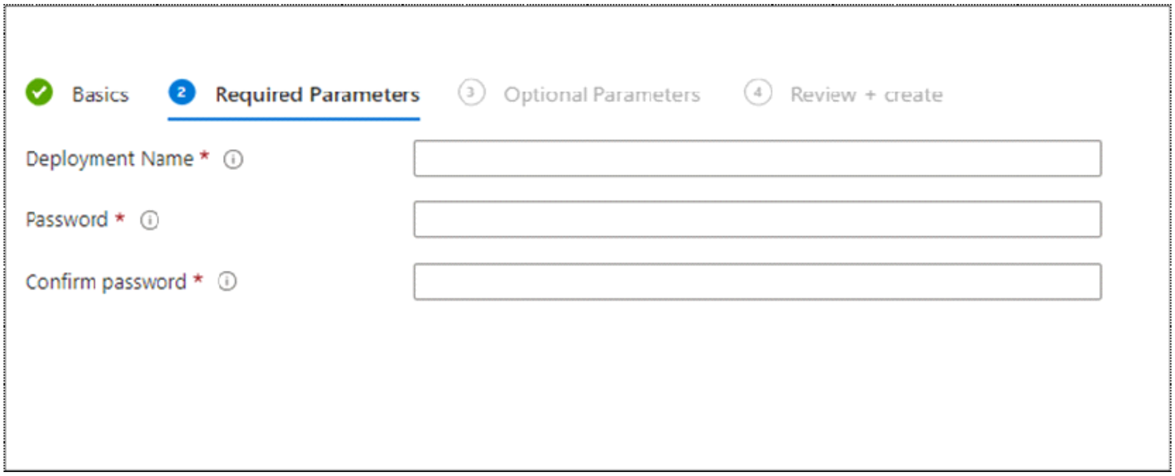
5. On the **Basics** screen, enter the **Subscription**, **Resource group**, and **Region** (if required) as defined in the following table.



Field	Description
Subscription	Select the required subscription option, in which you wish to deploy Historian from the drop-down list.
Resource Group	Select the required resource group option, in which you wish to deploy Historian from the drop-down list.
Region	If you are creating a new resource group, select the region for the new resource group. If an ex-

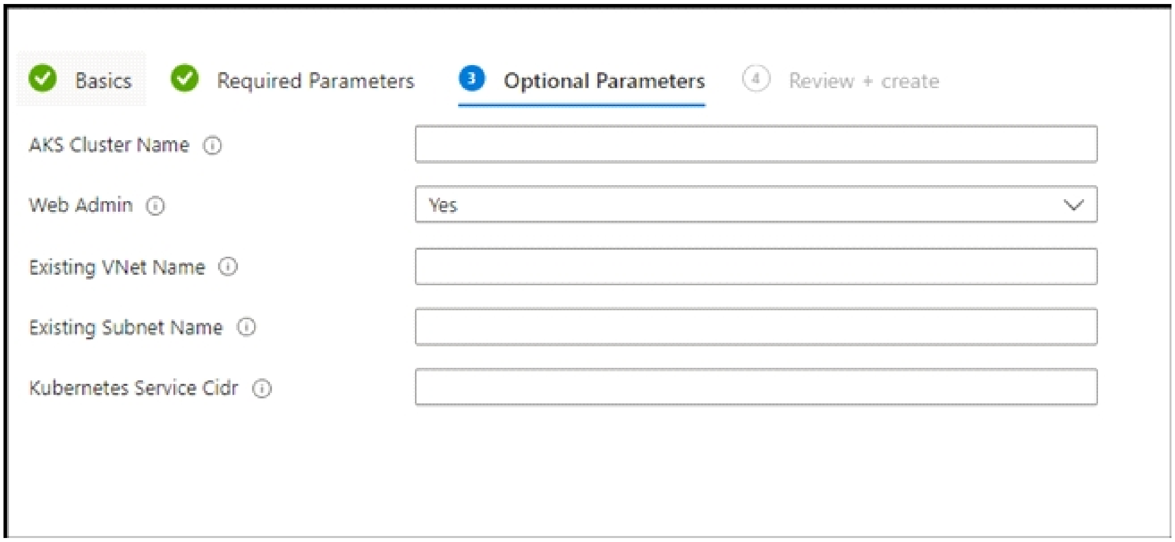
Field	Description
	isting resource group is selected, then the resources will be deployed in the region specified.

6. On the **Required Parameters** screen, enter the **Deployment Name** and **Password**.



Field	Description
Deployment Name	Enter a name for the current deployment. Constraints: Must start with an alphabet and must not container any spaces.
Password	Enter the password for the super user. Constraints: Password must contain minimum eight characters, at least one uppercase letter, one lowercase letter, one number and one special character. Password must start with the letters. Allowed Special Characters include: !@#*._^~:?-
Confirm Password	Re-enter the same password for confirmation.

7. On the **Optional Parameters** screen, enter the optional fields as described in the following table:



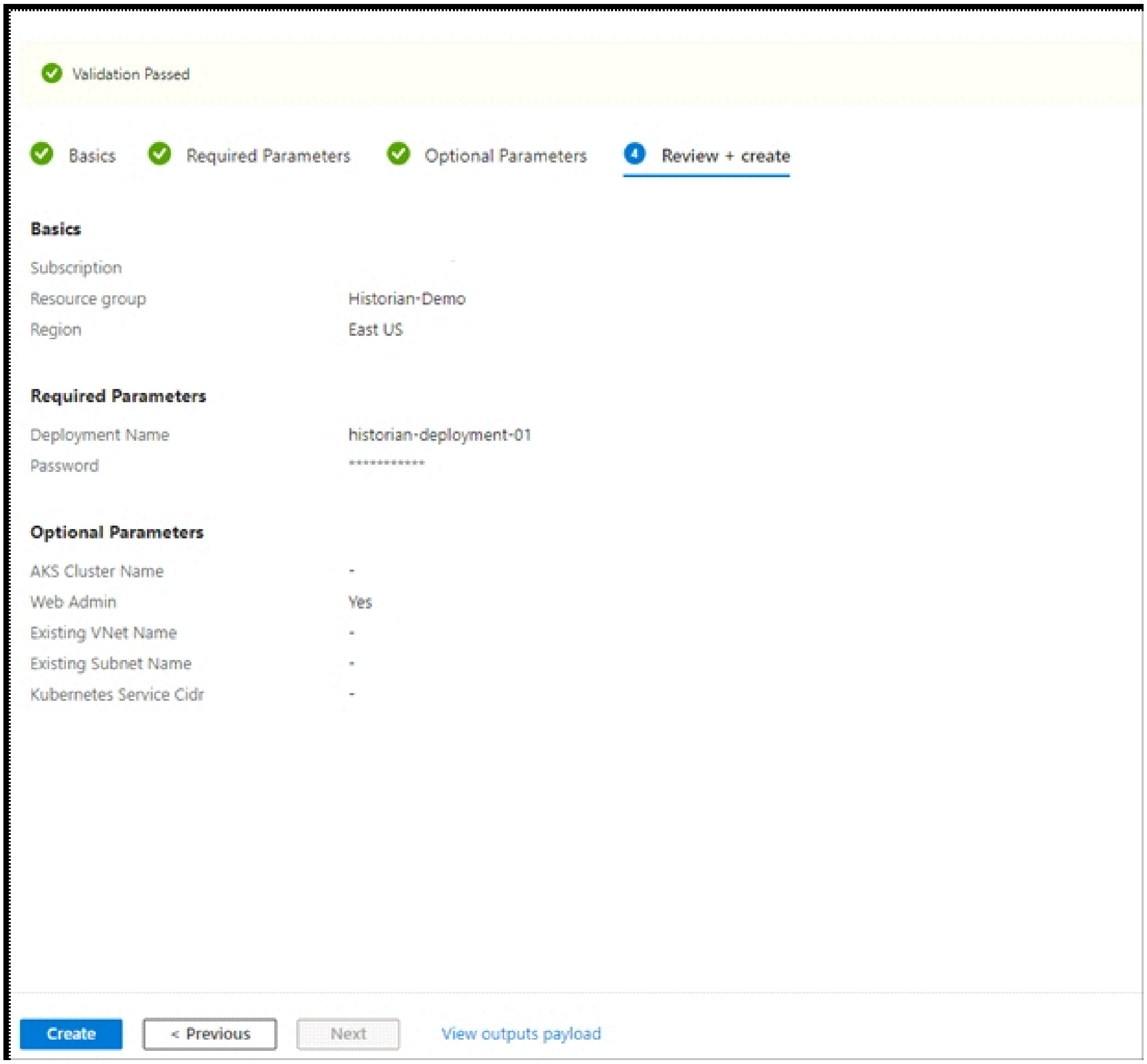
Field	Description
AKS Cluster Name	In this field, enter the AKS cluster name in which Historian will be deployed. Constraints: Cluster name must be 1-63 characters in length. The only allowed characters are letters, numbers, dashes, and the underscore. The first and last character must be a letter or a number.
Web Admin	In this field, select Yes to deploy the Historian Web Admin in AKS cluster.
Existing Vnet Name	Existing Vnet Name in which AKS worker node will run. If left empty, a new VNet will be created. Constraints: The Virtual Network name must be 2-64 characters in length. The only allowed characters are Alphanumerics, underscores, periods, and hyphens. Start with an alphanumeric, and end with an alphanumeric or underscore.
Existing Subnet Name	If this field is not entered with any values, a new subnet will be created. Constraints: The Subnet name must be 1-80 characters in length. The only allowed characters are Alphanumerics, underscores, periods, and hyphens. Start with an

Field	Description
	alphanumeric, and end an alphanumeric or underscore. **Note:** If the VNet Name specified, then it is mandatory to specify a Subnet Name. Refer to the next table for examples of various combinations of VNet and Subnet Name.
Kubernetes Service Cidr	If you are using existing VNet and if the VNet's address space is overlapping with "10.0.0.0/16", then you can specify a Kubernetes service Cidr so that it will not overlap with VNet's Address space. Refer to this link for additional details.

The following table shows examples of various combinations of VNet and Subnet Name.

vnet_-name	subnet_-name	Type
empty	empty	New Vnet and new Subnet will be created.
empty	not empty	New Vnet and new Subnet with specified name will be created.
not empty	empty	Invalid option. (If VNet specified, then its manadatory to specify subnet name)
not empty	not empty	Existing vnet and existing subnet will be used.

8. On the **Review and Create** screen, after the validation completes, click **Create**.



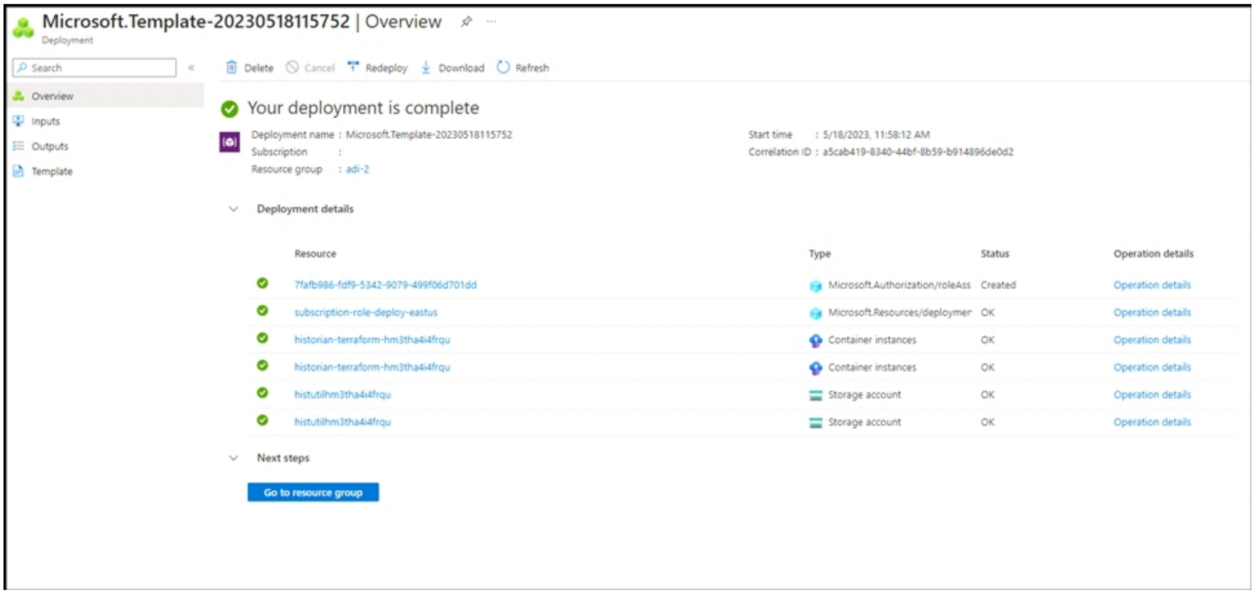
This action starts an ARM deployment.

Deployment Completed Example

The ARM template will deploy the following:

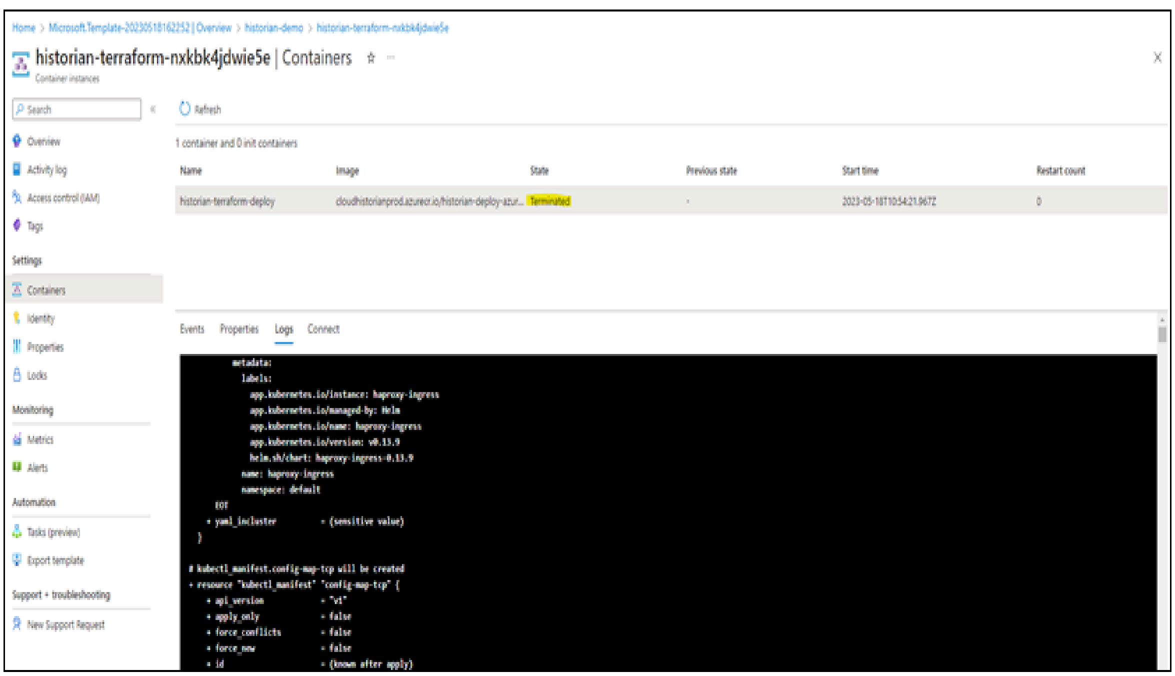
- Storage account starting with prefix: histutil<unique_string>.
- Container Instance with name: historian-terraform-<unique_string>.
- Role assignments for Container instance.

The container instance will run a docker container that will deploy the planned infrastructure using terraform. After the container instance state is terminated, you can check the status of the deployment.



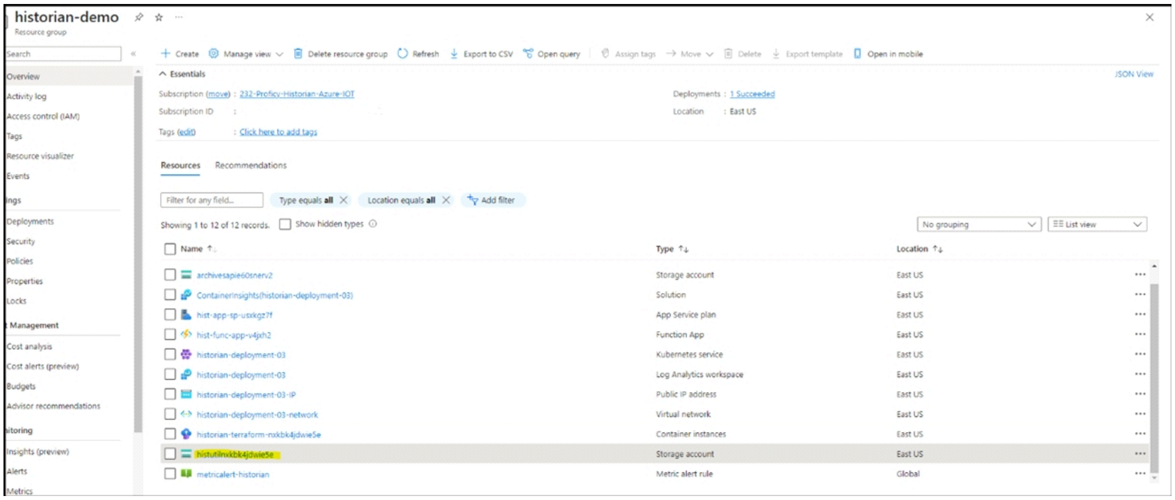
Checking the Status of a Deployment

1. Go to the resource group in which ARM template was deployed.
2. Go to the container instance with name `historian-terraform-<unique_string>`.
3. The status of the container should be Terminated. If it is still running, wait for it to stop.



4. After the container is terminated, download the terraform-output.txt file, and check for the logs in historian-deployment.log file. To download, follow these steps:

- Go to the resource group in which ARM template was deployed.
- Go to the storage account with name: `histutil<unique_string>`.



5. Go to containers and then Terraform-state. Download the file with name ending in **terraform-output.txt.**

Checking for Errors

To check for any errors during deployment, you can download the `<TIMESTAMP>_historian-deployment.log` file.

To check output of deployment, download `<TIMESTAMP>_terraform-output.txt` file

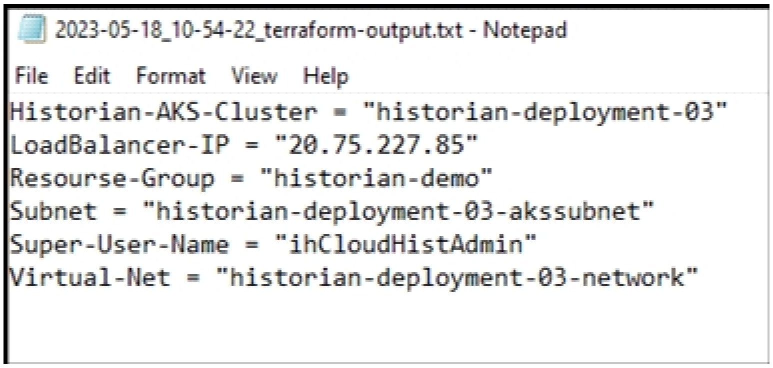
The files can be downloaded from ***Resource_Group -> histutil<unique_string> -> Containers***.

If deployment fails, the `<TIMESTAMP>_terraform-output.txt` file will not be uploaded to the blob container. In this case, check the `<TIMESTAMP>_historian-deployment.log` file.

The `<TIMESTAMP>_terraform-output.txt file` will contain value of these fields:

- Historian-AKS-Cluster
- LoadBalancer-IP
- Resourse-Group
- Subnet
- Super-User-Name
- Virtual-Net

An example of contents of output file is illustrated in the following graphic:


2023-05-18_10-54-22_terraform-output.txt - Notepad

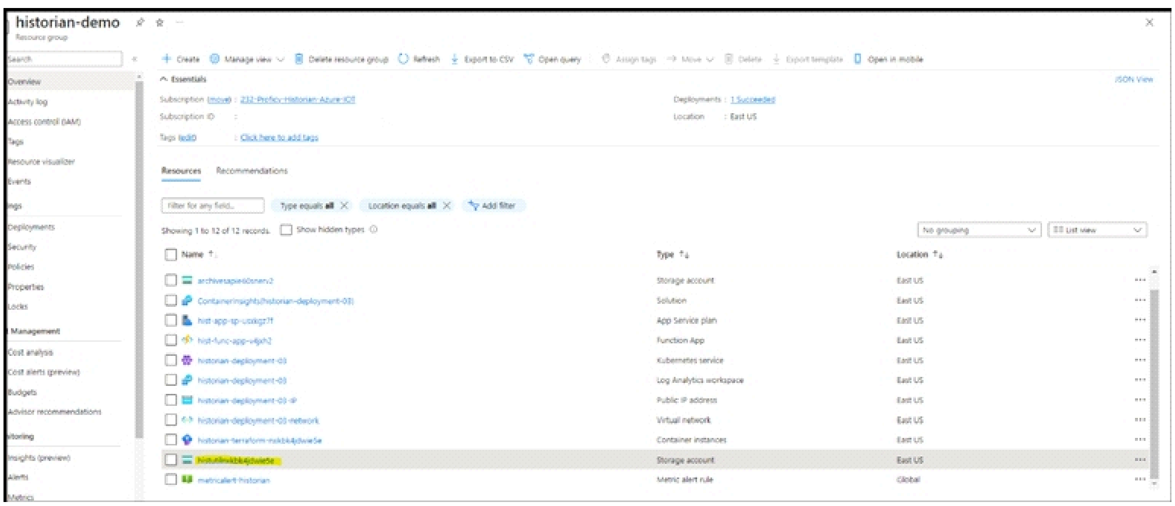
File Edit Format View Help

```
Historian-AKS-Cluster = "historian-deployment-03"
LoadBalancer-IP = "20.75.227.85"
Resourse-Group = "historian-demo"
Subnet = "historian-deployment-03-akssubnet"
Super-User-Name = "ihCloudHistAdmin"
Virtual-Net = "historian-deployment-03-network"
```

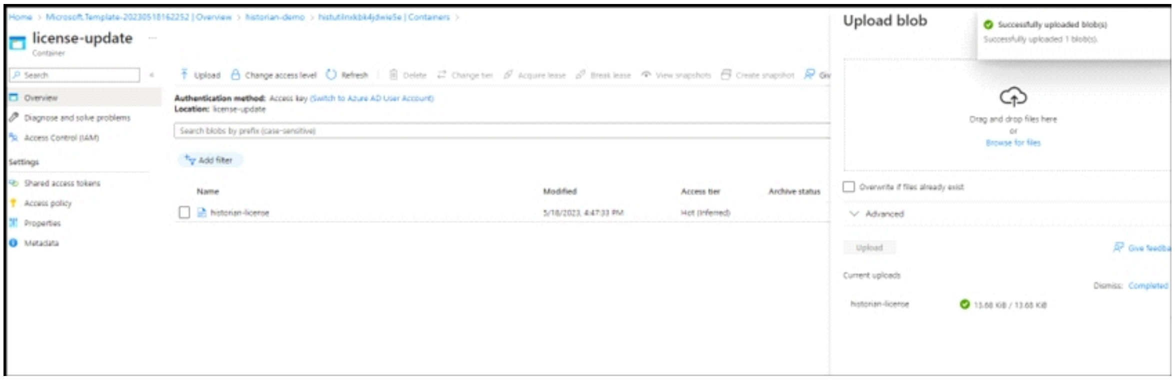
Applying the License

By default, historian will be deployed with a demo license. To update the license after successful deployment, follow below steps:

1. Go to the resource group in which ARM template was deployed.
2. Go to the storage account with name: **histutil<unique_string>**.

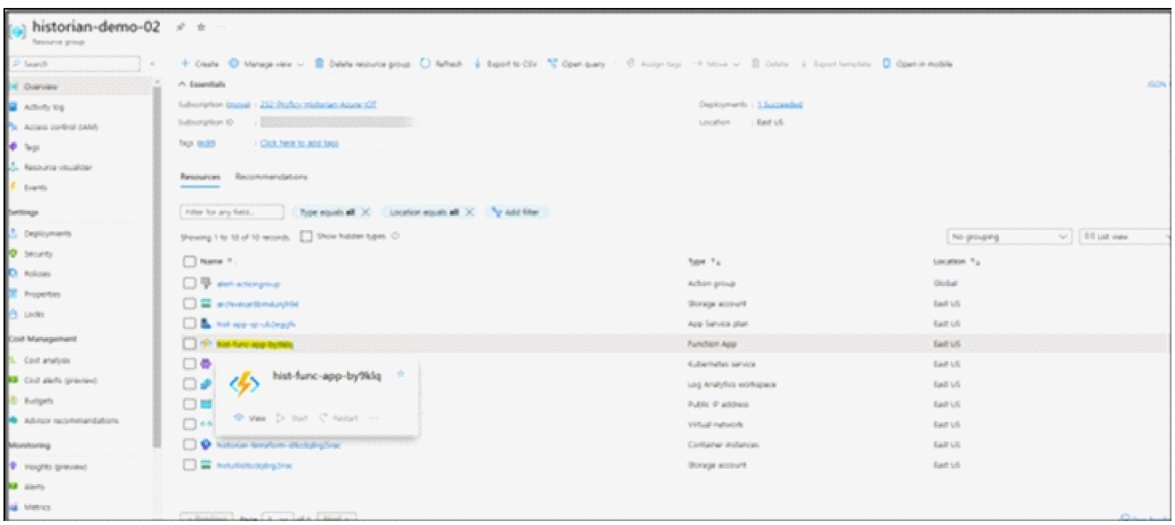


3. Go to **Containers**, and then **Terraform-state**, and **License-update**.
4. Upload the historian license file named '**historian-license**'. After uploading, an Azure function will be triggered to update the license in Data Archiver. Please wait for 5-10 minutes for the function to trigger and update the license.



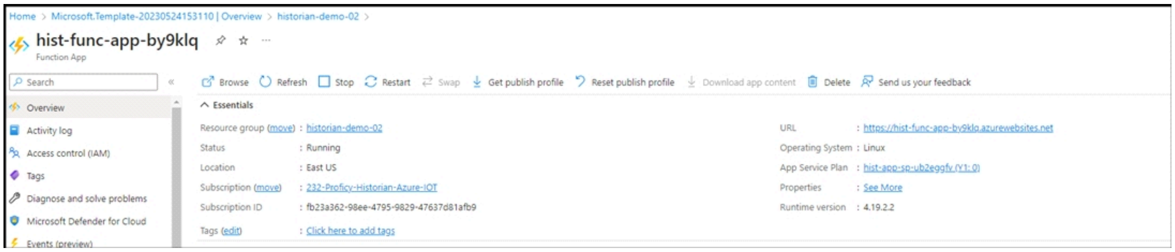
> **Note:**
> If the file is not named 'historian-license' then the Azure license-update function will not be triggered. If you have a license with a different name, rename it to 'historian-license' before uploading.

5. Go to **Resource Group**, and then **hist-func-app-<unique_string>**.



6. Ensure that the function app's status is running. If not, start the function app.

> **Note:**
> After uploading the license file to the Blob container and ensuring that the app is running, it could take at most 10 minutes for the license to apply.

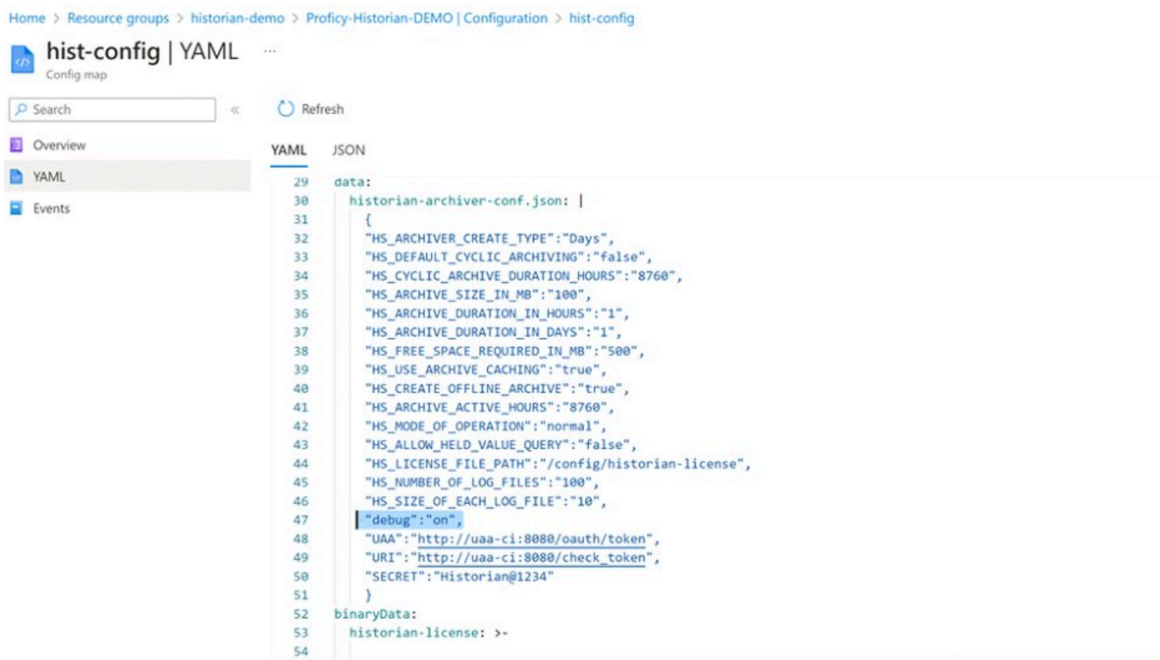


Enable Debugging

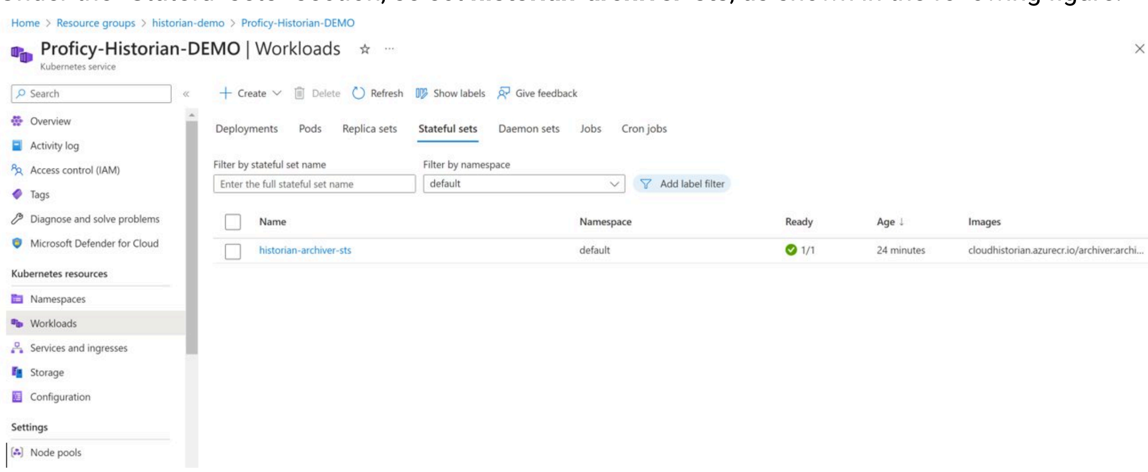
Deploy Proficy Historian for Azure Cloud *(on page 17)*.

This topic describes how to turn on the debug mode for archive logs. Using the debug mode helps in troubleshooting issues. However, there can be performance issues with the Data Archiver if debug mode is left on.

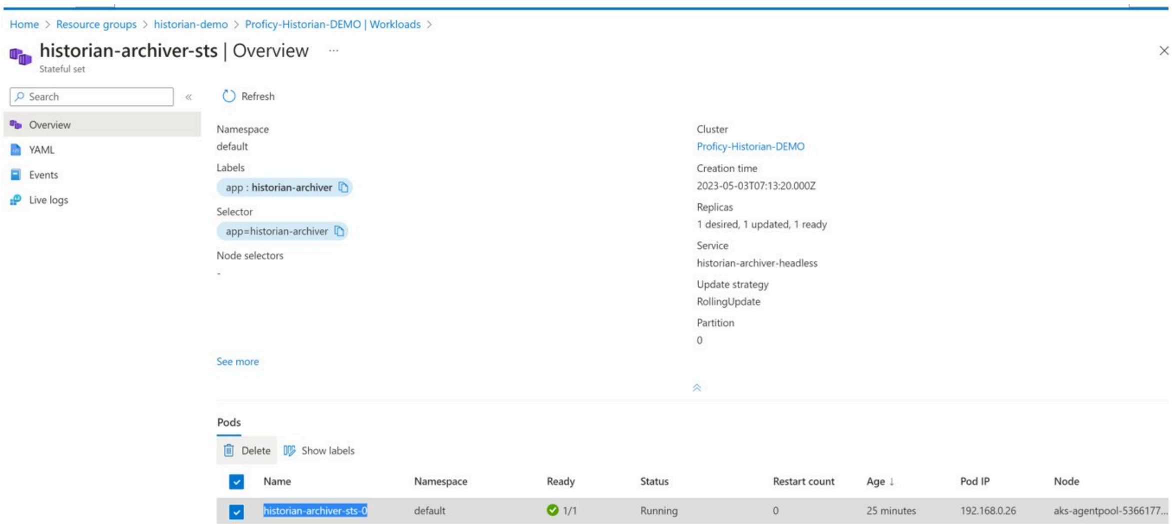
1. Login to Azure portal.
2. Go to the resource group in which Cloud Historian is deployed.
3. Go to <cluster-name> Kubernetes service -> Configuration.
4. Under the Configuration maps section, filter by the default namespace.
5. Select “hist-config”, click on YAML, and then scroll down to data section.
6. Modify the Debug field to on / off, as shown in the following figure, and then review and save your changes.



7. Go to <cluster-name> Kubernetes service -> Workloads -> Filter by default namespace.
8. Under the “Stateful sets” section, select **historian-archiver-sts**, as shown in the following figure.



9. Under Pods section, select “historian-archiver-sts-0” and then click Delete. This action will restart Data Archiver pod with new changes.



About Installing Collectors

Collectors are used to collect data from various data sources and send the data to the Historian server.

You can install collectors on-premises or on an Azure VM (Azure Virtual Machine) in a VNet (which can be the same one as the Historian server or a different one). However, you cannot install the on-premises and cloud versions of collectors on the same machine. You can choose a different machine or uninstall the existing version of collectors.

You can install collectors using the installer *(on page 29)* or at a command prompt *(on page 32)*.

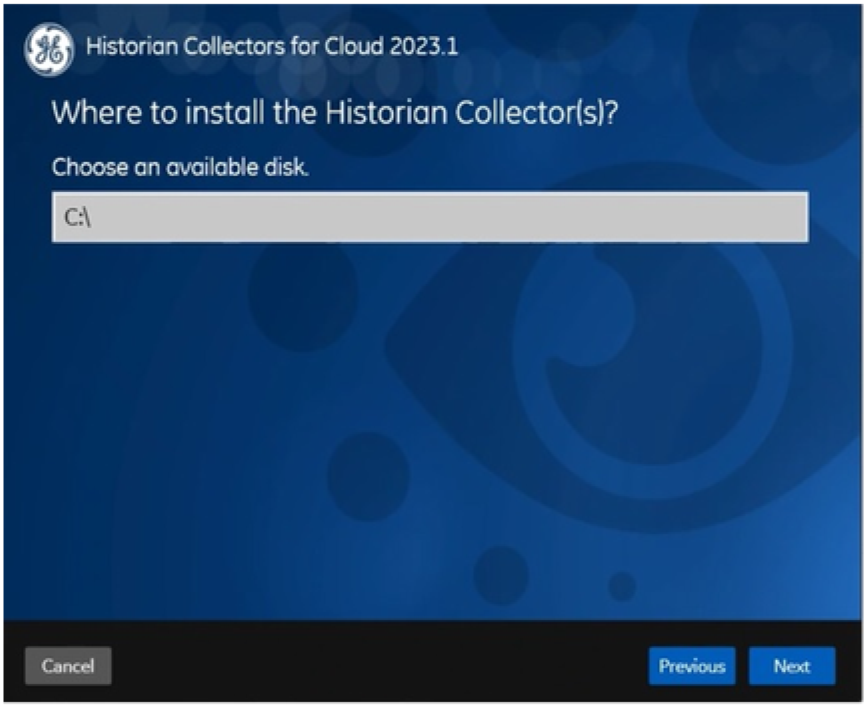
Install Collectors Using the Installer

Deploy Proficy Historian for Azure Cloud *(on page 17)*.

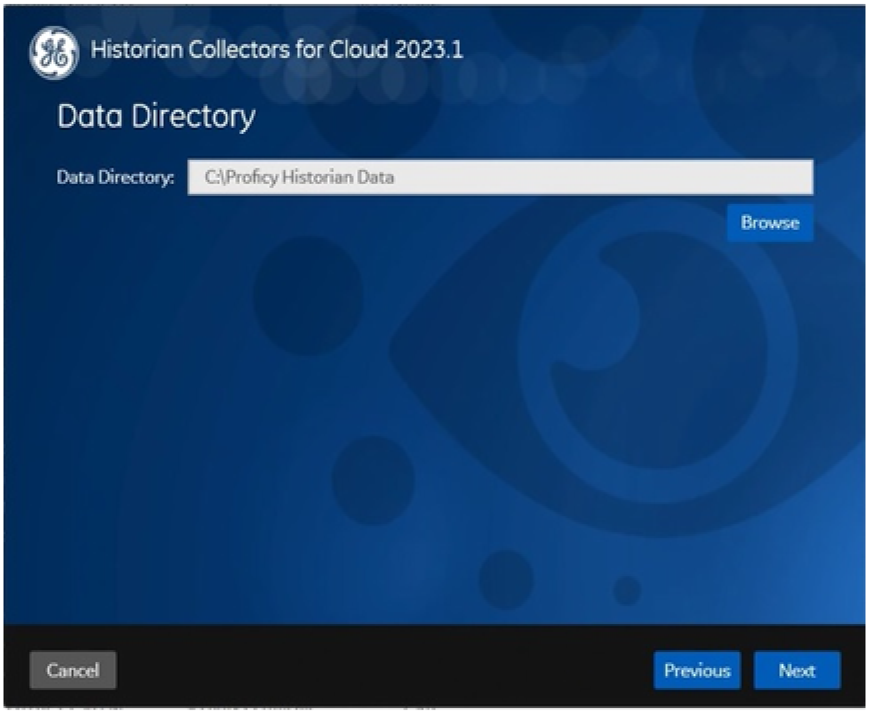
This topic describes how to install collectors using an installer. You can also install them at a command prompt *(on page 32)*.

1. Download the collectors installer from the following path: https://historian-collectors-and-clients.s3.us-east-2.amazonaws.com/collectors/Historian_Collectors_For_Cloud.zip
2. Extract the contents, and launch the collectors installer.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.

The installation drive page appears.



5. If needed, change the default installation drive, and then select **Next**.
   The data directory page appears.



6. If needed, change the folder for storing the collector log files, and then select **Next**.

   The destination Historian server page appears.
7. Enter values as described in the following table.

Field	Description
Historian Server	Enter the Azure Load Balancer IP. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.

Field	Description
User Name	Enter the username to connect to Proficy Historian for Azure Cloud.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field that you provided at the time of deployment.
Confirm Password	Re-enter the password.

8. Select **Next**.
   A message appears, stating that you are ready to install collectors.
9. Select **Install**.
   The installation begins. A message appears when the install completes. Reboot your system if prompted to do so.

- For Windows 64 bit, the 32-bit collector executable files are installed in the following folder: `<installation drive>:\Program Files (x86)\GE Digital\<collector name>`, and the 64-bit collector executable files are installed here: `<installation drive>:\Program Files\GE Digital\<collector name>`.
- For Windows 32 bit, the 32-bit collector executable files are installed in the following folder: `<installation drive>:\Program Files\GE Digital\<collector name>`. 64-bit collectors are not supported for Windows 32 bit.

Create a collector instance. For information on which collector type to use, refer to Choosing a Collector *(on page 44)*.

Installing a Collector at a Command Prompt

This topic describes how to install collectors at a command prompt. You can also install them using the installer *(on page 29)*.

1. Download the collectors installer from the following path: https://historian-collectors-and-clients.s3.us-east-2.amazonaws.com/collectors/Historian_Collectors_For_Cloud.zip
2. Extract the contents, and access the folder containing the `Collectors_Install.exe` file.
3. At a command prompt, enter:

```
Collectors_Install.exe -s RootDrive=<value> DestinationServerName=<value> DataPath=<value> UserName1=<value> Password=<value>
```

Parameter	Description	Default Value
RootDrive	The installation drive for the collectors.	`C:\`
DataPath	The folder for storing the collector log files.	`C:\Proficy Historian Data`
DestinationServer-Name	Enter the Azure Load Balancer IP. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.	local host name
UserName1	The username to connect to Proficy Historian for Azure Cloud.	
Password	The password to connect to Proficy Historian for Azure Cloud.	

For example: `Collectors_Install.exe -s RootDrive=C:\ DestinationServerName=myOrg.com DataPath=C:\Proficy Historian Data UserName1=user123 Password=xyz123`

4. Restart the machine. If you uninstall a collector or install another one before restarting the machine, an error may occur.

- For Windows 64 bit, the 32-bit collector executable files are installed in the following folder: `<installation drive>:\Program Files (x86)\GE Digital\<collector name>`, and the 64-bit collector executable files are installed here: `<installation drive>:\Program Files\GE Digital\<collector name>`.
- For Windows 32 bit, the 32-bit collector executable files are installed in the following folder: `<installation drive>:\Program Files\GE Digital\<collector name>`. 64-bit collectors are not supported for Windows 32 bit.

Create a collector instance. For information on which collector type to use, refer to Choosing a Collector *(on page 44)*.

Installing Historian Administrator

Install Historian Administrator Using the Installer

If you already have Historian Administrator on your machine (installed using on-premises Proficy Historian), you can just change the destination to the Azure Load Balancer IP, and begin using it:

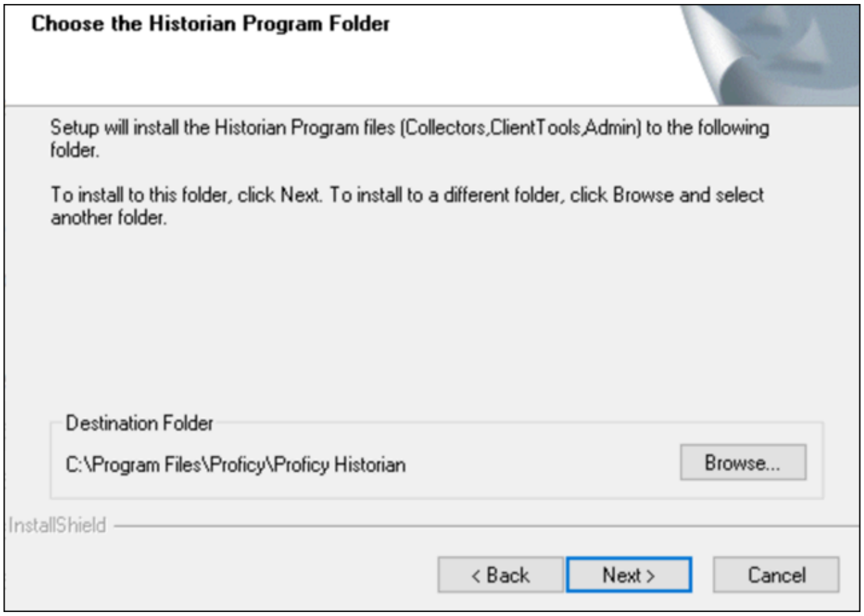
1. Select **Main**.

   A login window appears.

2. Provide the Azure Load Balancer IP, username, password, and domain information, and then select **OK**.

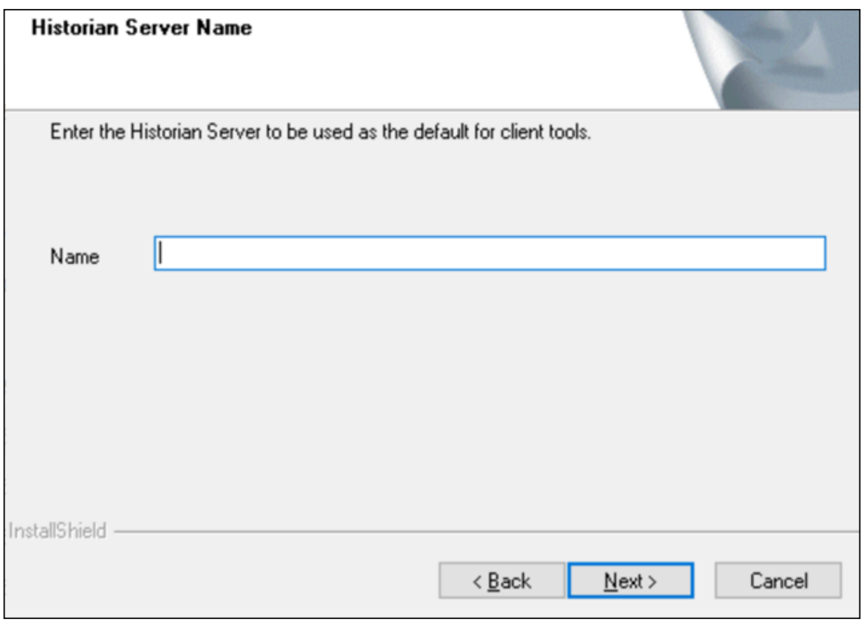
This topic describes how to install Historian Administrator using the installer. You can also install it at a command prompt *(on page 36)*.

1. Run the `InstallLauncher.exe` file. Contact the support team for this installer.
2. Select **Install Client Tools**.
   The **Select Features** page appears, displaying a list of components.
3. Select the **Historian Administrator** check box.
4. Select **Next**.
   The **Choose the Historian Program Folder** page appears.



5. As needed, change the destination folder of Historian Administrator, or leave the default folder, and then select **Next**.

   The **Historian Server Name** page appears.



6. Enter the Azure Load Balancer IP of Proficy Historian for Azure Cloud that you want to use with Historian Administrator, and then select **Next**.

> **Tip:**
>
> To find the Azure Load Balancer IP:

a. Go to the Azure portal.
b. Go to the **Resource Group** that was specified during deployment.
c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
d. Select or copy the IP Address.

7. When you are asked to reboot your system, select **Yes**.

Install Historian Administrator at a Command Prompt

Install Historian Administrator using the installer *(on page 34)* on a machine. When you do so, a template file named `setup.iss` is created at `C:\Windows`. This file stores the installation options that you have provided. You can then use this template to install Historian Administrator at a command prompt on other machines.

1. Copy the `setup.iss` file to the machine on which you want to install Historian Administrator at a command prompt.
2. In the folder in which you have copied the file, run the following command: `setup.exe /s /sms`
   The installer runs through the installation steps.

   Note:
   If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Historian Administrator is installed.

Installing the Excel Add-in for Historian

Install the Historian Excel Add-in Using the Installer

Install one of the following 32-bit or 64-bit Microsoft® Excel® applications:

- Microsoft® Excel® 2021 (32 & 64 bit)
- Microsoft® Excel® 2019 (32 & 64 bit)
- Microsoft® Excel® 2016 (32 & 64 bit)
- Microsoft® Excel® 2013 (32 & 64 bit)

This topic describes how to install Excel Add-In using the installer. You can also install it at a command prompt *(on page 37)*.

1. Run the `InstallLauncher.exe` file. Contact the Azure support team for the installer.
2. Select **Historian Excel Add-in**.
   The installer runs through the installation steps.

   > **Note:**
   > If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Excel Add-In is installed.

Activate Excel Add-In *(on page 186)*.

Install the Historian Excel Add-in at a Command Prompt

1. Install one of the following 32-bit or 64-bit Microsoft® Excel® applications:
   - Microsoft® Excel® 2019
   - Microsoft® Excel® 2016
2. Install Excel Add-in using the installer *(on page 36)* on a machine. When you do so, a template file named `setup.iss` is created at `C:\Windows`. This file stores the installation options that you have provided during the installation. You can then use this template to install Excel Add-in at a command prompt on other machines.

This topic describes how to install the Excel Addin for Historian at a command prompt. You can also install it using the installer *(on page 36)*.

1. Copy the `setup.iss` file to each machine on which you want to install Excel Add-in at a command prompt.
2. In the folder that contains the `setup.iss` file, run the following command: `setup.exe /s /sms`
   The installer runs through the installation steps.

> **Note:**
> If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Excel Add-In is installed.

Activate Excel Add-In *(on page 186)*.

Implementing Security

Default Security Groups

This topic provides a list of the default security groups created in Historian, along with the default user, ihCloudHistAdmin, for the ih_security_admins group. The password for this user is the one you enter in the at the time of deployment.

ih_security_admins

Historian power security users. Security administrators have rights to all Historian functions. By default, a user named ihCloudHistAdmin is added in this group.

ih_collector_admins

Allowed to add collector instances and change their destination.

ih_tag_admins

Allowed to create, modify, and remove tags. Tag-level security can override rights given to other Historian security groups. Tag admins can also browse collectors.

ih_archive_admins

Allowed to create, modify, and remove archives.

ih_unaudited_writers

Allowed to write data without creating any messages.

ih_unaudited_logins

Allowed to connect to Data Archiver without creating login successful audit messages.

ih_audited_writers

Allowed to write data and to produce a message each time a data value is added or changed.

Tag, archive, and collector changes log messages regardless of whether the user is a member of the ih_audited_writers group.

ih_readers

Allowed to read data and system statistics. Also allowed access to Historian Administrator.

The following table provides the types of user groups you must create based on your requirement.

Function	iH Security Admins	iH UnAudited Writers	iH UnAudited Login	iH Audited Writers	iH Readers	iH Archive Admins	iH Tag Admins	iH Collector Admins
Manage tags	X						X	
Create archive	X					X		
Read data	X				X			
Write data (unaudited)	X	X	X					
Write data (audited)	X			X				
Modify data	X	X	X	X				
Update tag security	X							
Migrate	X							
Login connection messages	X	X		X	X	X	X	X

Function	iH Security Admins	iH UnAudited Writers	iH Unaudited Login	iH Audited Writers	iH Readers	iH Archive Admins	iH Tag Admins	iH Collector Admins
Recalculate data	X		X	X				X

Note:
Regardless of the security group to which a user belongs, the user has full privileges to the Web Admin console.

For instructions on creating and managing users, refer to Managing Users and Groups *(on page 40)*.

Managing Users and Groups

Historian provides default security groups *(on page 38)* and a user, ihCloudHistAdmin, for the ih_security_admins group. This topic describes how to create more users and add them to groups. You can also delete a user or remove the user from a group.

1. Access the folder containing the `uaa_config_tool.exe` file. It is provided with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
2. To create a user, run the following command:

```
uaa_config_tool.exe add_user
-u <username>
-p <password>
-s <admin password>
-t https://<Azure Load Balancer IP>:8080
```

   For *<admin password>*, enter the password that you provided at the time of deployment.

   For *<password>*, enter a value that contains:
   - Minimum eight characters
   - At least one each of uppercase and lowercase letters
   - At least one number
   - At least one special character

   Tip:
   To find the Azure Load Balancer IP:

a. Go to the Azure portal.
b. Go to the **Resource Group** that was specified during deployment.
c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
d. Select or copy the IP Address.

3. To add a user to a group, run the following command:

```
uaa_config_tool.exe add_user_to_group

-g <group name>

-u <username>

-p <password>

-s <admin password>

-t https://<Azure Load Balancer IP>:8080
```

where *<admin password>* is the password that you provided at the time of deployment.

For *<password>*, enter a value that contains:
 - Minimum eight characters
 - At least one each of uppercase and lowercase letters
 - At least one number
 - At least one special character

Tip:
For a list of default security groups, refer to Default Security Groups *(on page 38)*.

4. To remove a user from a group, run the following command:

```
uaa_config_tool.exe remove_user_to_group

-g <group name>

-u <username>

-t https://<Azure Load Balancer IP>:8080
```

5. To delete a user, run the following command:

```
uaa_config_tool.exe remove_user

-g <group name>

-u <username>

-s <admin password>

-t https://<Azure Load Balancer IP>:8080
```

Chapter 4. Sending Data to the Historian Server

About Collectors

A collector collects tag data from various data sources.

How tag data is stored if using collectors of on-premises Proficy Historian (TLS encryption is not used):

1. Collectors send a request to the Azure Load Balancer to write tag data.
2. Azure Load Balancer sends the request to HA Proxy. HA Proxy routes the traffic to Data Archiver. If user authentication is needed, the Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, Load Balancer confirms to the collectors that data can be sent.
3. Data collected by the collector instances is sent to the Azure Load Balancer.
4. Azure Load Balancer sends the data to HA Proxy and HA Proxy again routes the traffic to Data Archiver. After authentication, the Data Archiver stores the data in the Azure File Share in .iha files.

How tag data is stored if using Historian Collectors for Cloud (TLS encryption is used):

1. Collectors send a request to the Azure Load Balancer to write tag data. Since the request is encrypted, port 443 is used.
2. Azure Load Balancer forwards the request to HA Proxy. HA Proxy decrypts the request and sends it to the Data Archiver. If user authentication is needed, the Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, the Azure Load Balancer confirms to the collectors that data can be sent.
3. Data collected by the collector instances is encrypted and sent to the Azure Load Balancer using port 443. The Azure Load Balancer forwards request to HA Proxy.
4. HA Proxy decrypts the data and sends it to the Data Archiver. After authentication, the Data Archiver stores the data in the Azure File Share in .iha files.

How data is retrieved:

1. Clients (that is, the Excel Addin, the Web Admin console, the REST Query service, or Historian Administrator) send a request to the Azure Load Balancer to retrieve data.
2. The Azure Load Balancer sends the request to HA Proxy, and then HA Proxy forwards requests to the Data Archiver, which retrieves data from Azure File Shares. If, however, user authentication is needed, the Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, data is retrieved from Azure File Share.

To send data using a collector, you must:

1. Install collectors *(on page 29)*.

   You can install collectors on multiple Windows machines. These machines can be on-premises or on an Azure Virtual Network (VNet).

2. Create a collector instance.

> **Note:**
> The following collectors are not supported by Proficy Historian for Azure Cloud:
>
> - The File collector
> - The HAB collector
> - The iFIX Alarms and Events collector
> - The OPC Classic Alarms and Events collector
> - The Windows Performance collector

What does SQ mean in the cloud output and what are the sub-quality values?

The full form of SQ is Sub Quality; the values range from 1 to 13.

The following are the sub-quality values:

ihOPCNonspecific = 0

ihOPCConfigurationError=1

ihOPCNotConnected=2

ihOPCDeviceFailure=3

ihOPCSensorFailure=4

ihOPCLastKnownValue=5

ihOPCCommFailure=6

ihOPCOutOfService=7

ihScaledOutOfRange=8

ihOffLine=9

ihNoValue=10

ihCalculationError=11

ihConditionCollectionHalted=12

ihCalculationTimeout=13

Choosing a Collector

The following table provides a list of collectors supported by Proficy Historian for Azure Cloud, along with their purpose and features.

Collector Type	Purpose	Supported Data Collection	Time Resolution	Supported Data Types
The Calculation collector *(on page 51)*	Performs calculations on values stored in Data Archiver.	Both polled and unsolicited	Seconds	Floating point, integer, binary, and string
The iFIX collector *(on page 53)*	Collects data from iFIX.	Only polled	Milliseconds or seconds	Boolean, floating point, integer, and string
The MQTT collector *(on page 55)*	Collects data published to a topic using an MQTT broker. The data should be in Predix time series data format.	Only unsolicited	Seconds, milliseconds, and microsecond	Boolean, floating point, integer, and string
The ODBC collector *(on page 61)*	Collects data from an ODBC data source.	Only unsolicited	1 millisecond	Floating point, integer, and string
The OPC Classic DA collector *(on page 63)*	Collects data from any OPC 1.0 or OPC 2.0-compliant OPC Classic DA server.	Both polled and unsolicited (unsolicited for OPC 2.0 only)	1 millisecond	Floating point, integer, binary, and string
The OPC Classic HDA collector *(on page 66)*	Collects historical data from any OPC HDA 1.2 - compliant OPC server.	Only unsolicited	1 millisecond	Floating point, integer, binary, and string
The OPC UA DA collector *(on page 70)*	Collects data from any OPC UA 1.0 or OPC 2.0-compliant OPC UA DA server.	Both polled and unsolicited	1 millisecond	Floating point, integer, binary, and string
The OSI PI collector *(on page 73)*	Collects data from an OSI PI data server.	Only unsolicited	Milliseconds and seconds	Floating point, integer, and string

Collector Type	Purpose	Supported Data Collection	Time Resolution	Supported Data Types
The OSI PI Distributor *(on page 75)*	Collects data from a Historian server and sends it to an OSI PI server.	Only unsolicited	Milliseconds and seconds	Floating point, integer, and string
The Server-to-Server collector *(on page 77)*	Collects data from an on-premises Historian server and sends it to Proficy Historian for Azure Cloud, or vice versa.	Only unsolicited	100 milliseconds	Floating point, integer, and string
The Server-to-Server Distributor *(on page 81)*	Collects data from a smaller Historian server to a large, centralized Historian server.	Only unsolicited	100 milliseconds	Floating point, integer, and string
The Simulation collector *(on page 84)*	Generates random numbers and string patterns for demonstration/testing purposes. You can configure the number of tags that you want to generate.	Only polled	1 millisecond	Floating point, integer, and string
The Wonderware collector *(on page 86)*	Collects data from Wonderware.	Only unsolicited	1 millisecond	Floating point, integer, and string

Note:
The following collectors are not supported by Proficy Historian for Azure Cloud:

- The File collector
- The HAB collector
- The Cygnet collector
- The iFIX Alarms and Events collector
- The Windows Performance collector

About Installing Collectors

Collectors are used to collect data from various data sources and send the data to the Historian server.

You can install collectors on-premises or on an Azure VM (Azure Virtual Machine) in a VNet (which can be the same one as the Historian server or a different one). However, you cannot install the on-premises and cloud versions of collectors on the same machine. You can choose a different machine or uninstall the existing version of collectors.

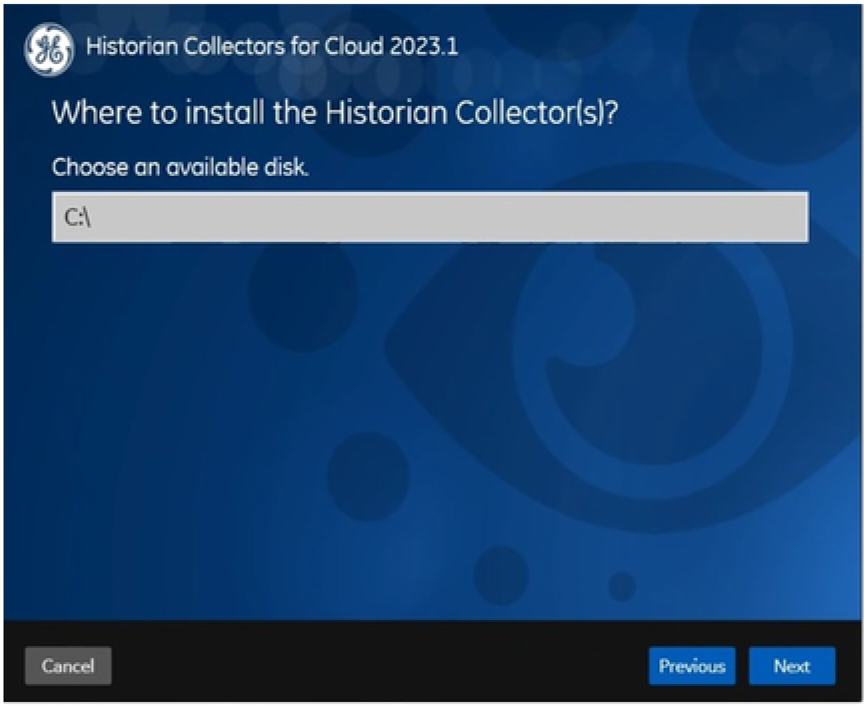
You can install collectors using the installer *(on page 29)* or at a command prompt *(on page 32)*.

Install Collectors Using the Installer

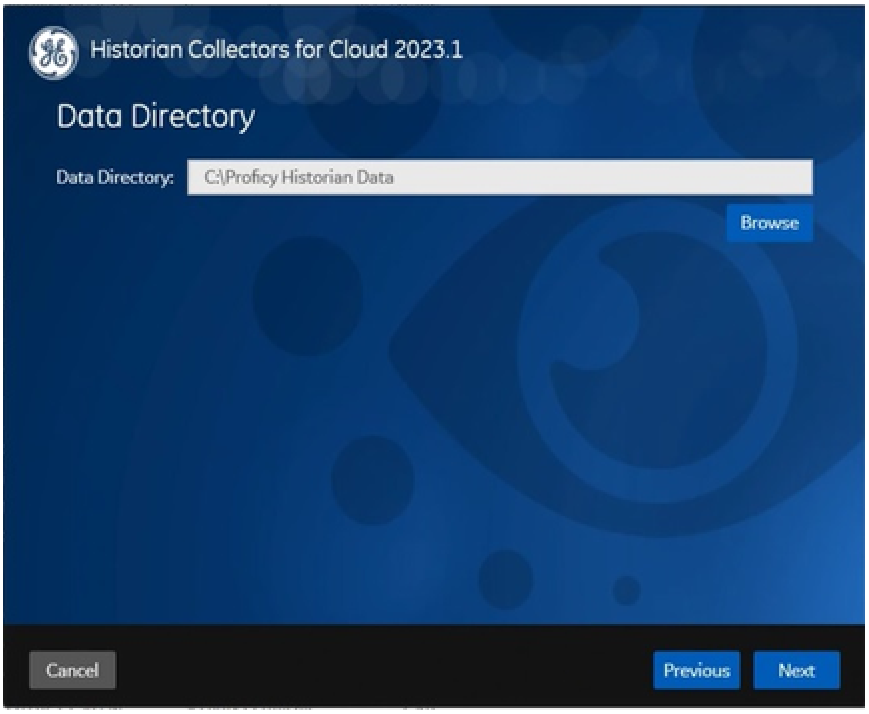
Deploy Proficy Historian for Azure Cloud *(on page 17)*.

This topic describes how to install collectors using an installer. You can also install them at a command prompt *(on page 32)*.

1. Download the collectors installer from the following path: https://historian-collectors-and-clients.s3.us-east-2.amazonaws.com/collectors/Historian_Collectors_For_Cloud.zip
2. Extract the contents, and launch the collectors installer.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.
   The installation drive page appears.



5. If needed, change the default installation drive, and then select **Next**.
   The data directory page appears.



6. If needed, change the folder for storing the collector log files, and then select **Next**.

   The destination Historian server page appears.
7. Enter values as described in the following table.

Field	Description
Historian Server	Enter the Azure Load Balancer IP. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.

Field	Description
User Name	Enter the username to connect to Proficy Historian for Azure Cloud.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field that you provided at the time of deployment.
Confirm Password	Re-enter the password.

8. Select **Next**.
   A message appears, stating that you are ready to install collectors.
9. Select **Install**.
   The installation begins. A message appears when the install completes. Reboot your system if prompted to do so.

- For Windows 64 bit, the 32-bit collector executable files are installed in the following folder: `<installation drive>:\Program Files (x86)\GE Digital\<collector name>`, and the 64-bit collector executable files are installed here: `<installation drive>:\Program Files\GE Digital\<collector name>`.
- For Windows 32 bit, the 32-bit collector executable files are installed in the following folder: `<installation drive>:\Program Files\GE Digital\<collector name>`. 64-bit collectors are not supported for Windows 32 bit.

Create a collector instance. For information on which collector type to use, refer to Choosing a Collector *(on page 44)*.

Installing a Collector at a Command Prompt

This topic describes how to install collectors at a command prompt. You can also install them using the installer *(on page 29)*.

1. Download the collectors installer from the following path: https://historian-collectors-and-clients.s3.us-east-2.amazonaws.com/collectors/Historian_Collectors_For_Cloud.zip
2. Extract the contents, and access the folder containing the `Collectors_Install.exe` file.
3. At a command prompt, enter:

```
Collectors_Install.exe -s RootDrive=<value> DestinationServerName=<value> DataPath=<value>
UserName1=<value> Password=<value>
```

Parameter	Description	Default Value
RootDrive	The installation drive for the collectors.	`C:\`
DataPath	The folder for storing the collector log files.	`C:\Proficy Historian Data`
DestinationServerName	Enter the Azure Load Balancer IP. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.	local host name
UserName1	The username to connect to Proficy Historian for Azure Cloud.	
Password	The password to connect to Proficy Historian for Azure Cloud.	

For example: `Collectors_Install.exe -s RootDrive=C:\ DestinationServerName=myOrg.com DataPath=C:\Proficy Historian Data UserName1=user123 Password=xyz123`

4. Restart the machine. If you uninstall a collector or install another one before restarting the machine, an error may occur.

- For Windows 64 bit, the 32-bit collector executable files are installed in the following folder: `<installation drive>:\Program Files (x86)\GE Digital\<collector name>`, and the 64-bit collector executable files are installed here: `<installation drive>:\Program Files\GE Digital\<collector name>`.
- For Windows 32 bit, the 32-bit collector executable files are installed in the following folder: `<installation drive>:\Program Files\GE Digital\<collector name>`. 64-bit collectors are not supported for Windows 32 bit.

Create a collector instance. For information on which collector type to use, refer to Choosing a Collector *(on page 44)*.

Creating a Collector Instance

Create a Calculation Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).

> **Important:**
> If installing on a VNet, it is recommended that you install the Calculation collector on the virtual machine deployed in the same VNet the Data Archiver is running.

The Calculation collector performs calculations on tag data that is stored in Data Archiver. It stores the calculation results in new tags. You can then access this data and plot a trend chart or analyze it.

Features: The Calculation collector performs calculations on the following values:

- Current values of other Historian tags in the same archiver.
- Previous raw samples of other tags in the same archiver.
- Calculated values of other Historian tags in the same archiver, such as minimum, maximum, average, or standard deviation. You can specify a time range for these calculations or perform a filtered query. You can use the resulting single number in a calculation.
- Interpolated values of other Historian tags in the same archiver.
- Any data retrievable using a VBScript, file I/O, ADO, and so on.

Advantages of using the Calculation collector:

- The Calculation collector can keep a history of the calculated values.
- It can perform thousands of calculations per second. Therefore, it is generally preferred to a VB SDK program performing the same functions.
- It can perform calculations on data stored in the following sources:
    - A SCADA database (such as iFIX)
    - A VB SDK program
    - A Historian collector (using input scaling)
    - By the Calculation collector
    - The Historian OLE DB provider
    - Reporting tools such as Crystal Reports
    - The Historian Excel Add-In

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
   A list of collectors that you can create appears.
4. Enter the number corresponding to the collector that you want to create.
5. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.

Field	Description
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.

The Calculation collector is created and started.

Create an iFIX Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on an Azure Virtual Machine in a VNet (which can be the same one as the Historian server or a different one).
3. Ensure that the iFIX server is running.

The iFIX collectors collect data from iFIX and store it in the Historian server.

They use the Easy Data Access (EDA) protocol to retrieve data from a running iFIX system.

Features:

- You can browse the source for tags and their attributes.
- Only the polled data collection is supported; unsolicited collection is not supported. The minimum poll interval is 100ms.
- The supported timestamp resolution is milliseconds or seconds.
- The collector accepts device timestamps.
- Floating point, integer, string, and binary data are supported.
- You can create Python Expression Tags for those collectors that support them.

Supported tag attributes:

- Tagname
- Source Address
- Engineering Unit Description
- Data Type

- Hi Engineering Units
- Lo Engineering Units

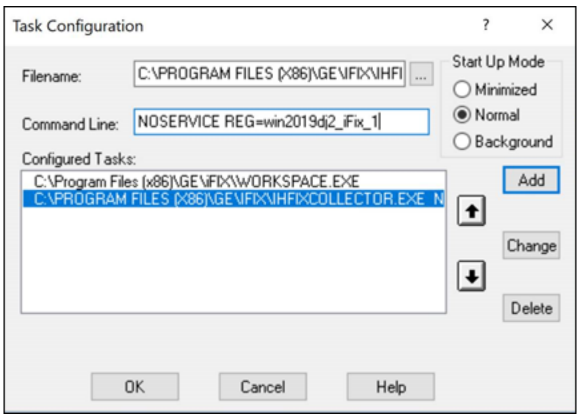
1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
   A list of collectors that you can create appears.
4. Enter the number corresponding to the collector that you want to create.
5. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered at the time of deployment.

Field	Description
iFIX Server	Enter the host name or IP address of the iFIX server. A value is required. The default value is FIX, which indicates the local machine.

The iFIX collector is created.

Start the collector: Add the collector to the iFIX System Configuration (SCU) startup list. The collector then starts automatically whenever you start iFIX. To do so, set the task parameters to `NOSERVICE REG=<collector name>`, as shown in the following image for a collector with the interface name win2019dj2_iFix_1.



Note:
If an error occurs, stating that the collector fails to start and prompting you to delete the collector:

1. Select No.
2. In iFIX System Configuration (SCU), set the task parameters as follows:
   - **Filename**: Enter `<installation drive>:\Program Files (x86)\GE Digital\Historian iFix Collector`.
   - **Command Line**: Enter `NOSERVICE REG=<collector name>`.

Create an MQTT Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).

The MQTT collector collects data published to a topic using an MQTT broker. The data should be in Predix time series data format.

Supported MQTT versions:

- MQTT V5
- MQTT V3.1.1

Supported data formats:

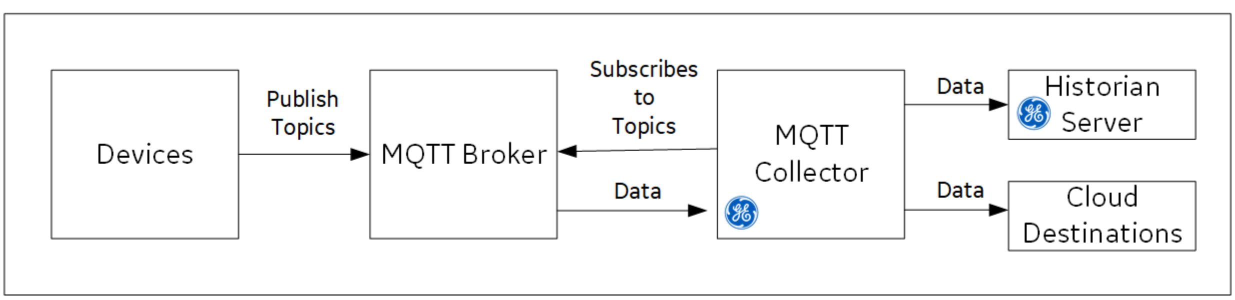
- Sparkplug B V1.0
- KairosDB (that is, the Predix Timeseries format)

Features:

- You can subscribe for multiple-level topics using a wildcard.
- Only the unsolicited data collection is supported; polled collection is not supported.
- The timestamp resolution is seconds, milliseconds, and microseconds.
- Boolean, floating point, integer, and string data types are supported.

How it works:

1. The MQTT collector connects to an MQTT broker and subscribes to a topic. You can use username/password-based authentication or certificate-based authentication. Transport Layer Security (TLS) authentication is used for subscribing the data from message broker to avoid middleware attacks so that the data is securely transferred from message broker to the MQTT collector.
2. The collector converts the data from the Sparkplug B v1.0 or the KairosDB format to a Historian-understandable format.
3. It verifies whether the tag is available in Historian; if not, it will add the tag and then add the data samples, and streams the data to the Historian server or a cloud destination.



KairosDB Message Format:

```
{
"body":
[
    {
        "attributes":{"machine_type":"<value>"},
        "datapoints":[[<value>,<value>,<value>]],
        "name":"<value>"}],
        "messageId":"<value>"}
```

The following table describes these parameters.

JSON Parameter	Description	Required/Optional
machine_type	The name of the machine from which you want to collect data.	Optional
datapoints	Time (in epoch format), value, and quality.	Required
name	The tag name	Required
messageId	The type of the message	Optional

Note:
For the parameters marked optional, you need not enter values. However, you must enter the parameter names. For example:

```
{"body":[{"attributes":{"machine_type":" "},
"datapoints":[[1558110998983,9547909,3]],"name":"QuadInteger"}],"messageId":" "}
```

Supported Data Types

Source Data Type	Historian Data Type
DoubleFloat, DoubleInteger, FixedByte, QuadInteger, SingleFloat	ihDoubleFloat
ByteString, String	ihVariableString
Boolean	ihBool

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
4. Enter the number corresponding to the collector that you want to create.
5. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.
MQTT server	Enter the IP address or host name of the MQTT broker using which you want to collect data. A value is required. By default, it considers the local host name.

Field	Description
MQTT topic	Enter the MQTT topic from which you want to collect data. A value is required. You can enter multiple topics separated by commas. If you want to use the SparkplugB format, enter a value in the following format: `namespace/group_id/message_type/edge_node_id/device_id` where: ◦ `namespace` is the Sparkplug version. Enter `spBv1.0`. ◦ `group_id` is the ID of the group of nodes from which you want to collect data. ◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types. ◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure. ◦ `device_id` a device attached to the MQTT EoN node either physically or logically. You can use the wildcard character # for any of these parameters (except for namespace).
MQTT port	Enter the port number of the MQTT broker. A value is required.
Quality of service	Enter the quality of service that you want to use while collecting data from an MQTT broker. ◦ `0`: Indicates that the message is delivered at most once or it is not delivered at all. ◦ `1`: Indicates that the message is always delivered at least once. ◦ `2`: Indicates that the message is delivered once. For more information, refer to https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/.
MQTT version	Enter the version of the MQTT broker that you want to use.
Clean session	Enter one of the following values:

Field	Description
	◦ `true`: Enter this value if you do not want to create a new session when the MQTT broker and the collector are disconnected from each other. ◦ `false`: Enter this value if you want to retain the session when the MQTT broker and the collector are disconnected from each other. This ensures that there is no loss of data. If you want to choose this option, ensure that you have entered 1 or 2 for the quality of service.

6. If you do not want to use MQTT broker authentication, for the Authentication enabled for MQTT broker field, enter `N`.
   The MQTT collector is created and started.
7. If you want to use certificate-based authentication to connect to the MQTT broker, enter values as described in the following table.

Field	Description
Authentication enabled for MQTT broker	Enter `Y`.
Certificate-based authentication	Enter `Y`.
Root CA server certificate file path	Enter the path to the CA server root file to connect to the MQTT broker. A value is required.
Client certificate file path	Enter the path to the client certificate file to connect to the MQTT broker. A value is required.
Client key file path	Enter the path to the private key file to connect to the MQTT broker. A value is required.

   The MQTT collector is created and started.
8. If you want to use username-password-based authentication to connect to the MQTT broker, enter values as described in the following table.

Field	Description
Authentication enabled for MQTT broker	Enter `Y`.

Field	Description
Certificate-based authentication	Enter N.
MQTT username	Enter the username to connect to the MQTT broker. A value is required.
MQTT password	Enter the password to connect to the MQTT broker. A value is required.

The MQTT collector is created and started.

Create an ODBC Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on an Azure Virtual Machine in a VNet (which can be the same one as the Historian server or a different one).

The ODBC collector collects data from an application based on an ODBC driver and stores the data in Data Archiver. It supports collecting of all the Historian supported data types of data from the ODBC server.

Features:

- You can browse the source for tags and their attributes on an ODBC server that supports browsing.
- Only the unsolicited data collection is supported; when changes to the ODBC source tags are detected, they are sent to the Historian server. The minimum poll interval is 100ms. The collector duplicates raw samples from the ODBC server into the Historian data archive.
- The supported timestamp resolution is 1ms.
- Floating point, integer, and string data are supported.

Supported Data Attributes:

Historian Data Type	ODBC Server Data Type
ihByte	Byte
ihFloat	SingleFloat
ihDoubleFloat	DoubleFloat
ihInteger	SingleInteger

Historian Data Type	ODBC Server Data Type
ihDoubleInteger	DoubleInteger
ihScaled	Not applicable
ihFixedString	Not applicable
ihVariableString	Not applicable
ihBlob	Not applicable
ihTime	Not applicable
ihInt64	Not applicable
ihUInt64	Not applicable
ihUInt32	Not applicable
ihUInt16	Not applicable
ihBool	Not applicable

Limitations:

- A single collector instance can collect data from a single ODBC server. To collect data from multiple ODBC servers, you must add multiple instances.
- Only good and bad quality types are supported. OPC Quality and OPC Subquality are not supported.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
   A list of collectors that you can create appears.
4. Enter the number corresponding to the collector that you want to create.
5. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required.

Field	Description
	Tip: To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.
ODBC Server	Enter the host name or IP address of the ODBC server. A value is required. By default, the local host name is considered.
ODBC Username	Enter the username to connect to the ODBC server. A value is required.
ODBC Password	Enter the password to connect to the ODBC server. A value is required.

The ODBC collector is created and started.

Create an OPC Classic Data Access (DA) Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).

The OPC Classic DA collector collects data from any OPC 1.0 or OPC 2.0 compliant OPC server. The collector automatically determines the capability of the OPC server to which it is connected and supports appropriate features based on this information.

Features:

- You can browse the source for tags and their attributes on an OPC server that supports browsing.
- Both the polled and unsolicited data collection are supported; when changes to the OPC source tags are detected, they are sent to the Historian server. Unsolicited data collection is supported for OPC 2.0 only. The minimum poll interval is 100ms. The collector duplicates raw samples from the OPC server into the Historian data archive.

  For unsolicited data collection, if collector compression is disabled, all new values produce an exception. And, the deadband percentage is determined by the collector deadband percent. You can only configure the collector deadband percent by enabling compression.
- The supported timestamp resolution is 1ms.
- Floating point, integer, binary, and string data are supported.
- Python expression tags are supported.
- Device timestamps are accepted.

Supported data types:

The OPC Data Type	Recommended Data Type in Historian
I1 - 16 bit signed integer	Single Integer
I4 - 32 bit signed integer	Double Integer
R4 - 32 bit float	Single Float
R8 - 64 bit double float	Double Float
UI2 - 16 bit unsigned single integer	Unsigned Single Integer
UI4 - 32 bit unsigned double integer	Unsigned Double Integer
UI8 - 64 bit unsigned quad integer	Unsigned Quad Integer
I8 - 64 bit quad integer	Quad Integer
BSTR	Variable String
BOOL	Boolean

The OPC Data Type	Recommended Data Type in Historian
I1 - 8 bit single integer	Byte

Note:
The collector requests data from the OPC server in the native data type. Then the collector converts the received value to a Historian Data Type before sending it to the data archiver.

Supported tag attributes:

- Tagname
- Source Address
- Engineering Unit Description
- Data Type
- Hi Engineering Units
- Lo Engineering Units

The Engineering Unit Description, Hi Engineering Units and Lo Engineering Units vary based on the OPC server vendor.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
4. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required. **Tip:** To find the Azure Load Balancer IP:

Field	Description
	a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.
OPC server	Enter the prog ID of the OPC Classic server. A value is required.

The OPC Classic DA collector is created.

5. Start the collector instance (as a Windows service).

Create an OPC Classic Historical Data Access (HDA) Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).

The OPC Classic HDA collector collects historical data from any OPC HDA 1.2 - compliant OPC server. The collector automatically determines the capability of the OPC server to which it is connected and supports the appropriate features based on this information.

Features:

- You can browse the source for tags and their attributes on an OPC server that supports browsing.
- Only unsolicited data collection is supported; when changes to the OPC source tags are detected, they are sent to the Historian server. The minimum poll interval is 100ms. The collector duplicates raw samples from the OPC server into the Historian data archive.

  For unsolicited data collection, if collector compression is disabled, all new values produce an exception. And, the deadband percentage is determined by the collector deadband percent. You can only configure the collector deadband percent by enabling compression.
- The supported timestamp resolution is 1ms.
- Floating point, integer, binary, and string data are supported.
- Device timestamps are accepted.

Supported data types:

The OPC Data Type	Recommended Data Type in Historian
I1- 16 bit signed integer	Single Integer
I4- 32 bit signed integer	Double Integer
R8- 64 bit double float	Single Float
UI2- 16 bit unsigned single integer	Double Float
UI4- 32 bit unsigned double integer	Unsigned Integer
UI8- 64 bit unsigned quad integer	Unsigned Double Integer
I8- 64 bit quad integer	Quad Integer
BSTR	Variable Sting
BOOL	Boolean
I1- 8 bit single integer	Byte

Note:
The OPC Classic HDA collector requests data from the OPC Classic HDA server in the native data type. The OPC Classic HDA collector then converts the received value to a Historian data type before sending it to Data Archiver.

Supported tag attributes:

- Tagname
- Source Address
- Engineering Unit Description
- Data Type
- Hi Engineering Units
- Lo Engineering Units

The Engineering Unit Description, Hi Engineering Units and Lo Engineering Units vary based on the OPC server vendor.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
4. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required.

Field	Description
	Tip: For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.
OPC HDA server	Enter the prog ID of the OPC Classic server. A value is required.

The OPC UA HDA collector is created.

5. Start the collector instance (as a Windows service).

Reconnect Automatically to the OPC Classic HDA Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).
3. Create an OPC Classic Historical Data Access (HDA) Collector *(on page 66)*.

You can reconnect to the OPC Classic HDA server automatically as soon as the server is up and running. By default, the collector polls for the server connection every 5 seconds. You can change this interval as well. The collector is stopped until reconnected to the server.

1. Access the following registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\iHistorian\Services\OPCHDACollector
```

2. Locate the Key created with the ProgID of the OPC Classic HDA server.
3. Create a DWORD named `EnableOPCHDAReconnect`.
4. Enter the decimal value 1.
5. If you want to change the reconnection interval (from the default value of 5 seconds):
    a. Create a DWORD named `ReconnectInterval`.
    b. Enter a decimal value between 5 and 60. This value represents the number of seconds for the collector to wait before trying to reconnect to the OPC server.
6. Select **OK**, then close the Registry.

Create an OPC UA Data Access Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).

The OPC UA DA collector collects data from a OPC UA 1.0 and OPC 2.0-compliant OPC UA DA server. The collector automatically determines the capability of the OPC UA DA server to which it is connected, and supports the appropriate features based on this information.

Features:

- You can browse the source for tags and their attributes on an OPC server that supports browsing.
- Both the polled and unsolicited data collection are supported; when changes to the OPC source tags are detected, they are sent to the Historian server. Unsolicited data collection is supported for OPC 2.0 only. The minimum poll interval is 100ms. The collector duplicates raw samples from the OPC server into Data Archiver.

  For unsolicited data collection, if collector compression is disabled, all new values produce an exception. And, the deadband percentage is determined by the collector deadband percent. You can only configure the collector deadband percent by enabling compression.
- The supported timestamp resolution is 1ms.
- Floating point, integer, binary, and string data are supported.
- Python expression tags are supported.
- Device timestamps are accepted.

Supported data types:

OPC UA DA Collector Data Type	Recommended Data Type in Historian
`OpcUaType_Null`	`ihTKVariableString`
`OpcUaType_Boolean`	`ihTKBool`
`OpcUaType_SByte`	`ihTKByte`
`OpcUaType_Byte`	`ihTKByte`
`OpcUaType_Int16`	`ihTKInteger`
`OpcUaType_UInt16`	`ihTKUInt16`

OPC UA DA Collector Data Type	Recommended Data Type in Historian
`OpcUaType_Int32`	`ihTKDoubleInteger`
`OpcUaType_UInt32`	`ihTKUInt32`
`OpcUaType_Int64`	`ihTKInt64`
`OpcUaType_UInt64`	`ihTKUInt64`
`OpcUaType_Float`	`ihTKFloat`
`OpcUaType_Double`	`ihTKDoubleFloat`
`OpcUaType_DateTime`	`ihTKVariableString`
`OpcUaType_Guid`	`ihTKDataTypeUndefined`
`OpcUaType_StatusCode`	`ihTKDataTypeUndefined`
`OpcUaType_String`	`ihTKVariableString`
`OpcUaType_ByteString`	`ihTKDataTypeUndefined`
`OpcUaType_XmlElement`	`ihTKDataTypeUndefined`
`OpcUaType_NodeId`	`ihTKDataTypeUndefined`
`OpcUaType_ExpandedNodeID`	`ihTKDataTypeUndefined`
`OpcUaType_DiagnosticInfo`	`ihTKDataTypeUndefined`
`OpcUaType_QualifiedName`	`ihTKDataTypeUndefined`
`OpcUaType_LocalizedText`	`ihTKDataTypeUndefined`
`OpcUaType_ExtensionObject`	`ihTKDataTypeUndefined`
`OpcUaType_DataValue`	`ihTKDataTypeUndefined`

Supported tag attributes:

- Tagname
- Source Address
- Engineering Unit Description
- Data Type
- Hi Engineering Units
- Lo Engineering Units

The Engineering Unit Description, Hi Engineering Units and Lo Engineering Units vary based on the OPC server vendor.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
4. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.

Field	Description
OPC UA Server URI	Enter the URI to connect to the OPC server in the following format: `opc.tcp://<host name or IP address of the OPC UA server>:<port number>`

The OPC UA DA collector is created and started.

Create an OSI PI Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).
3. Install PI AF SDK version 2.7.5 or later.

The OSI PI collector collects data from an OSI PI data server and sends it to Data Archiver. Data is collected directly from OSI PI Data Archive v3.2 or later via OSI PI AOSI PI v1.3.4 or later.

One OSI PI collector can collect data from a single OSI PI data server. To collect from multiple OSI PI data servers, you must create multiple OSI PI collector instances.

Features

- You can browse the source for tags and their attributes.

> **Note:**
> Tag browsing performance with OSI PI has been confirmed as satisfactory up to 130,000 tags. Beyond that threshold, OSI PI may take a long time to return the large number of tags. In such a case, we recommend that you first export the tags to an Excel worksheet.

- Only unsolicited data collection is supported.
- The supported timestamp resolution is milliseconds or seconds.
- Python expression tags are supported.
- Floating point, integer, and string data are supported.
- Device timestamps are accepted.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.

3. Enter the number corresponding to creating a collector instance.
4. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.
OSI PI server	Enter the host name or IP address of the machine on which the OSI PI server is installed. A value is required.
OSI PI username	Enter the username to connect to the OSI PI server. If you just press ENTER, the default username is used.
OSI PI password	Enter the password to connect to the OSI PI server. If you just press ENTER, the default password is used.

Field	Description
PI source	Specify whether the OSI PI source is archive or snapshot. If you just press ENTER, *archive* is considered as the PI source.

The OSI PI collector is created and started.

Create an OSI PI Distributor

The OSI PI distributor collects data from a Historian server and sends it to an OSI PI server. You can use OSI PI v1.3.4 or greater.

The OSI PI distributor uses unsolicited distribution, whereby changes in Historian tags values are detected, and are forwarded to a remote OSI PI data server. The distributor duplicates data from a Historian archive to an OSI PI data archive.

One OSI PI distributor can distribute data to a single OSI PI data archive. To distribute to multiple OSI PI archives from an Historian archive, you must create multiple instances of the OSI PI distributor. You can also configure multiple OSI PI distributors for a single OSI PI data archive.

> **Note:**
> The OSI PI distributor can send data only to PI Archive, not to PI Snapshot.

Features:

- You can browse the source for tags and their attributes on an OSI PI server.
- Only unsolicited data collection is supported.

  For unsolicited data collection, if collector compression is disabled, all new values produce an exception. And, the deadband percentage is determined by the collector deadband percent. You can only configure the collector deadband percent by enabling compression.
- The supported timestamp resolution is milliseconds or seconds.
- Floating point, integer, and string data are supported.
- Device timestamps are accepted.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.

4. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.
OSI PI server	Enter the host name or IP address of the machine on which the OSI PI server is installed. A value is required.
OSI PI username	Enter the username to connect to the OSI PI server. If you just press ENTER, the default username is used.
OSI PI password	Enter the password to connect to the OSI PI server. If you just press ENTER, the default password is used.

Field	Description
PI source	Specify whether the OSI PI source is archive or snapshot. If you just press ENTER, *archive* is considered as the PI source.

The OSI PI distributor is created and started.

Create a Server-to-Server Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).

Using the Server-to-Server collector, you can send data as described in the following table.

Source	Destination
On-premises Historian server	Proficy Historian for Azure Cloud
Proficy Historian for Azure Cloud	On-premises Historian server
Proficy Historian for Azure Cloud	Proficy Historian for Azure Cloud

Important:
If you want to send data from cloud to on-premises, you must disable TLS encryption by setting the following registry entry to 0: `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE Digital\iHistorian\Services\ServerToServerCollector\<collector instance>\TLSEnable`

Features:

- You can browse the source for tags and their attributes.
- Only unsolicited data collection is supported.
- The supported timestamp resolution is 100 milliseconds.
- Data compression is supported.
- Floating point, integer, and string data are supported.
- Device timestamps are accepted.

When a time-based or an event-based trigger of a destination tag occurs:

1. The calculation formula for the destination tag is executed.

   This typically involves fetching data from one or more tags on the source server.
2. A raw sample or calculation error is determined.

   You can use conditional logic in your calculation formula to determine if a sample should be sent to the destination.
3. The raw sample is delivered to the destination server, utilizing store and forward when necessary.

- When a tag is added by browsing, only certain tag properties are copied from the source tag to the destination tag. Consider what properties are necessary for your application and configure them manually. For information on which properties are copied, refer to Tag Properties that are Copied.
- If you change a tag property on the source tag (EGU Limits, descriptions, and so on), the property does not automatically change on the destination tag. You can manually change the properties of a destination tag.

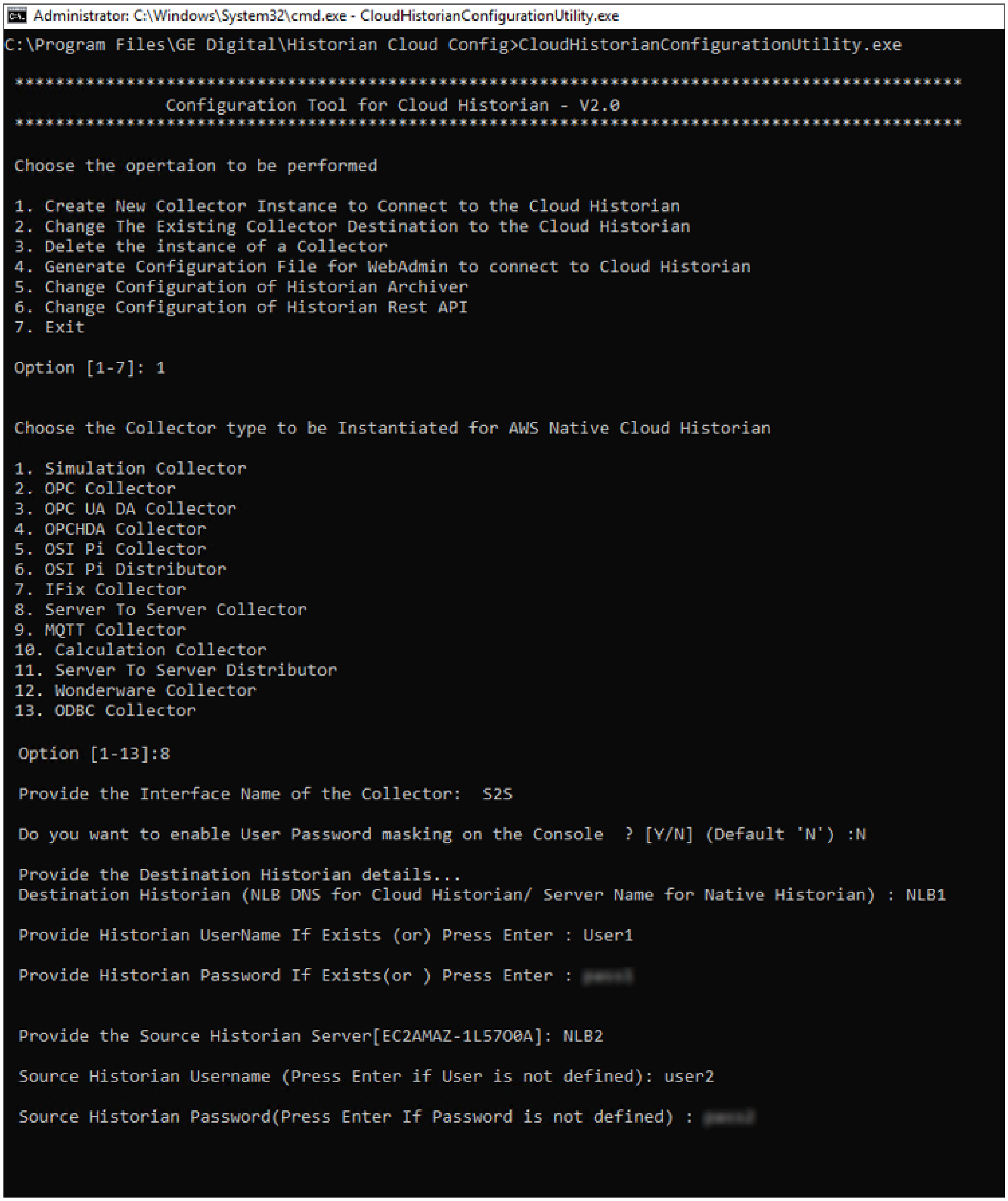
Best Practices

- We recommend that you install the Server-to-Server collector on the source Historian machine. When you do so, the collector can preserve the collected data (store and forward) even if the collector and the destination server become disconnected.
- Collection on a tag-by-tag basis is preferred, according to scheduled poll times or upon data changes. One sample is collected for each trigger.
- The Server-to-Server collector can perform calculations on multiple input tags as long as the input tags are on the same source Historian.
- Use polled triggers to perform scheduled data transformations like daily or hourly averages. Use unsolicited triggers to replicate data in real time, as it changes.
- Use event-based triggers to replicate data throughout the day. The samples can be held incoming and outgoing store and forward buffer when necessary. You cannot schedule batch replication of raw samples. For example, you cannot, at the end of the day, send all raw samples for tags to the destination.
- All input source tags for the calculations must originate from the source archiver. For instance, you cannot directly add a tag from `server1` plus a tag from `server2` and place the result on `server2`. You can, however, collect tags from `server1` to `server2`, and then use the Server-to-Server collector to accomplish this. This requires two Server-to-Server collector instances, one running on each machine.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
4. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
User Password Masking	Enter **Y** if you want to mask the password, or **N** if you want to proceed without masking the password.
Destination Historian Server	Enter the host name or IP address where the destination Historian server is installed.
Destination Historian Username	Enter the username to connect to the destination server.
Destination Historian Password	Enter the password to connect to the destination server.
Source Server	Enter the host name or IP address where the source Historian server is installed.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. If you want to send data to an on-premises Historian server, press ENTER.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. If you want to send data to an on-premises Historian server, press ENTER. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.

The Server-to-Server collector is created and started.



```
Administrator: C:\Windows\System32\cmd.exe - CloudHistorianConfigurationUtility.exe
C:\Program Files\GE Digital\Historian Cloud Config>CloudHistorianConfigurationUtility.exe

 ***********************************************************************************************
               Configuration Tool for Cloud Historian - V2.0
 ***********************************************************************************************

 Choose the opertaion to be performed

 1. Create New Collector Instance to Connect to the Cloud Historian
 2. Change The Existing Collector Destination to the Cloud Historian
 3. Delete the instance of a Collector
 4. Generate Configuration File for WebAdmin to connect to Cloud Historian
 5. Change Configuration of Historian Archiver
 6. Change Configuration of Historian Rest API
 7. Exit

 Option [1-7]: 1


 Choose the Collector type to be Instantiated for AWS Native Cloud Historian

 1. Simulation Collector
 2. OPC Collector
 3. OPC UA DA Collector
 4. OPCHDA Collector
 5. OSI Pi Collector
 6. OSI Pi Distributor
 7. IFix Collector
 8. Server To Server Collector
 9. MQTT Collector
 10. Calculation Collector
 11. Server To Server Distributor
 12. Wonderware Collector
 13. ODBC Collector

  Option [1-13]:8

  Provide the Interface Name of the Collector:  S2S

  Do you want to enable User Password masking on the Console  ? [Y/N] (Default 'N') :N

  Provide the Destination Historian details...
  Destination Historian (NLB DNS for Cloud Historian/ Server Name for Native Historian) : NLB1

  Provide Historian UserName If Exists (or) Press Enter : User1

  Provide Historian Password If Exists(or ) Press Enter :


  Provide the Source Historian Server[EC2AMAZ-1L57O0A]: NLB2

  Source Historian Username (Press Enter if User is not defined): user2

  Source Historian Password(Press Enter If Password is not defined) :
```

Create a Server-to-Server Distributor

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).

The Historian Server-to-Server distributor is used to send data from a smaller Historian server to a larger, centralized Historian server. You can then use this data for reporting and analytics.

You can use either the Server-to-Server collector or the Server-to-Server distributor to send data to a central Historian. However, using the Server-to-Server distributor has the following advantages:

- It simplifies the process of configuring tags at the destination Historian.
- It provides more flexibility at the SCADA level for tag configuration compared to the Server-to-Server collector.
- It allows you to manage tags both from the source and destination Historian servers, whereas the Server-to-Server collector allows you to manage tags only from the destination Historian server.

> **Important:**
> If you want to send data from cloud to on-premises, you must disable TLS encryption by setting the following registry entry to 0: `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE Digital\iHistorian\Services\ServerToServerCollector\<collector instance>\TLSEnable`

Features

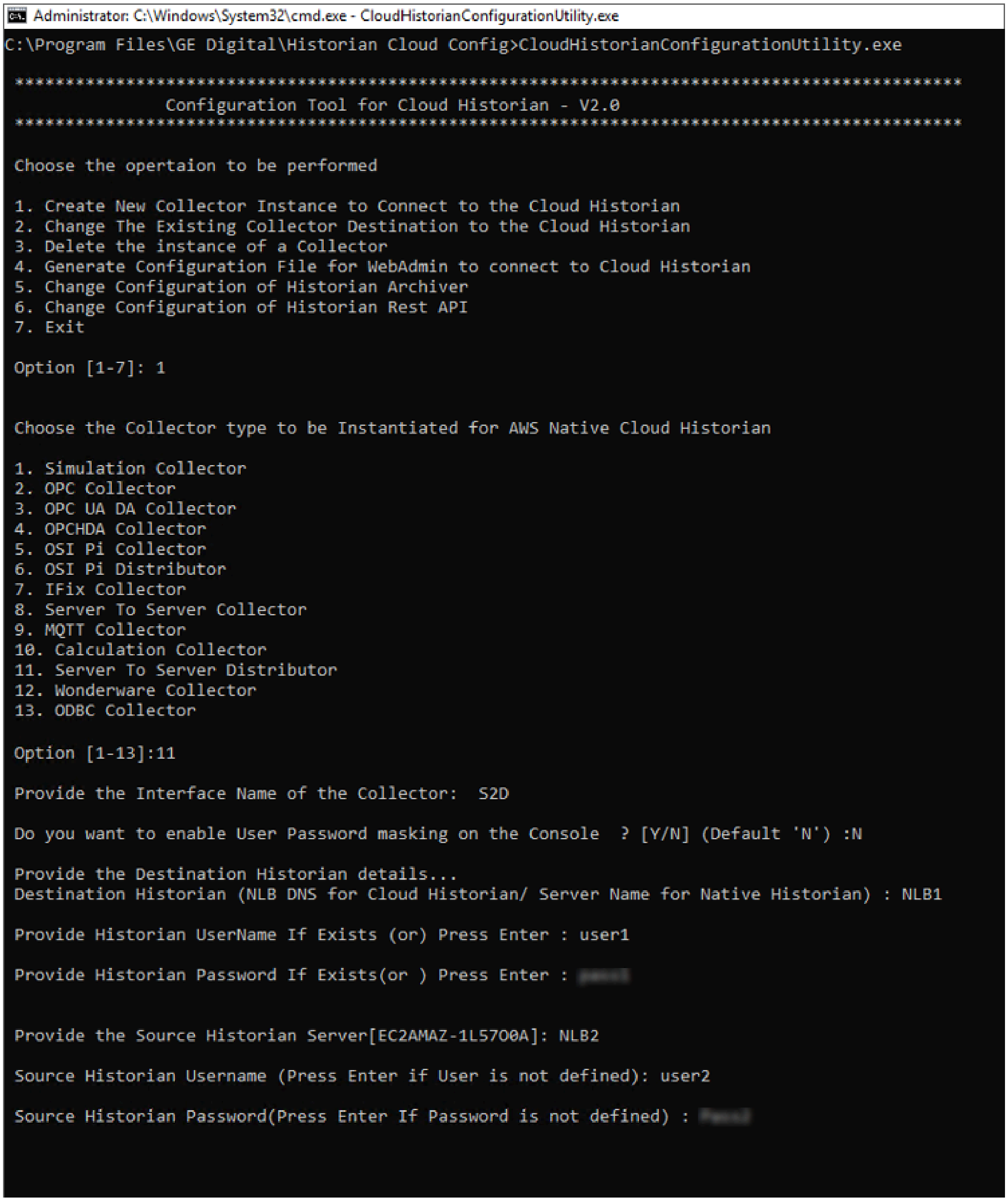
- You can browse the source for tags and their attributes.
- Only the unsolicited data collection is supported; polled collection is not supported.
- The supported timestamp resolution is 100 milliseconds.
- The collector accepts device timestamps.
- The collector supports data compression.
- Floating point, integer, and string data are supported.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.

4. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
User Password Masking	Enter **Y** if you want to mask the password, or **N** if you want to proceed without masking the password.
Destination Historian Server	Enter the host name or IP address where the destination Historian server is installed.
Destination Historian Username	Enter the username to connect to the destination server.
Destination Historian Password	Enter the password to connect to the destination server.
Source Server	Enter the host name or IP address of the source server.
Username	Enter the username to connect to the source server. If you want to send data to an on-premises Historian server, press ENTER.
Password	Enter the password for the user you want to connect to Proficy Historian for Azure Cloud. If you want to send data to an on-premises Historian server, press ENTER. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.

The Server-to-Server collector is created and started.



Create a Simulation Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on a VNet (which can be different from the one on which Proficy Historian for Azure Cloud is deployed).

The Simulation collector generates random numbers and string patterns for demonstration purposes. You can configure the number of tags that you want to generate.

Features:

- The collector generates random scaled values between 0 and 32,767. It uses the high and low engineering units fields of each tag to scale the 0 to 32,767 pre-set values into appropriate engineering units.
- The collector also provides five-string simulation tags that generate random alphanumeric data.
- In addition to generating random values, the collector can generate sequential values for some tags. For a list of such tags, refer to Tags with Sequential Values.
- You can import browse for tags and their attributes.
- The supported timestamp resolution is 1ms.
- Floating point, integer, and string data are supported. Binary data is not supported.
- You can create Python Expression tags.
- Only polled data collection is supported with a minimum poll interval of 100ms.

> **Note:**
> You can create more simulation string tags by manually adding string tags with the following naming convention to the collector: `CollectorName.Simulation.StringXXXX`

Supported Tag Attributes:

- Tagname
- Data Type
- Hi Engineering Units
- Lo Engineering Units
- Hi Scale
- Lo Scale

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
4. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.

The collector instance is created and started.

Create a Wonderware Collector

1. Deploy Proficy Historian for Azure Cloud. *(on page 17)*
2. Install collectors *(on page 29)*. You can install them on-premises or on an Azure Virtual Machine in a VNet (which can be the same one as the Historian server or a different one).
3. Ensure that the iFIX server is running.

The Wonderware collector collects data from a Wonderware Historian 2014 R2 Server application and stores it in Data Archiver.

Features:

- Only unsolicited data collection is supported.
- The supported timestamp resolution is 100 milliseconds.
- Floating point, integer, and string data are supported.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to creating a collector instance.
   A list of collectors that you can create appears.
4. Enter the number corresponding to the collector that you want to create.
5. Enter the following details:

Field	Description
Interface Name	Enter the name that you want to provide for the collector instance.
Azure Load Balancer IP	Enter the Azure Load Balancer IP. A value is required. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment.

Field	Description
	c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud. A value is required.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. A value is required. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.
Wonderware Server	Enter the host name or IP address of the Wonderware server. A value is required. By default, the local host name is considered.

The Wonderware collector is created and started.

Change the Destination of a Collector

You can change the destination of a collector from Proficy Historian for Azure Cloud deployed on one Azure Virtual Machine to another.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to changing the destination of a collector.
4. Enter the number corresponding to the collector instance whose destination you want to change.
5. Enter the following details:

Field	Description
Interface name	Enter the interface name of the collector instance whose destination you want to change.

Field	Description
Azure Load Balancer IP	Enter the Azure Load Balancer IP of the *destination* Azure Virtual Machine. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Username	Enter the username to connect to Proficy Historian for Azure Cloud.
Password	Enter the password to connect to Proficy Historian for Azure Cloud. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field at the time of deployment.

The destination of the collector instance is changed.

Delete a Collector Instance

If you no longer want to use a collector instance to collect data, you can delete it. When you delete a collector instance, the Windows service for the collector, the Registry folder, and the buffer files are deleted as well.

Note:
When you delete a collector instance, be sure to delete the instance from the Historian Administrator client.

1. Run Command Prompt as an administrator.
2. Run the `AzureCloudHistorianConfigurationUtility.exe` file. It is provided along with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital\Historian Cloud Config` folder by default.
3. Enter the number corresponding to deleting a collector instance.
   A list of collector types appears.
4. Enter the number corresponding to the collector type whose instance you want to delete.
5. Enter the interface name of the collector instance that you want to delete.

   > **Tip:**
   > To find the interface name of a collector, access Windows services; each collector instance runs as a service.

   The collector instance is deleted.

Chapter 5. Using the Web Admin Console

About the Web Admin Console

The Web Admin console is a web-based user interface, which you can use to monitor, supervise, archive, retrieve, and control data stored in the Historian server. It extends the functionality of Proficy Historian for Azure Cloud. It also contains Configuration Manager.

Using the Web Admin console, you can:

- Maintain and configure the Historian System.
- Retrieve and analyze archived information.
- Set up and maintain configuration and other parameters for tags, collectors, and archives.
- Perform specific supervisory and security tasks for the Historian system.

Access the Web Admin Console

1. Access the following URL: `https://<IP address>:9443/historian-visualization/hwa`, where *<IP address>* is the public IP address of the Azure Virtual Machine on which you have installed the Web Admin console.
2. Enter the password for the user. For ihCloudHistAdmin, enter the value that you provided at the time of deployment.
   The Web Admin console appears.

About Data Stores

A data store is logical collection of tags. It is used to store, organize, and manage tags according to the data source and storage requirements. A data store can have multiple archive files (*.IHA), and includes both logical and physical storage definitions.

Tags can be segregated into separate archives through the use of data stores. The primary use of data stores is to segregate tags by data collection intervals. For example, you can put a name plate or static tags where the value rarely changes into one data store, and your process tags into another data store. This can improve query performance.

Historian data stores are stored as archive files that contain data gathered from all data sources during a specific period of time. You can write and read data from the archive files.

You can define two types of data stores:

- **Historical Data Store**: Tags stored under historical data store will store data as long as the disk space is available. Depending on your license, you may be able to create multiple historical data stores. The maximum number of historical data stores supported depends on the license.
- **SCADA Buffer Data Store**: Tags stored under the SCADA buffer data store will store data for a specific duration of time based on license.

When you install the Historian server, two historical data stores are installed by default.

- **System**: Stores performance tags. This is only for internal usage within Historian, and you cannot add tags to this data store. You must not rename or delete the system data store.
- **User**: Stores tag data. This is a default data store. You can rename and delete a user data store as long as there is another default data store set for tag addition.

Based on your license, a SCADA Buffer data store may also be installed. It stores short-term tags and data.

Create a Data Store

The number of data stores that you can create depends on your license.

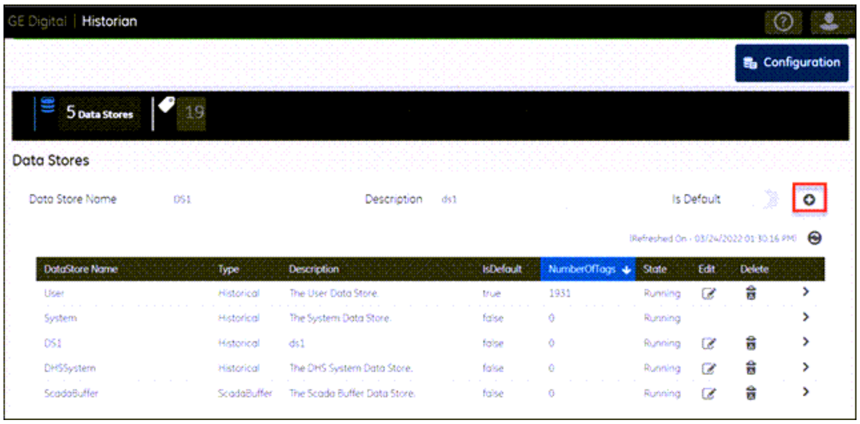
1. Access the Web Admin console *(on page 90)*.
2. Select **Data Stores**.

   A list of data stores appears.
3. Enter values as described in the following table.

Field	Description
Data Store Name	Enter a unique name for the data store. A value is required. You can use all alphanumeric characters and special characters except / \ * ? < > |

Field	Description
Description	Enter a description for the data store.
Is Default	Switch the toggle on if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default.

4. Select .



The data store is created.

When you add tags to the data store, it will have its own set of .IHA (iHistorian Archive) files. Ensure that you back up the new data store archives periodically using AFS.

Access a Data Store

1. Access the Web Admin console *(on page 90)*.
2. Select **Data Stores**.

A list of data stores appears. By default, the list is refreshed every 10 minutes. You can, however, refresh it manually by selecting .

3. In the row containing the data store that you want to access, select .
The details of the data store appear, displaying the following information:

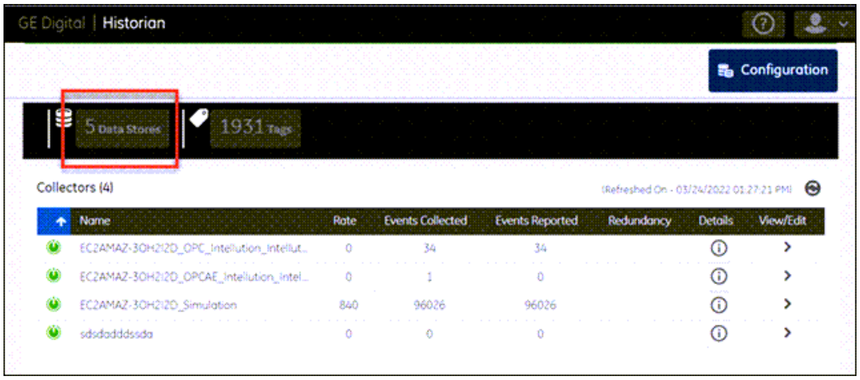
Field	Description
Archive Compression	Displays the current effect of archive data compression. If the value is zero, it indicates that archive compression is either ineffective or turned off. To increase the effect of data compression, increase the value of archive compression deadbands on individual tags. In calculating the effect of archive compression, Historian counts internal system tags as well as data source tags. Therefore, when working with a very small number of tags and with compression disabled on data source tags, this field may indicate a value other than zero. If you use a realistic number of tags, however, system tags will constitute a very small percentage of total tags and will therefore not cause a significant error in calculating the effect of archive compression on the total system.
Free Space	Not applicable
Consumption Rate	Displays how fast the archive disk space is consumed. If the value is too high, you can reduce it by slowing the poll rate on selected tags or data points or by increasing the filtering on the data (widening the compression deadband to increase compression).
Write Cache Hits	Displays the hit ratio of the write cache in percentage of total writes. It is a measure of how efficiently the system is collecting data. Typically, this value should range from 95 to 99.99%. If the data is changing rapidly over a wide range, however, the hit percentage drops significantly because current values differ from recently cached values. More regular sampling may increase the hit percentage. Out-of-order data also reduces the hit ratio.

Field	Description
Estimated Days to fill	Not applicable
Failed Writes	Displays the number of samples that failed to be written. Since failed writes are a measure of system malfunctions or an indication of offline archive problems, this value should be zero. If you observe a non-zero value, investigate the cause of the problem and take corrective action. Historian also generates a message if a write fails. Note that the message only appears once per tag, for a succession of failed writes associated with that tag. For example, if the number displayed in this field is 20, but they all pertain to one Historian tag, you will only receive one message until that Historian tag is functional again.
Alerts Since Startup	Not applicable
Receive Rate	Displays how busy the server is at a given instance and the rate at which the server is receiving data from collectors.
Messages Since Startup	Not applicable
Number of Archives	The number of archives in the data store.
Number of Tags	The number of tags in the data store.

Modify a Data Store

You can change the name and description of a data store. You can also set a data store as default. In addition, you can configure the default settings of archives.

1. Access the Web Admin console *(on page 90)*.
2. Select **Data Stores**.



A list of data stores appears.

3. In the **Edit** column, select .

   The **Edit Data Store Configuration** window appears.

4. Modify values as described in the following table.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Data Store Name</td><td>Enter a unique name for the data store. A value is required. You can use all alphanumeric characters and special characters except / \ * ? < > |

Note:
You cannot rename the system data store.</td></tr>
<tr><td>Description</td><td>Enter a description for the data store.</td></tr>
<tr><td>Is Default</td><td>Switch the toggle on if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default.</td></tr>
</table>

5. Select **Save**.

   The data store is modified.

6. If you want to modify the default settings of archives in the data store, in the row containing the data store, select .

   The details of the data store appear.

7. Select **Configuration**, and then modify values as described in the following table.

Field	Description
Default Archive Path	The default path of the folder in which you want to create the archives for the data store. **Note:** We recommend not to use a period in the default archive path field. If you do so, you will not be able to specify a default archive name.
Base Archive Name	A prefix that you want to add to all the archive files.
Use Caching	Indicates whether caching must be enabled. When reading data from the archiver, some data is saved in the system memory and retrieved using caching. This results in faster retrieval as the data is already stored in the buffer. This option is not available for SCADA Buffer data stores.
Default Backup Path	Not applicable. Use AFS to back up and restore archives.
Free Space Required (MB)	Indicates the remaining disk space required after a new archive is created. If the available space is less than the requirement, a new archive is not created.
Generate Message on Data	If this option is enabled, an audit log entry will be made any time the value of a previously archived data point in the Historian archive is overwritten. This log entry will contain both the original and new values. If you want to create multiple archives at the same time, disable this option. This option is not available for SCADA buffer data stores.
Archive Duration	The duration of a newly created archive in days or hours. A new archive will be created after the selected number of days or hours.
Automatically Create Archives	If enabled, the server automatically creates a new archive in the default path directory whenever the current archive is full. If dis-

Field	Description
	abled, no new data will be written to the archives after the current archive is full.
Store OPC Quality	Stores the OPC data quality. To create multiple archives at the same time, **Store OPC Quality** must be *Disabled*.
Data Read Only After (hrs)	The number of hours, prior to now, for which data can be stored in a read/write archive. After the time expires, that portion of the archive file is automatically made read-only. Incoming data values with timestamps prior to this time are rejected. A single archive file, therefore, may contain a read-only section, another read-write section containing recently written data, and unused free space.
Overwrite Old Archives	When enabled, the system replaces the oldest archived data with new data when the default size has been reached. ◦ To create multiple archives at the same time, disable this option. ◦ If you enable both **Automatically Create Archives** and **Overwrite Old Archives**, you must set `ihArchiveFreeSpaceHardLimit` to TRUE using APIs.
Stale Period	Specifies the duration in days after which tags are considered stale for this data store. Valid values are: ◦ 0: If you enter this value, tags are never considered stale. ◦ 7 to 36500 (100 years) Stale tags are tags that have no new data samples within a specified period of time. These tags add to system overhead and slow down user queries.
Stale Period Check	Specifies the frequency in days with which the staleness of the tag is checked. You can enter a value between 1 and 30. Stale tags are tags that have no new data samples within a specified period of time. These tags add to system overhead and slow down user queries.

8. Select **Update**.

   The archive settings of the data store are modified.

Delete Data Store

You can delete a data store if it is no longer needed. You must not delete the system data store. You cannot delete the last user data store; at least one user data store must exist.

1. Access the Web Admin console *(on page 90)*.
2. Select **Data Stores**.

   A list of data stores appears.
3. In the **Delete** column, select .

   A message appears, asking you to confirm that you want to delete the data store.
4. Select **Yes**.

   The data store is deleted.

About Tags

A Historian tag is used to store data related to a property.

For example, if you want to store the pressure, temperature, and other operating conditions of a boiler, a tag will be created for each one in Historian.

When you collect data using a collector, tags are created automatically in Historian to store these values. These tags are mapped with the corresponding properties in the source.

For example, suppose you want to store OSI PI data in Historian. You will specify the OSI PI tags for which you want to collect data. The OSI PI collector creates the corresponding tags in Historian, and it stores the values in those tags.

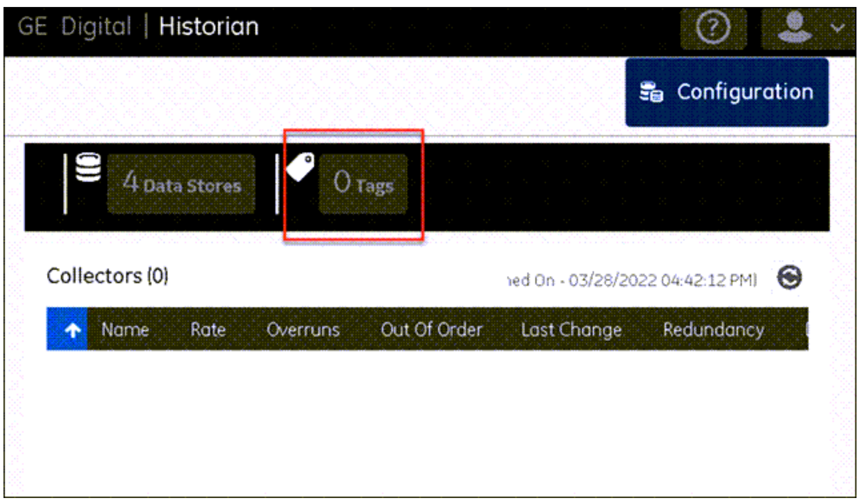
You can also choose to create tags manually.

Add a Tag Manually

If you want to associate tags with a collector, install collectors *(on page 29)*, and then create a collector instance *(on page 44)*.

Typically, when you browse a data source for tags, you can choose to add those tags to Historian *(on page 100)*. The corresponding tags are then created in Historian. However, you can also add tags manually (without browsing from a data source). You can choose to associate these tags with a collector, which then collects data for the tag.

1. Access the Web Admin console *(on page 90)*.
2. Select **Tags**.



   A list of tags appears.

3. Select , and then select **Add Tag**.

The **Add Tag** window appears.

4. Enter values as described in the following table.

Field	Description
Collector Name	Select the collector instance using which you want to collect the tag data.
Source Address	Enter the host name or IP address of the data source from which you want to collect the tag data.
Tag Name	Enter a tag name. A value is required and must be unique.
Data Store	Select the data store in which you want to store the tag data. By default, the default data store is selected.
Data Type	Enter the data type of the tag.
Time Resolution	Enter the time resolution of the tag. For example, if you select **Seconds**, tag data is stored every second.

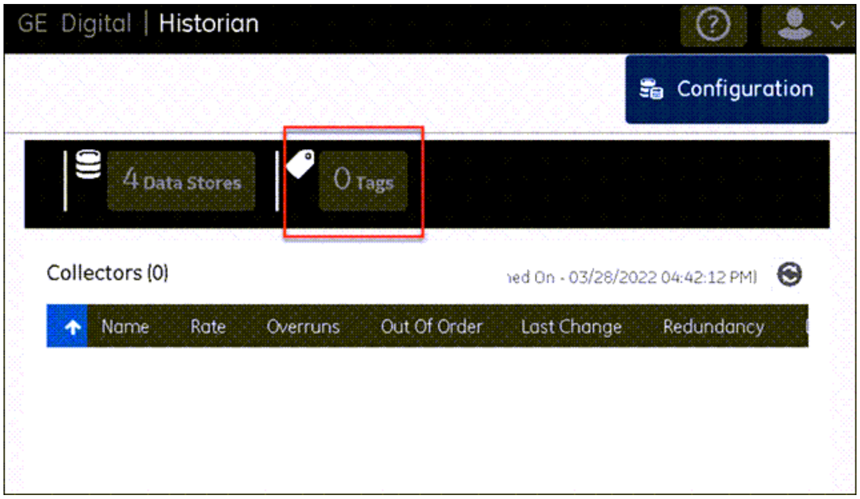
5. Select **Add**.

   The tag is added.

Add a Tag from a Collector

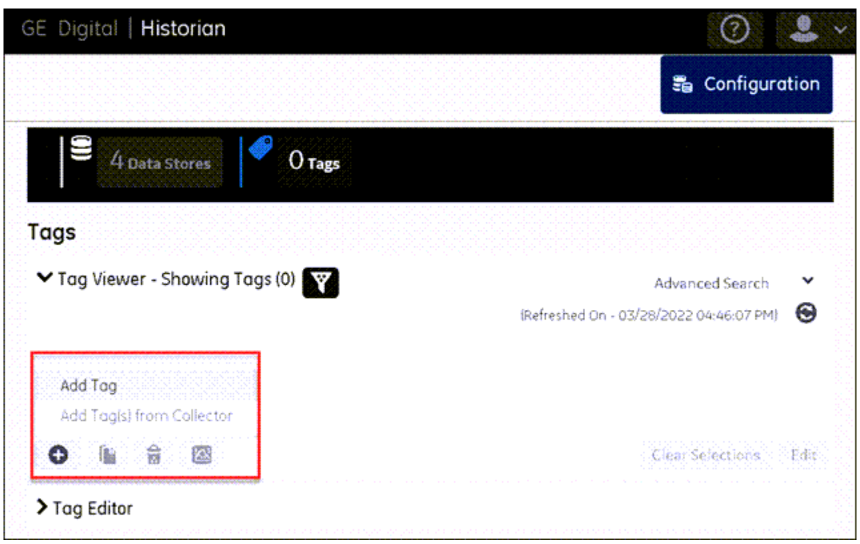
When you add tags from a collector, the collector browses the data source for a list of tags. You can select the ones that you want to add to Historian. These tags are then created in the Historian database. You can also create tags manually *(on page 99)*.

1. Access the Web Admin console *(on page 90)*.
2. Select **Tags**.



A list of tags appears.

3. Select , and then select **Add Tag(s) from Collector**.



The **Add Tag(s) from Collector** window appears.

4. Enter values as described in the following table.

Field	Description
Collector	Select the collector using which you want to browse the source.
Source Tag Name	Specify the names of tags that you want to add.

A list of tags that match the criteria appear.

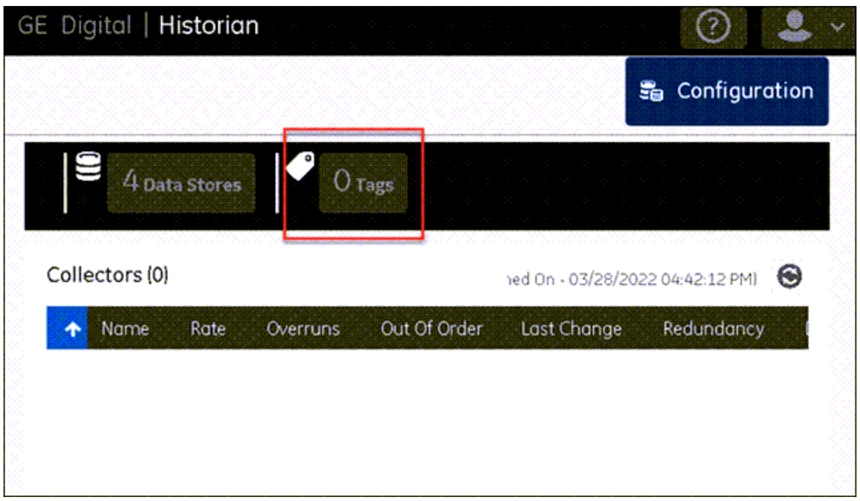
5. Select **Add** or **Add All**.

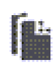
   The selected tags appear in the right section.
6. Select **Add Selected Tags**.

   The tags are added.

Copy a Tag

1. Access the Web Admin console *(on page 90)*.
2. Select **Tags**.

   A list of tags appears.
3. Select the tag that you want to copy, and then select .

   The **Copy Tag** window appears.
4. In the **New Tag Name** field, enter a unique name for the tag, and then select **Ok**.

Access Tag Values

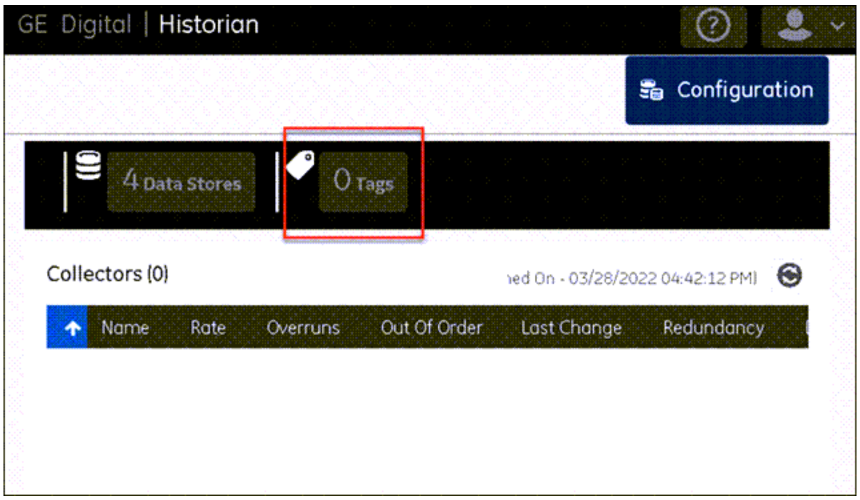
You can access tag values in any of the following formats:

- **Trend chart:** Plots the trend chart of the tag values interpolated in the last 10 minutes. You can choose among a line, column, and an area graph. You can use a trend chart to compare the values of multiple tags.
- **Last 10 raw values:** Displays a list of the last 10 raw values of tags. The values of each tag are displayed separately.

For either of these options, you can select up to 10 tags.

The difference in the timestamp of consecutive values depends on the time resolution of the tag. For example, if the time resolution is seconds, the timestamp of consecutive values of the tag will be one second apart.

1. Access the Web Admin console *(on page 90)*.
2. Select **Tags**.



A list of tags appears.

> **Tip:**
> You can filter the list of tags by selecting or **Advanced Search**.

3. Select the tags whose data you want to access/plot. You can select up to 10 tags at a time.
4. Select , and then select one of the following options:
   - **Trend**: Select this option if you want to plot a trend chart of the tag values. You can choose among a line graph, a bar graph, and an area graph. The data is interpolated for the last 10 minutes.
   - **Last 10 Raw Values**: Select this option if you want to view the last 10 values of each tag.

   The trend chart or the last 10 values of the selected tags appear.

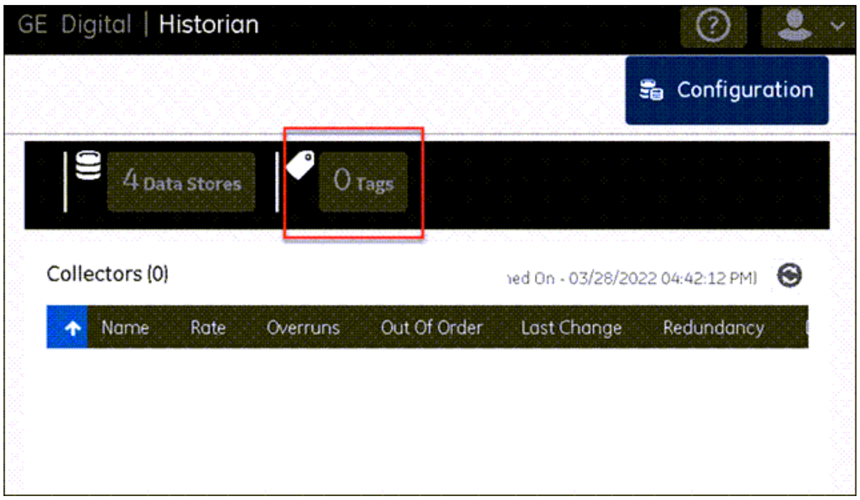
Delete a Tag

When you delete a tag, you can choose between the following options:

- **Remove a tag from the system:** When you remove a tag from a system, the tag and its data will still be available. Therefore, you cannot create a tag with the same name.
- **Delete a tag permanently:** When you delete a tag, it is deleted from Historian, and the tag data will no longer be available.

You can delete multiple tags at a time.

1. Access the Web Admin console *(on page 90)*.
2. Select **Tags**.



   A list of tags appears.

3. Select the tag that you want to delete, and then select .
   A message appears, asking you to choose between the following options:
   - **Remove a tag from the system:** When you remove a tag from a system, the tag and its data will still be available. Therefore, you cannot create a tag with the same name.
   - **Delete a tag permanently:** When you delete a tag, it is deleted from Historian, and the tag data will no longer be available.
4. Select the appropriate option, and then select **Ok**.
   A message appears, asking you to confirm that you want to remove/delete the tag.
5. Select **Yes**.
   The tag is removed/deleted.

Chapter 6. Using the REST APIs

Overview of the Historian REST APIs

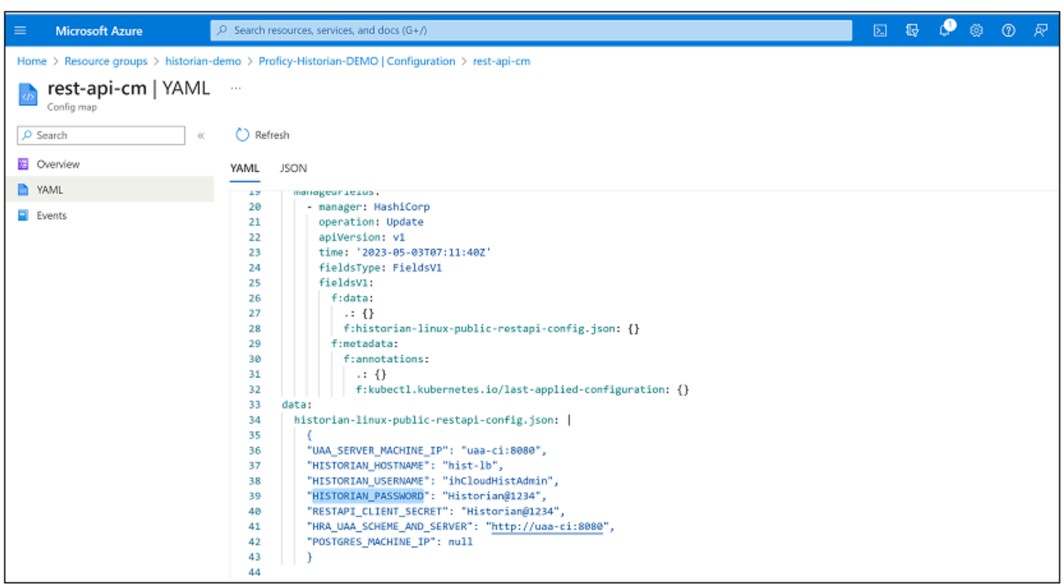
Introduction

The REST APIs are used to fetch data from the Historian database. You can fetch data such as the latest data point of a tag or data points for a duration. Using these APIs, you can also work on aggregation techniques like average, minimum, and maximum values.

Modify REST API User on Azure

To modify the REST API on Azure, follow these steps:

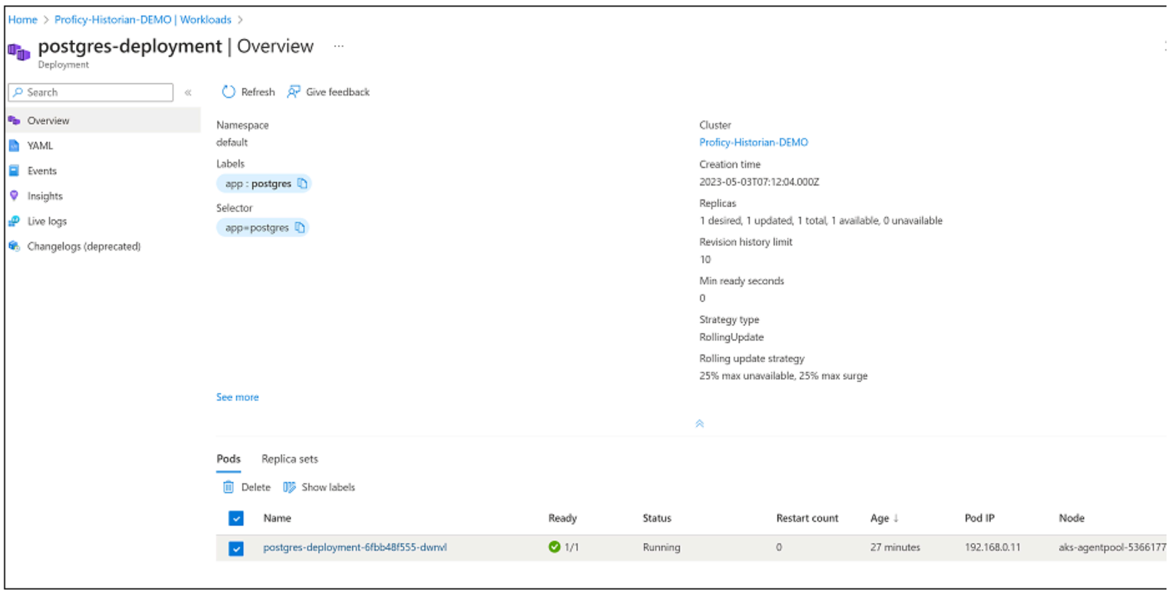
1. Login to Azure portal.
2. Go to the resource group in which CloudHistorian is deployed.
3. Go to <cluster-name> Kubernetes service -> Configuration.
4. Under the Config maps section, filter by **default** namespace.
5. Select **rest-api-cm** and then click on YAML and scroll down to the data section.
6. Modify user name under **HISTORIAN_USERNAME** field and password under **HISTORIAN_PASSWORD** field.
7. Click **Review** and **Save**.



8. Go to <cluster-name> Kubernetes service -> Workloads -> Filter by default namespace.

9. Select **rest-api-deployment** under deployment section.
10. Under Pods section select **rest-api-deployment-xxxxxx**.
11. Click **Delete**.

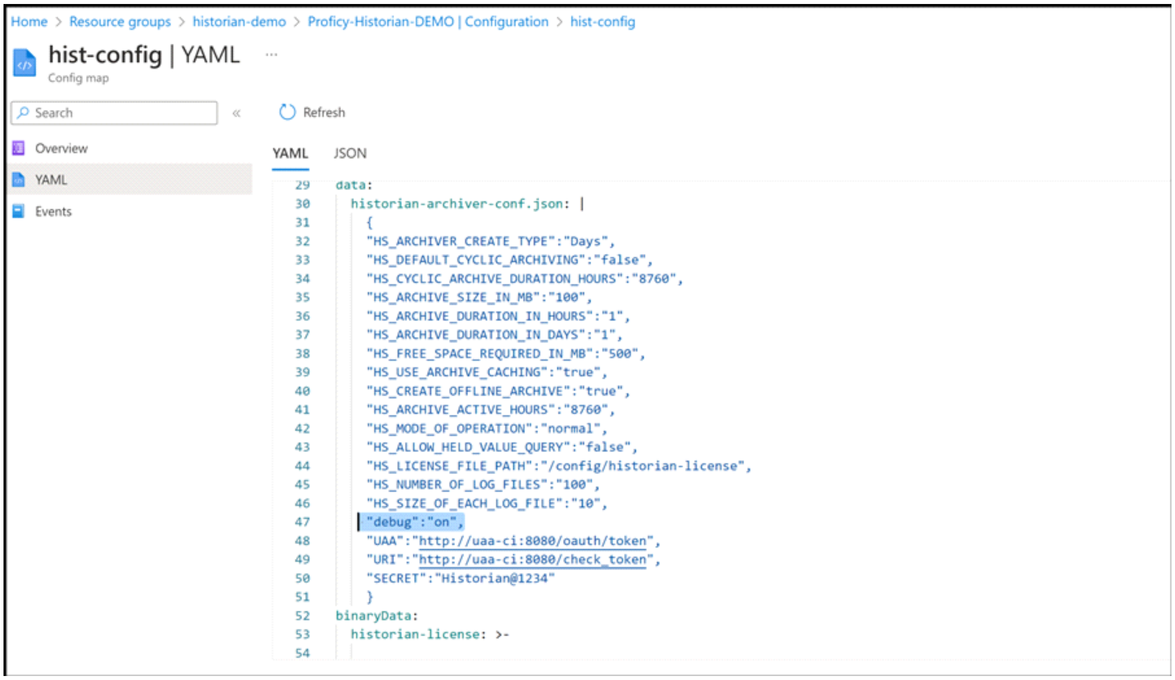
    This will restart rest API pod with new changes.



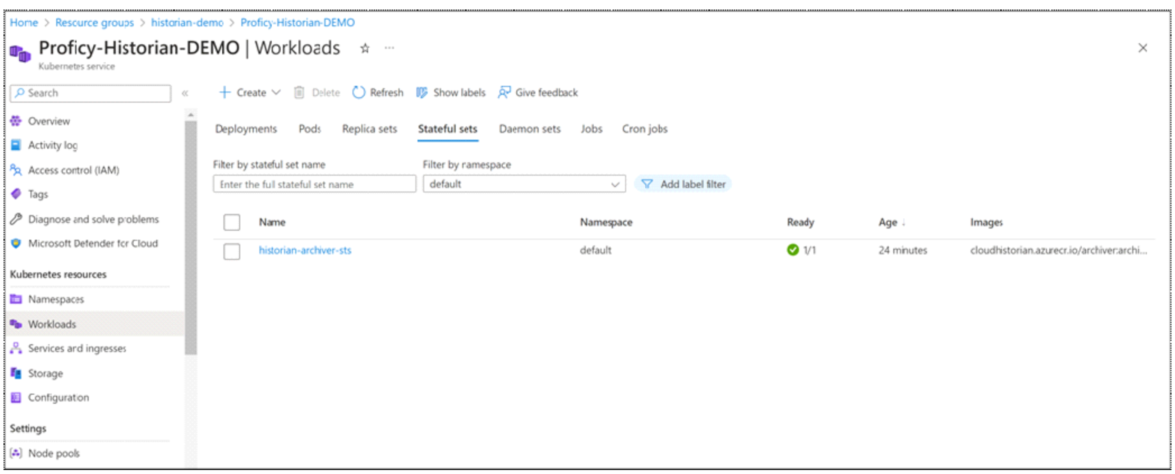
Change Debug mode of Data Archiver

To change the debug mode for the Data Archiver, follow these steps:

1. Login to Azure portal.
2. Go to the resource group in which CloudHistorian is deployed.
3. Go to <cluster-name> Kubernetes service -> Configuration.
4. Under the Config maps section, filter by **default** namespace.
5. Select **hist-config**, and then click on YAML and scroll down to the data section.
6. Modify **debug** field to on / off .
7. Click **Review** and **Save**.



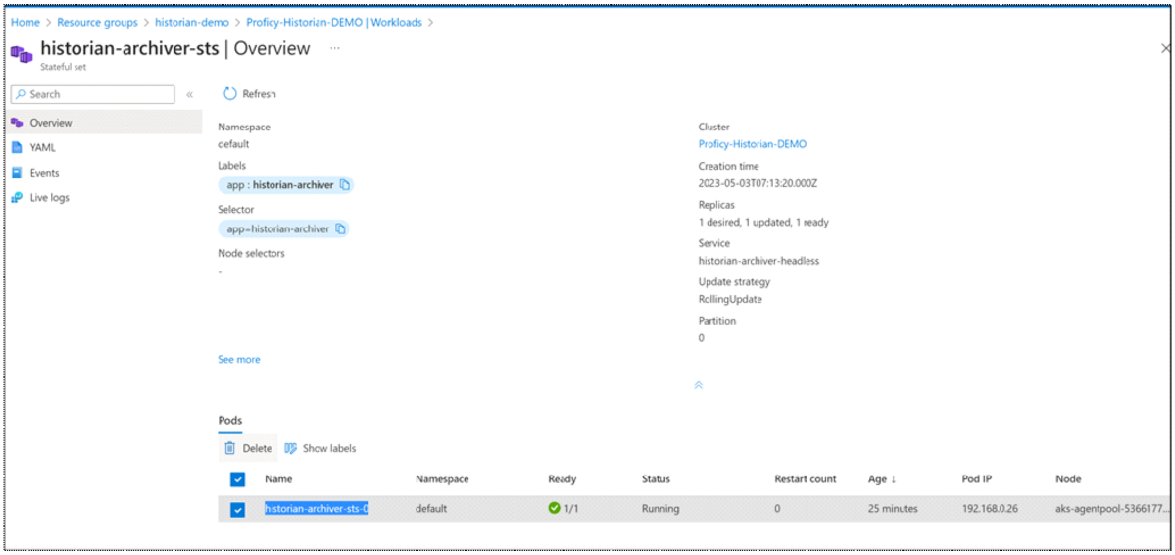
8. Go to <cluster-name> Kubernetes service.
9. Select Workloads and choose the Filter **default** namespace.
10. Under the Stateful sets section, select **historian-archiver-sts**.



11. Under Pods section select **historian-archiver-sts-0**.

12. Click **Delete**.

This action will restart Data Archiver pod with new changes.



Get an Authorization Token

1. Deploy Proficy Historian for Azure Cloud *(on page 17)*.
2. Install an API platform (such as Postman).

To connect the Historian server with the REST APIs, you must get an authorization token.

1. Access an API platform, and request for an authorization token.
2. Enter values as described in the following table and submit.

> **Note:**
> The names of these fields may vary depending on the API platform that you are using.

Field	Description
Token name	Provide a name for the token.
Grant type	Select Client Credentials.
Access token URL	Enter a value in the following format: `https://<Azure Load Balancer IP>:8080/oauth/token`

Field	Description
	Tip: To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Client ID	Enter `historian_rest_api`.
Client secret	Enter the value you entered in the **Password** field at the time of deployment.

The token is generated. You can use this token to use REST APIs.

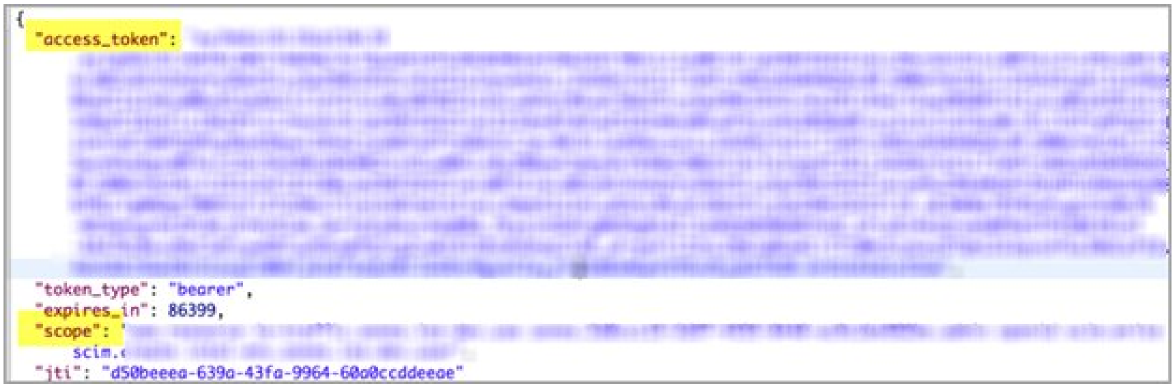
3. Use the following command to authenticate the REST API authentication with password credentials:

```
curl -d
 "client_id=<value>&client_secret=<value>&grant_type=password&username=<value>&password=<value>&token_format=op
aque&response_type=token" https://Azure-Load-Balancer-IP:8080/oauth/token
```

For example:

```
curl -d
 "client_id=server1.admin&client_secret=adminsecret&grant_type=password&username=user1&password=pwd123&token_fo
rmat=opaque&response_type=token" https://Azure-Load-Balancer-IP:8080/oauth/token
```

The following image shows an example of OAuth access. The actual token text is blurred for security concerns.



Client applications can access data using service the REST API endpoints. Your application makes an HTTP request and parses the response. You can use any web-development language to access the APIs.

Common API Parameters

Overview of Commonly Used API Parameters

The Historian REST service provides various REST API calls to retrieve the current tags list and query data with different sampling modes. Most of these API calls use the following common parameters:

- tagNames *(on page 110)*
- Start and End timestamps *(on page 111)*
- TagSamples *(on page 111)*
- DataSamples *(on page 113)*
- SamplingModeType *(on page 114)*
- Direction *(on page 116)*
- CalculationModeType *(on page 116)*
- FilterModeType *(on page 122)*
- ReturnDataFields *(on page 123)*
- Payload *(on page 124)*
- Error Code Definitions *(on page 130)*

TagNames Parameter

By default, the Historian REST service provides support to read samples for multiple tags. Multiple tag names are separated by semicolons (;). For example, "tagname1;tagname2;tagname3".

```
https://<historianservername>:9090/historian-rest-api

/v1/datapoints/currentvalue?tagNames=tagName1;tagname2;tagname3
```

Encode the semicolon as `%3B` if using the URI format, as the semicolon is also a valid character for a Historian name, and the web service parses the tag names incorrectly if a tag name contains a semicolon.

Start and End Timestamps Parameter

For the Start and End Timestamps parameter, the Timestamp format in the URI must be in ISO data format, such as `YYYY-MM-DDTHH:mm:ss.SSSZ`.

EPOCH time (standard base time) is only valid in the JSON-format request body or response body, such as `\/Date(928167600000-0500)\/`. If you use the same timestamp for start and end timestamps, the request returns a single result.

All timestamps passed to the REST service must be formatted as UTC timestamps.

Object Name	Description
StartTime	Start time of the query. This represents the earliest timestamp for any tag contained in the query. If no StartTime is specified, the start time is two hours prior to running the query.
EndTime	End time of the query. This represents the latest timestamp for any tag contained in the query. If no EndTime is specified, the end time is the time that the query runs.

TagSamples Parameter

The TagSamples parameter is the output from the REST API calls.

Property Name	Property Type	Description
TagName	String	Name of the tag.
DataType	String	**Tag Data Type Value:**

Property Name	Property Type	Description
		• **Blob** – Stores tags as binary large objects. The Blob datatype generally refers to undetermined binary data types, such as an Excel spreadsheet, a PDF file, or a Word file. • **Boolean** (one byte) – Stores boolean values. Valid values for the boolean data type are 0=FALSE and 1=TRUE. If the user sends zero, the value is taken as zero. Anything other than zero, the value is treated as one. • **Byte** (one byte) – Stores integer values. Valid values for the byte data type are -128 to +127. • **SingleFloat** (four bytes) – Stores decimal values up to six places. Valid ranges for the single float data type are 1.175494351e-38F to 3.402823466e+38F • **DoubleFloat** (eight bytes) – Stores decimal values up to 15 places. Valid values for the double float data type are 2.2250738585072014e-308 to 1.7976931348623158e+308. • **SingleInteger** (two bytes) – Stores whole numbers, without decimal places. Valid values for the single integer data type are -32767 to +32767. • **DoubleInteger** (four bytes) – Stores whole numbers, without decimal places. Valid values for the double integer data type are -2147483648 to +2147483648. • **FixedString** (Configured by user) – Stores string data of a fixed size. Valid values are between 0 and 255 bytes. • **Float** – Single float. • **Integer** – Single integer. • **MultiField** – Stores string data that has multiple words. • **QuadInteger** (eight bytes) – Stores whole numbers without decimal places. Valid values for the quad integer data type are -9,223,372,036,854,775,808 (negative nine quintillion) to +9,223,372,036,854,775,807 (positive nine quintillion). • **Scaled** (two bytes) – Lets you store a four-byte float as a twobyte integer in the Historian archive. The scaled data type saves disk space but sacrifices data precision as a result. • **Time** – Returns or sets the type of time stamping applied to data at collection time.

Property Name	Property Type	Description
		• **UDoubleInteger (Unsigned Double Integer)** (four bytes) – Stores whole numbers without decimal places. Valid values for the unsigned double integer data type are 0 to 4,294,967, 295 (4.2 billion). • **Undefined** – Data type is not defined. • **UQuadInteger (Unsigned Quad Integer)** (eight bytes) – Stores whole numbers without decimal places. Valid values for the unsigned quad integer data type are 0 to 18,446,744,073,709,551,615 (19 quintillion). • **USingleInteger (Unsigned Single Integer)** (two bytes) – Stores whole numbers without decimal places. Valid values for the unsigned single integer data type are 0 to 65535. • **VariableString** (No fixed size) – Stores string values of undetermined size. This data type is useful if you cannot rely on a constant string length from your data source. • **Array** – Returns an array of tags from your data source. You can specify orientation, size, and number of rows returned in the array.
ErrorCode	Error Code	Error Code Definition See Error Code Definition *(on page 130)* for more information.
Samples	Data Sample	See DataSample Parameter *(on page 113)* for more information.

DataSample Parameter

The DataSample Parameter specifies the number of data samples to retrieve from the archive. Samples are evenly spaced within the time range defined by start time and end time for most sampling modes.

Property Name	Property Type	Description
Value	String	Format for a multi-field tag like `{ "field1":"1","field2":"1000.0" }` (user-defined type tag).

Property Name	Property Type	Description
		JavaScript code can parse the value string as a JSON object. All field values are string.
TimeStamp	DateTime	Start and end times of the query. If no start time is specified, the start time is two hours prior to running the query. If no EndTime is specified, the end time is the time the query runs.
Quality	Integer (Enumerated value of DataQuality.StatusType)	Data type consisting of a set of named values called elements, members or enumerators of the type. Property values reflect quality as "quality is good" or " quality is bad". **Value and Status** • **0** – Bad • **1** – Uncertain • **2** – NA • **3** – Good

SamplingModeType Parameter

The SamplingModeType parameter is the mode of sampling data from the archive. The default setting for the Sampling Mode is `Calculated`.

Properties	Description	Value
Undefined	Sampling mode is not defined.	0
CurrentValue	Retrieves the current value. The time- interval criteria are ignored.	1
Interpolated	Retrieves evenly-spaced, interpolated values based on interval or NumberOfSamples and the time-frame criteria.	2
Trend	Returns the raw minimum and raw maximum value for each specified interval.	3

Properties	Description	Value
	Use the Trend sampling mode to maximize performance when retrieving data points for plotting. For the Trend sampling mode, if the sampling interval does not evenly divide by the interval length, Historian ignores any leftover values at the end, rather than putting them into a smaller interval.	
RawByTime	Retrieves raw archive values based on time-frame criteria.	4
RawByNumber	Retrieves raw archive values based on the StartTime criteria, the NumberOfSamples, and Direction criteria. The EndTime criteria is ignored for this sampling mode.	5
Calculated	Retrieves evenly spaced calculated values based on NumberOfSamples, interval, the time frame criteria, and the CalculationMode criteria.	6
Lab	Returns actual collected values without interpolation.	7
InterpolatedtoRaw	Provides raw data in place of interpolated data when the number of samples are fewer than the available samples.	8
TrendtoRaw	The TrendtoRaw sampling mode almost always produces the same results as the Trend sampling mode. However, when more samples are requested than there are raw data points, the TrendtoRaw sampling mode returns all available raw data points with no further processing. Use TrendtoRaw in place of Trend when this condition exists.	9

Properties	Description	Value
LabtoRaw	Provides raw data for the selected calculated data, when NumberOfSamples is less than the available samples.	10
RawByFilterToggle	Returns filtered time ranges using the following values: • 1 – true • 0 – false This sampling mode is used with the time range and filter tag conditions. The response string starts with a starting time stamp and ends with an ending timestamp.	11

Direction Parameter

The Direction Parameter specifies the direction (Forward or Backward from the starting time) of data sampling from the archive. The default value is Forward.

Direction	Value
Forward	0
Backward	1

CalculationModeType Parameter

The CalculationModeType parameter is only applied if the Sampling Mode is set to Calculated. It represents the type of calculation to use on the archive data. The default Calculation Mode, if none is specified, is Average.

Calculation Mode Type	Description	Value
Undefined	Calculation mode is not defined.	0
Average	Retrieves the time-weighted average for each calculation interval.	1

Calculation Mode Type	Description	Value
StandardDeviation	Retrieves the time-weighted standard deviation for each calculation interval.	2
Total	Retrieves the time-weighted rate total for each calculation interval. Use rate totals when working with a continuous measurement. Time weighting takes into account that compressed data is not evenly spaced in time. A factor must be applied to the total value to convert into appropriate engineering units. As a rate total, the default is Units/Day. If the actual units of the continuous measurement are Units/Minute, you would multiply the results by 1440 (minutes per day) to convert the total into appropriate engineering units.	3
Minimum	Retrieves the minimum value for each calculation interval.	4
Maximum	Retrieves the maximum value for each calculation interval.	5
Count	Counts the number of raw samples found with good quality in the interval. Value is the count of raw samples with good quality in the interval. The values of each sample are ignored. The Count does not include any samples of bad quality, including the start and end of collection markers. For Quality, the percentage of good samples is always 100, even if the interval does not contain any raw samples, or contains only bad quality samples. Count is useful for analyzing the distribution of the raw data samples to determine the ef-	6

Calculation Mode Type	Description	Value
	fect of compression deadbands. It is also useful to determine which tags are consuming the most archive space.	
RawAverage	Retrieves the arithmetic average of all good quality raw samples for each calculation interval. Value is the sum of all good quality samples in the interval, divided by the number of good quality samples in the interval. All bad quality samples are ignored. That is RawAverage is equivalent to RawTotal divided by the Count. For Quality, if there are no raw samples in the interval or if they all are bad quality, then the percentage of good is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad quality samples. RawAverage is useful for calculating an accurate average when a sufficient number of raw samples are collected.	7
RawStandardDeviation	Retrieves the arithmetic standard deviation of raw values for each calculation interval. For Value, any raw point of bad data quality is ignored. For Quality, if there are no raw samples in the interval or they all have bad quality, then the percentage of good is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad quality samples. RawStandardDeviation is useful for calculating an accurate standard deviation when	8

Calculation Mode Type	Description	Value
	a sufficient number of raw samples are collected.	
RawTotal	Retrieves the arithmetic total (sum) of sampled values for each interval. Value is the sum of the good quality values of all raw samples in the interval. All bad quality samples are ignored. For Quality, the percentage of good samples is always 100, even if the interval does not contain any raw samples or it contains only bad quality samples. If the same start and end times are used, and the time span is treated as a single interval, then all values are added together. RawTotal is useful for calculating an accurate total when a sufficient number of raw samples are collected. Note that unlike ihTotal, this is a simple sum with no assumption that the values are rate values.	9
MinimumTime	Retrieves the timestamp of the minimum value found within each calculation interval. It can be a raw or an interpolated value. The minimum must be a good data quality sample.	10
MaximumTime	Retrieves the timestamp of the maximum value found within each calculation interval. It can be a raw or an interpolated value. The maximum must be a good data quality sample.	11
TimeGood	Retrieves the amount of time (milliseconds) during the interval when the data is of good quality and the filter condition is met.	12

Calculation Mode Type	Description	Value
StateCount	Retrieves the amount of time a tag uses to transition to another state from a previous state during a time interval.	13
StateTime	Retrieves the duration that a tag stayed in a given state within an interval.	14
OPCQAnd	Retrieves the OPCQAND, bit-wise AND operation of all the 16-bit OPC qualities of the raw samples stored in the specified interval. Note that OPC Quality is a subfield for Quality-Value-Timestamp (QVT), so when this calculation mode is used, OPC Quality is considered for calculation.	15
OPCQOr	Retrieves the OPCQOR, bit-wise OR operation of all the 16-bit OPC qualities of the raw samples stored in the specified interval. Note that OPC Quality is a subfield for Quality-Value-Timestamp (QVT), so when this calculation mode is used, OPC Quality is considered for calculation.	16
FirstRawValue	Retrieves the first good raw sample value for a given interval. Value is the value of the raw sample, or zero if there are no good raw samples in the interval. For Quality, if there are not good raw samples in the interval, then the percentage of good is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad quality samples. Note that Quality is the same for FirstRawValue and FirstRawTime.	17

Calculation Mode Type	Description	Value
	The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent.	
FirstRawTime	Retrieves the first good raw timestamp for a given interval. Value is the timestamp of the sample or the year 1969 if there are no good raw samples in the interval. For Quality, if there are not good raw samples in the interval, then the percentage of good is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad quality samples. Note that Quality is the same for FirstRawValue and FirstRawTime. The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent.	18
LastRawValue	Retrieves the last good raw sample value for a given time interval. Value is the value of the raw sample or zero if there are no good raw samples in the interval. For Quality, if there are no good raw samples in the interval, the percentage of good samples is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad samples. Note that Quality is the same for LastRawValue and LastRawTime. The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent.	19

Calculation Mode Type	Description	Value
LastRawTime	Retrieves the last good timestamp of the last value for a given time interval. Value is the timestamp of the sample or the year 1969 if there are no good raw samples in the interval. For Quality, if there are no good raw samples in the interval, the percentage of good samples is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad samples. Note that Quality is the same for LastRawValue and LastRawTime. The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent.	20
TagStats	Retrieves the statistics for a tag from the archive stored in the specified interval.	21

FilterModeType Parameter

The FilterModeType parameter defines how time periods before and after transitions in the filter condition should be handled.

When the FilterModeType parameter is applied, then the Start time and End time are specified as:

- ExactTime
- BeforeTime
- AfterTime
- BeforeAndAfterTime

For example, AfterTime indicates that the filter condition should be True starting at the timestamp of the archive value that triggered the True condition, and leading up to the timestamp of the archive value that triggered the False condition.

Properties	Description	Value
ExactTime	Retrieves data for the exact times that the filter condition is True.	1
BeforeTime	Retrieves data from the timestamp of the last False filter condition to the timestamp of the next True condition.	2
AfterTime	Retrieves data from the timestamp of the True filter condition to the time-stamp of the next False condition.	3
BeforeAndAfterTime	Retrieves data from the timestamp of the last False filter condition to the timestamp of the next False condition.	4

ReturnDataFields Parameter

The ReturnDataFields bitwise parameter specifies which data fields are returned in a query. Using it in a query returns data such as TimeStamp, and each field returns a Boolean value.

Each time-series data sample contains QVT (quality, value, and timestamp) values. If ReturnDataFields is not provided, then the default value of 0 is considered, and all QVT values are returned for each data sample. To return one of the data field properties, such as TimeStamp, use the TimeStamp option as shown in the table.

Properties	Description	Field value (Boolean)
All Fields	Specifies that all data fields are returned in the query.	0 (0000)
TimeStamp	The time stamp of the col-lected sample or an inter-val time stamp. When speci-fied in the query, returns the TimeStamp property.	1 (0001)
Value	The collected value or sam-pled value; the data type of the value will be the same	2 (0010)

Properties	Description	Field value (Boolean)
	data type as the tag's raw data.	
Quality	When specified in the query, returns the Quality property. Each sample in Current Value and Raw query retrieval has a quality of: • Good (3) • Not Available (2) • Uncertain (1) • Bad (0) Interpolated and Lab Retrieval express quality as "percent good".	4 (0100)

Payload Parameter

This parameter queries for the tag properties requested from the server.

Use the Payload parameter to query for all the tag properties to return from the server. In the Update Tag Configuration API, you must provide the actual tag property value. However, in the Get Tag Properties API, you must provide the property name and value of 1 (true), so the property can be read from the server and returned.

The properties listed in the following table are valid in APIs that use the Payload parameter, unless otherwise specified. For Property Names used in the Get Tag Properties API, the property name is always a Boolean (true/false) value, while it can be a string or integer for other APIs.

Property Name	Property Type	Description
AllFields	Boolean	Used for Get Tag Properties API.
Name	Boolean, String	Used for the Get Tag Properties API, Add Single Tag API, and Add Bulk Tags API.
Description	String	
EngineeringUnits	String	

<table>
<tr><th>Property Name</th><th>Property Type</th><th>Description</th></tr>
<tr><td>Comment</td><td>String</td><td></td></tr>
<tr><td>DataType : ihDataType</td><td>SignedInte-gral</td><td>Type definition is an enumerated type "ihData-Type".
<pre>{
ihDataTypeUndefined = 0,
ihScaled,
ihFloat,
ihDoubleFloat,
ihInteger,
ihDoubleInteger,
ihFixedString,
ihVariableString,
ihBlob,
ihTime,
ihInt64,
ihUInt64,
ihUInt32,
ihUInt16,
ihByte,
ihBool,
ihMultiField,
ihArray,
ihMaxDataType
} ihDataType;</pre></td></tr>
<tr><td>FixedStringLength</td><td>UnsignedChar</td><td></td></tr>
<tr><td>CollectorName</td><td>String</td><td></td></tr>
<tr><td>SourceAddress</td><td>String</td><td></td></tr>
<tr><td>CollectionType : ihCollectionType</td><td>SignedInte-gral</td><td>Type definition is an enumerated type "ihCollec-tionType".
<pre>{
ihUnsolicited = 1,
ihPolled
} ihCollectionType;</pre></td></tr>
</table>

Property Name	Property Type	Description
CollectionInterval	SignedIntegral	
CollectionOffset	UnsignedLong	
LoadBalancing	Boolean	

TimeStampType : ihTimeStampType	SignedIntegral	Type definition is an enumerated type "ihTimeStampType".
		`{`
		`ihSource = 1,`
		`ihInterface,`
		`} ihTimeStampType;`

HiEngineeringUnits	Double	
LoEngineeringUnits	Double	
InputScaling	Boolean	
HiScale	Double	
LoScale	Double	
CollectorCompression	Boolean	
CollectorDeadbandPercentRange	Float	
ArchiveCompression	Boolean	
ArchiveDeadbandPercentRange	Float	
General1	String	
General2	String	
General3	String	
General4	String	
General5	String	
ReadSecurityGroup	String	
WriteSecurityGroup	String	

<table>
<tr><th>Property Name</th><th>Property Type</th><th>Description</th></tr>
<tr><td>AdministratorSecurityGroup</td><td>String</td><td></td></tr>
<tr><td>LastModified</td><td>Boolean</td><td>Used for Get Tag Properties API.</td></tr>
<tr><td>LastModifiedUser</td><td>Boolean</td><td>Used for Get Tag Properties API.</td></tr>
<tr><td>InterfaceType</td><td>Boolean</td><td>Used for Get Tag Properties API.</td></tr>
<tr><td>CollectorType : ihInterfaceType</td><td>SignedIntegral</td><td>Type definition is an enumerated type "ihInterfaceType".
{
ihInterfaceUndefined = 0,
ihIFix,
ihRandom,
ihOPC,
ihFile,
ihIFixLabData,
ihManualEntry,
ihOther,
ihCalcEngine,
ihServerToServer,
ihPI,
ihOPCAE,
ihCIMPE,
ihPIDistributor,
ihCIMME,
ihPerfTag,
ihCustom,
ihServerToServerDistributor,
ihWindowsPerfMon,
} ihInterfaceType;</td></tr>
<tr><td>UTCBias</td><td>SignedIntegral</td><td></td></tr>
<tr><td>AverageCollectionTime</td><td>Boolean</td><td>Used for Get Tag Properties API.</td></tr>
<tr><td>CalculationDependencies</td><td>StringArray</td><td></td></tr>
<tr><td>CollectionDisabled</td><td>Boolean</td><td></td></tr>
</table>

Property Name	Property Type	Description
ArchiveCompressionTimeout	Unsigned-Long	
CollectorCompressionTimeout	Unsigned-Long	
SpikeLogic	Boolean	
SpikeLogicOverride	Boolean	
CollectorAbsoluteDeadbanding	Boolean	
CollectorAbsoluteDeadband	Double	
ArchiveAbsoluteDeadbanding	Boolean	
ArchiveAbsoluteDeadband	Double	
StepValue	Boolean	

TimeResolution : ihTimeResolution	SignedIntegral	Type definition is an enumerated type "ihTimeResolution".
		`{`
		`ihSeconds = 0,`
		`ihMilliseconds,`
		`ihMicroseconds,`
		`ihNanoseconds`
		`} ihTimeResolution;`

ConditionCollectionEnabled	Boolean	
ConditionCollectionTriggerTag	String	

ConditionCollectionComparison : ihConditionCollectionComparison	SignedIntegral	Type definition is an enumerated type "ihConditionCollectionComparison".
		`{`
		`ihConditionComparisonUndefined = 0,`
		`ihConditionComparisonEqual,`
		`ihConditionComparisonLessThan,`
		`ihConditionComparisonLessThanEqual,`
		`ihConditionComparisonGreaterThan,`
		`ihConditionComparisonGreaterThanEqual,`

Property Name	Property Type	Description
		`ihConditionComparisonNotEqual` `} ihConditionCollectionComparison;`
ConditionCollectionCompareValue	String	
ConditionCollectionMarkers	Boolean	
Calculation	String	When the Calculation field is used, then two more conditions are required. Calculation is not a specific field for a tag property. If the tag's collector or interface type is Server-to-server and the Calculation field is set (not Null), then the field value is set to the source address.
TagId	Boolean	Used for Get Tag Properties API.
EnumeratedSetName	String	
DataStoreName	String	
DefaultQueryModifiers	Long Long	
UserDefinedTypeName	String	
NumberOfElements	SignedIntegral	

DataDensity : ihTagDataDensity	SignedIntegral	Type definition is an enumerated type "ihTagDataDensity".
		`{`
		`ihDataDensityUndefined = 0,`
		`ihDataDensityMinimum = 1,`
		`ihDataDensityMedium = 4,`
		`ihDataDensityMaximum = 7`
		`} ihTagDataDensity;`

CalcType : ihTagCalcType	SignedIntegral	Type definition is an enumerated type "ihCalcType".
		`{`
		`ihRawTag = 0,`
		`ihAnalyticTag = 1,`

Property Name	Property Type	Description
		`ihPythonExprTag = 2` `} ihTagCalcType;`
HasAlias	Boolean	Used for Get Tag Properties API.
IsStale	Boolean	Used for Get Tag Properties API.

Error Code Definitions

The following table provides the values and definitions for the ErrorCode parameter.

Table 1. Error Code Definitions

Error Code Value:	Error Code Definition
Success = 0	Operation successful.
Failed = -1	Operation failed.
Timeout = -2	Operation failed due to timeout.
NotConnected = -3	Not connected to Historian server.
CollectorNotFound = -4	The given collector does not exist on the server.
NotSupported = -5	Operation not supported.
DuplicateData = -6	Attempt to overwrite an existing data sample.
InvalidUsername = -7	Bad user name or password.
AccessDenied = -8	Insufficient permissions for operation.
WriteInFuture = -9	Attempted data write too far in the future.
WriteArchiveOffline = -10	Attempted data write to an offline archive.
WriteArchiveReadonly = -11	Attempted data write to a read-only archive.
WriteOutsideActiveRange = -12	Attempted data write beyond the configured active range.
WriteNoArchiveAvailable = -13	Attempted data write with no available archives.
InvalidTagname = -14	The requested tag was not found.
LicensedTagCountExceeded = -15	Number of licensed tags exceeded.

Table 1. Error Code Definitions (continued)

Error Code Value:	Error Code Definition
LicensedConnectionCountExceeded = -16	Number of licensed server connections exceeded.
InternalLicenseError = -17	Internal license error.
NoValue = -18	No available tag data.
DuplicateCollector = -19	The given collector name already exists on the server.
NotLicensed = -20	Server or feature is not licensed.
CircularReference = -21	Circular reference detected in calculation.
BackupInsufficientSpace = -22	Insufficient disk space to perform backup.
InvalidServerVersion = -23	Operation unsupported due to server version.
QueryResultSizeExceeded = -24	Upper limit on query results exceeded.
DeleteOutsideActiveRange = -25	Attempted data delete outside allowed modification interval.
AlarmArchiverUnavailable = -26	Alarms and Events subsystem unreachable.
ArgumentException = -27	A supplied argument is invalid.
ArgumentNullException = -28	A supplied argument is NULL.
ArgumentOutOfRangeException = -29	A supplied argument is outside the valid range.
InvalidEnumeratedSet = -30	The requested Enumerated Set was not found.
InvalidDataStore = -31	The requested data store was not found.
NotPermitted = -32	Operation not permitted.
InvalidCustomDataType = -33	The Custom data type is not supported.
ihSTATUS_EXISTING_USERDEF_REFERENCES = -34	N/A
ihSTATUS_INVALID_TAGNAME_DELETEDTAG = -35	N/A
ihSTATUS_INVALID_DHS_NODENAME = -36	N/A
ihSTATUS_DHS_SERVICE_IN_USE = -37	N/A

Table 1. Error Code Definitions (continued)

Error Code Value:	Error Code Definition
ihSTATUS_DHS_STORAGE_IN_USE = -38	N/A
ihSTATUS_DHS_TOO_MANY_NODES_IN_MIRROR = -39	N/A
ihSTATUS_ARCHIVE_IN_SYNC = -40	N/A
InvalidArchiveName= -41	N/A
InvalidSession = 1	Session id is invalid.
SessionExpired = 2	Session has expired.
UnknownError = 3	Unknown error, please check server log.
NoValidClientBufferManager= 4	No valid client buffer manager.
NoValueInDataSet = 5	No value in returned data set.
TagNotExisting = 6	Tag doesn't exist.
ClientBufferManagerCommunicationError = 7	Service call to central buffer server fail.
TagTypeNotSupported=8	Tag type is not supported.
ValueTypeNotMatchTagDataType = 9	Value type doesn't match tag data type.
InvalidParameter=10	Invalid query parameter.
TagSearchResultIsHuge = 11	Tag Search Criteria result was more than 5000.
InvalidHistorianServer=12	No valid server or historian server name isn't in the server list.
ihSTATUS_INVALID_INTERFACETYPE = -49	The collector type is not valid.
ihSTATUS_INTERFACE_START_FAIL = -50	Starting the collector has failed.
ihSTATUS_INTERFACE_STOP_FAIL = -51	Stopping the collector has failed.

Managing Tags

The Tags API

The Tags API retrieves the qualified tag name list by a given `nameMask`.

> **Note:**
> URI format supports asterisks (*) and question marks (?).

METHOD	GET
URI	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/{nameMask}/{maxNumber}`
SAMPLE URI	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags?nameMask=*&maxNumber=100`
SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/tags?nameMask=*&maxNumber=<Number_Of_Tags>`

Table 2. Query Parameters

Parameter	Description	Required?	Values
`nameMask`	Tagmask that searches for all tags that match the mask and applies the remaining criteria to retrieve data. The mask can include wildcards, such as asterisks (*).	Optional	String
`maxNumber`	Maximum tag number provides the limit while returning the results (0 by default). This means that for a query, if using 0, all tags are returned. If a negative number is used, then 0 is used for the maxNumber. If a positive number is used, then that number of tags is returned.	Optional	Integer 0 by default

Table 2. Query Parameters (continued)

Parameter	Description	Required?	Values
	In addition, an error number of +14 notifies the user that there are more than the requested number of tags in the system.		

Table 3. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Number	Yes	For example, ErrorCode = 0, which means the operation was successful.
ErrorMessage	String	Yes	For example, NULL.
tags	String	Yes	Includes the following: • ALT_SENSOR • tagName1 • tagName2

The Raw Data API

The Raw Data API queries raw data, such as a number of samples or the time range for a list of tags. If the count is not zero, then the API service returns the number of raw samples taken beginning from the start time. If the count is zero, then the service returns the raw samples taken between the start time and the end time.

METHOD:	GET, POST

URI:	**GET**
	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/raw/{tagNames}/{start}/{end}/{direction}/{count}`
	POST
	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/raw/{start}/{end}/{direction}/{count}`

SAMPLE GET URI:	**Raw By Number**
	
	Count value is a non-zero positive number, and end time is greater than start time.
	
	`https://<Azure-Load-Balancer-IP-of-Proficy-Histori-`
	`an-for-Cloud>:9090/historian-rest-api/datapoints/raw/tag-`
	`Name1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/`
	`0/100https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-`
	`Cloud>:9090/historian-rest-api/datapoints/raw?tagNames=tag-`
	`Name1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-`
	`.111Z&count=100&direction=0`
	
	Raw By Time
	
	The count value equals 0.
	
	`https://<Azure-Load-Balancer-IP-of-Proficy-Histori-`
	`an-for-Cloud>:9090/historian-rest-api/datapoints/raw/tag-`
	`Name1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/`
	`0/0https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-`
	`Cloud>:9090/historian-rest-api/datapoints/raw?tagNames=tag-`
	`Name1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-`
	`.111Z&count=0&direction=0`

SAMPLE POST URI:	**Raw By Number**
	
	Count value is a non-zero positive number, and end time is greater than start time.
	
	`https://<Azure-Load-Balancer-IP-of-Proficy-His-`
	`torian-for-Cloud>:9090/historian-rest-api/data-`
	`points/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/`
	`0/100https://<Azure-Load-Balancer-IP-of-Proficy-Histo-`
	`rian-for-Cloud>:9090/historian-rest-api/datapoints/raw?`
	`start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-`
	`.111Z&count=100&direction=0`
	
	Raw By Time
	
	The count value equals 0.

	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/datapoints/raw?start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-.111Z&count=0&direction=0`
SAMPLE cURL COMMAND (GET): [Raw By Number]	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/ datapoints/raw/<tagName>/<start time>/<end time>/<direction>/<count>`
SAMPLE cURL COMMAND (GET): [Raw By Time]	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/ datapoints/raw/<tagName>/<start time>/<end time>/<direction>/0`
SAMPLE cURL COMMAND (POST): [Raw By Number]	`curl –X POST -i -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” -d “{\”tagNames\”:\”<tagName>;<tagName>\”}” http:// <Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>/ historian-rest-api/v1/ datapoints/raw/ <start time>/<end time>/<direction>/<count>`
SAMPLE cURL COMMAND (POST): [Raw By Time]	`curl –X POST -i -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” -d “{\”tagNames\”:\”<tagName>;<tagName>\”}” http:// <Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>/ historian-rest-api/v1/ datapoints/raw? start=<start time>&end=<end time>&direction=<direction>&count=<count>`

Table 4. Query Parameters

Parameter	Description	Required?	Values
TagNames	Queries the specified tag names.	Yes	String
Start	Start time of the query, in ISO data format (such	Yes	DateTime

Table 4. Query Parameters (continued)

Parameter	Description	Re-quired?	Values
	as YYYY-MM-DDTHH:mm:ss.SSSZ).		
End	End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
Direction	Specifies the direction (Forward or Backward from the starting time) of data sampling from the archive. The default value is Forward (0).	Yes	Integer, with a value such as 0.
Count	Count of archive values within each calculation interval.	Yes	Integer, with a value such as 100.

Table 5. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorMessage	String	Yes	For example, NULL.
Data	String	Yes	The object container for the following parameters: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful.

Table 5. Response Parameters (continued)

Parameter	Data Type	Required?	Description
			TagName Example: TagName1. **Samples** Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2013-10-02T11:30:00.111Z, Value = 34.26155, and Quality = 3.

The Interpolated Data API

The Interpolated Data API queries interpolated values for a list of tags. If the start time equals the end time, the request returns one sample.

METHOD:	GET, POST

URI:	**GET**
	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/{tagNames}/{start}/{end}/{count}/{intervalMs}`
	POST
	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/{start}/{end}/{count}/{intervalMs}`

SAMPLE GET URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/tagName1/2013-10-02T11:30:00.111111Z/2013-10-02T11:31:11.111Z/100/10000`

SAMPLE POST URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/2013-10-02T11:30:00.111111Z/2013-10-02T11:31:11.111Z/100/10000`
SAMPLE cURL COMMAND (GET):	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/ datapoints/interpolated/<tagName>/<start time>/<end time>/<count>/<intervalMS>`
SAMPLE cURL COMMAND (POST):	`curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" -d "{\"tagNames\":\"<tagName>\"}" http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/ datapoints/interpolated/<start time>/<end time>/<count>/<intervalMS>`

Table 6. Query Parameters

Parameter	Description	Required?	Values
TagName	Queries the tag names specified.	Yes	String
Start	Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
End	End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
Count	Count of archive values within each calculation interval.	Yes	Integer, with a value such as 100.
intervalMS	Interval in milliseconds.	Yes	64-bit signed integer, with a value such as 10000.

Table 7. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorMessage	String	Yes	For example, NULL.
Data	String	Yes	The object container for the following parameters: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful. **TagName** Example is TagName1. **Samples** Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2013-10-02T11:30:00.111Z, Value = 34.26155, and Quality = 3.

The Current Value API

The Current Value API queries the current value data and reads the current values for a list of tags. If the start time is equal to end time, the request returns one sample.

METHOD:	GET, POST
URI:	**GET**

	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/raw/{tagNames}/{start}/{end}/{direction}/{count}`
	POST
	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`

SAMPLE GET URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue?tagNames=tagName1`
SAMPLE POST URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
SAMPLE cURL COMMAND (GET):	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/ datapoints/currentvalue/<tagName>`
SAMPLE cURL COMMAND (POST):	`curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” -d “{\”tagNames\”:\”<tagName>\”}” http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/ datapoints/currentvalue`

Table 8. Query Parameters

Parameter	Description	Required?	Values
TagNames	Queries the specified tag names.	Yes	String

Table 9. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorMessage	String	Yes	For example, NULL.

Table 9. Response Parameters (continued)

Parameter	Data Type	Required?	Description
Data	String	Yes	The object container for the following parameters: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful. **TagName** Example is TagName1. **Samples** Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2014-01-01T12:00:00Z, Value = 34.26155, and Quality = 3.

The Calculated Data API

The Calculated Data API queries the calculated data for a list of tags. Data can be requested using a number of samples or a time range for a list of tags. If the count is not zero, the service returns the number of raw samples beginning from the start time. If the count is zero, the services uses the interval, start time, and end time to calculate the required sample number.

METHOD:	GET, POST
URI:	**GET**

	https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/calculated/{tagNames}/{start}/{end}/{calculationMode}/{count}/{intervalMs} **POST** https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/calculated/{start}/{end}/{calculationMode}/{count}/{intervalMs}
SAMPLE GET URI:	**Number of Samples** https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/calculated/tagName1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/1/100/1000 **Time Range for List of Tags** https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/calculated?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&count=100&calculationMode=1&intervalMs=1000
SAMPLE POST URI:	**Number of Samples** https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/calculated/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/1/100/1000 **Time Range for List of Tags** https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/calculated?start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&count=100&calculationMode=1&intervalMs=1000

SAMPLE cURL COMMAND (GET):	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:8843/ historian-rest-api/v1/ datapoints/calculated/<tag-Name>/<start time>/<end time>/<count>/<calculation mode>/<intervalMS>`
SAMPLE cURL COMMAND (POST):	`curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" -d "{\"tagNames\":\"<tagName>\"}" http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:8843/ historian-rest-api/v1/ datapoints/calculated/<start time>/<end time>/<count>/<calculation-mode>/<intervalMS>`

Table 10. Query Parameters

Parameter	Description	Required?	Values
TagNames	GE identifier for a location.	Yes	1000000106
Start	Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
End	End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
Count	Count of archive values within each calculation interval.	Yes	Integer, with a value such as 100.
Calculation Mode	End time in milliseconds.	Yes	Integer, with a value such as 1.
IntervalMS	Interval in milliseconds.		64-bit signed integer, with a value such as 1000.

Table 11. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorCode	String	Yes	For example, NULL.
Data	String	Yes	The object container for the following parameters: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful. **TagName** Example is TagName1. **Samples** Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2013-10-02T11:30:00.111Z, Value = 34.26155, and Quality = 3.

The Sampled Data API

The Sampled Data API queries the sampled data for a list of tags. Data can be requested using a number of samples or a time range for a list of tags. If the count is not zero, the service returns the number of raw samples beginning from the start time. If the count is zero, the services uses the interval, start time, and end time to calculate the required sample number.

Note:
For the query, you can also use optional parameters such as FilterMode and ReturnDataFields. Unused parameters can be omitted.

METHOD:	GET, POST
URI:	**GET** `https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/sampled` **POST** `https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/sampled`
SAMPLE GET URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/sampled?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&samplingMode=1&calculationMode=1&direction=0&count=0&intervalMs=1000`
SAMPLE POST URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/sampled`
SAMPLE cURL COMMAND (GET):	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/ datapoints/sampled/<tagName>/<start time>/<end time>/<direction>/<count>/<intervalMS>`
SAMPLE cURL COMMAND (POST):	`curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”tagNames\”:\”<tagName>\”, \”start\”: \”<start>\”, \”end\”: \”<end>\”, \”samplingMode\”: <samplingMode>, \”calculationMode\”: <calculationMode>, \”direction\”: <direction>, \”count\”: <count>, \”returnDataFields\”: <returnDataFields>, \”intervalMs\”: <intervalMs>, \”queryModifier\”: <queryModifier>, \”filterMode\”: <filterMode>, \”filterExpression\”:`

	`\”<filterExpression>\”}” http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/data-points/sampled`

Table 12. Query Parameters

Parameter	Description	Required?	Values
TagNames	Queries the tag names specified.	Yes	String
Start	Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
End	End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
SamplingMode	Also known as SamplingModeType.	Optional	Integer, with a value such as 1.
CalculationMode	Also known as CalculationModeType.	Optional	Integer, with a value such as 1.
Direction	Specifies the direction (Forward or Backward from the starting time) of data sampling from the archive. The default value is Forward (0).	Optional	Integer, with a value such as 0.
Count	The count of archive values within each calculation interval.	Optional	Integer, with a value such as 0.
IntervalMS	Interval in milliseconds.	Optional	64-bit signed integer, with a value such as 1000.

Table 13. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorCode	String	Yes	For example, NULL.
Data	String	Yes	The object container for the following parameters: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful. **TagName** Example is TagName1. **Samples** Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2013-10-02T11:30:00.111Z, Value = 34.26155, and Quality = 3.

The Trend Data API

The Trend Data API queries the trend data for a list of tags.

> **Note:**
> For the query, you can also use optional parameters such as FilterMode and StatisticsItemFilter. Unused parameters can be omitted.

METHOD:	GET, POST
URI:	**GET** https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/trend **POST** https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/trend
SAMPLE GET URI:	https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/trend?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&samplingMode=1&calculationMode=1&direction=0&count=0&intervalMs=1000
SAMPLE POST URI:	https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/trend
SAMPLE cURL COMMAND (GET):	curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/ datapoints/trend/<tagName>/<start time>/<end time>/<samplingMode>/<calculationMode>/<direction>/<count>/<intervalMS>
SAMPLE cURL COMMAND (POST):	curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”tagNames\”:\”<tagName>\”, \”start\”: \”<start>\”, \”end\”: \”<end>\”, \”samplingMode\”: <samplingMode>, \”calculationMode\”: <calculationMode>, \”direction\”: <direction>, \”count\”: <count>, \”returnDataFields\”: <returnDataFields>, \”intervalMs\”: <intervalMs>, \”queryModifier\”: <queryModifier>, \”filterMode\”: <filterMode>, \”filterExpression\”: \”<filterExpression>\”}” http://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/trend

Table 14. Query Parameters

Parameter	Description	Required?	Values
TagNames	Queries the tag names specified.	Yes	String
Start	Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
End	End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ).	Yes	DateTime
SamplingMode	Also known as SamplingModeType.	Optional	Integer, with a value such as 1.
CalculationMode	Also known as CalculationModeType.	Optional	Integer, with a value such as 1.
Direction	Specifies the direction (Forward or Backward from the starting time) of data sampling from the archive. The default value is Forward (0).	Optional	Integer, with a value such as 0.
Count	The count of archive values within each calculation interval.	Optional	Integer, with a value such as 0.
IntervalMS	Interval in milliseconds.	Optional	64-bit signed integer, with a value such as 1000.

Table 15. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.

Table 15. Response Parameters (continued)

Parameter	Data Type	Required?	Description
ErrorMessage	String	Yes	For example, NULL.
Data	String	Yes	The object container for the following parameters: **TagName** Name of the tag, such as ahistfile.Simulation00001. **TagSource** Location where tags are being searched for. **DataType** Float, which stores decimal values up to 6 places. **Trend** Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2016-03-15T04:53:17.000Z, Value = 170903.6563, and Quality = True.

The Add Single Tag API

For the Add Single Tag API, you can add a new tag to Historian, and the tag name and data type must be provided in the payload (parameter) of the method. All other tags are optional. If a property is provided, the respective validation is performed at the server end. If the tag exists, then any new properties provided in the payload are applied to the existing tag.

METHOD:	POST
URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/addtag`

SAMPLE DELETE URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/addtag`
	
	`Payload:`
	
	`{`
	`"Name" : "SampleTag",`
	`"DataType" : 3`
	`}`

SAMPLE cURL COMMAND:	`curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>" -d "{ \"Name\": \"Sampletag\",\"DataType\":3}" -X POST https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/addtag`

Table 16. Query Parameters

Parameter	Description	Required?	Values
Payload	JSON array of PropertyName and PropertyValue.	Yes. "Name" and "DataType" properties are required. All other properties are optional.	Multidata types. See Payload Parameter *(on page 124)* for a list of tag properties used to update a tag configuration.

Sample Response

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorMessage	String	Yes	For example, NULL.

The Add Bulk Tags API

For the Add Bulk Tags API, you can add new tags to Historian using an array, and the tag names and data types must be provided in the payload (parameter) of the method. All other tags are optional. If a property is provided, the respective validation is performed at the server end. If the tags exist, then any new properties provided in the payload are applied to the existing tags. The payload is be an array of tags defined.

METHOD:	POST
URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/addtags`

SAMPLE DELETE URI:	https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/addtags
	
	Payload:
	
	[
	{
	"Name" : "SampleTag1",
	"DataType" : 3
	},
	{
	"Name" : "SampleTag2",
	"DataType" : 3
	}
	]

SAMPLE cURL COMMAND:	`curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>" -d "[{ \"Name\": \"Sampletag1\"}, { \"Name\":\"Sampletag2\"}]" -X POST https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/addtags`

Table 17. Query Parameters

Parameter	Description	Required?	Values
Payload	JSON array tags with individual tags of PropertyName and PropertyValue.	Yes. "Name" and "DataType" properties are required. All other properties are optional.	Multidata types. See Payload Parameter *(on page 124)* for a list of tag properties used to update a tag configuration.

Table 18. Response Parameters

Parameter	Data Type	Exists?	Description
TagName	String	Yes	Tag name.
ErrorCode	Integer	Yes	For example, 0.
ErrorMessage	String	Yes	For example, NULL.

The Update Tag Configuration API

The Update Tag Configuration API allows you to set or modify any tag property values. You cannot, however, rename a tag using this API.

METHOD:	PUT
URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName`

SAMPLE DELETE URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName`
	
	`Payload:`
	`{`
	`    "PropertyName" : "PropertyValue"`
	`}`

SAMPLE cURL COMMAND:	`curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”Description\”:\”SampleDesc\”}” -X PUT https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName`

Table 19. Query Parameters

Parameter	Description	Required?	Values
tagName	Tag name for which properties need to be set or modified.	Yes	String

Table 19. Query Parameters (continued)

Parameter	Description	Required?	Values
Payload	JSON array of PropertyName and PropertyValue.	At least one property must be provided.	Multidata types. See Payload Parameter *(on page 124)* for a list of tag properties used to update a tag configuration.

Table 20. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorMessage	String	Yes	For example, NULL.

The Get Tag Properties API

You can use this API to specify which properties are required for retrieval. If no property names are provided, then all properties are retrieved. When using the Get Tag Properties method, requesting a non-existent tag name returns an error.

METHOD:	GET / POST
URI: (GET)	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName` This URI returns all tag properties.

URI: (POST)	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName`
	`Payload`
	`{`
	`"PropertyName1" : 1,`
	`"PropertyName2" : 1`
	`}`

SAMPLE GET URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName`

SAMPLE POST URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName`
	
	`Payload:`
	`{`
	`    "Description" : 1`
	`}`

SAMPLE cURL GET COMMAND:	`curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -X GET https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName`
SAMPLE cURL POST COMMAND:	`curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”Description\”: 1}” -X POST https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/properties/tagName`

Table 21. Query Parameters

Parameter	Description	Required?	Values
tagName	Tag name for which properties need to be retrieved.	Yes	String
Payload	JSON array of PropertyName and boolean (true/false).	At least one property must be provided.	Multi data types. See Payload Parameter *(on page 124)* for a list of tag properties used to update a tag configuration.

Note:
The query payload contains all the tag properties you want returned from the server. In the Update Tag Config method, you need to provide the actual tag property value. However, in the Get Tag Properties method, you need to provide the property and a value of 1 (true), to allow it to be read from the server and returned.

Table 22. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorMessage	String	Yes	For example, NULL.
Name	String	Optional	If no error, then the tag name of query is returned and all requested parameters.

The Delete Tag API

The Delete Tag API provides the ability to delete an existing tag from the Historian server.

Its URI format supports question marks (?).

METHOD:	DELETE
URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/tagName?{permanentDelete}`
SAMPLE DELETE URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/tagName?permanent-Delete=true`
SAMPLE CURL COMMAND:	`curl -i -H "Authorization: Bearer <TOKEN>” -X DELETE https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/tagName?permanentDelete=<true|false>`

Table 23. Query Parameters

Parameter	Description	Required?	Values
tagName	Name of the tag to be deleted.	Yes	String
permanent-Delete	Deletes the tag permanently from the Historian server if the value passed in is true. If the parameter is not provided, then permanent-	Optional (false is default)	Boolean, true or false

Table 23. Query Parameters (continued)

Parameter	Description	Re-quired?	Values
	Delete is assumed to be false.		

Table 24. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Number	Yes	For example, ErrorCode=0, which means the operation was successful.
ErrorMessage	String	Yes	For example, NULL.

The Query Results API

The Query Results API enables you to include the number of samples required, by providing an end point to configure query results.

The minimum number of samples should be 1000.

METHOD:	PUT
URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/ configuration/{max-DataQueryResultSize}`
SAMPLE URI:	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/ configuration?max-DataQueryResultSize=6000`
SAMPLE CURL COMMAND:	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/ historian-rest-api/v1/datapoints/configuration? max-DataQueryResultSize=<Number_Of_Query_Results>`

Table 25. Query Parameters

Parameter	Description	Required?	Values
maxDataQueryResultSize	Maximum samples that should be configured as part of Query Results	Yes	Integer

Maximum DataQueryResultSize set to 6000

The Tag Rename API

This API allows the administrator to rename tags.

<table>
<tr><th>METHOD:</th><th>PUT</th></tr>
<tr><td>URI</td><td><code>https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/tagrename/oldtagname/newtagname?{truerename}</code></td></tr>
<tr><td>SAMPLE URI</td><td><code>https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/tagrename/GDW14NV2E.Simulation0000101/GDW14NV2E.Simulation0000101newname?truerename= <true | false></code></td></tr>
<tr><td>SAMPLE CURL COMMAND</td><td><code>curl -i -H "Accept: application/json" -i -H "Content-Type: application/json"-H "Authorization: Bearer <TOKEN> -X PUT https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/tags/tagrename/<oldtagname>/<newtagname>?truerename=<true | false></code></td></tr>
</table>

Table 26. Query Parameters

Parameter	Description	Required?	Values
oldtagname	Tag which is to be renamed.	Yes	String
newtagname	New name for the selected tag.	Yes	String
truerename	Renames the tag permanently if the value entered is true.	Optional (false is default)	Boolean (true or false)

Table 26. Query Parameters (continued)

Parameter	Description	Required?	Values
	Creates an alias if the value entered is false.		

Table 27. Response Parameters

Parameter	Data Type	Required?	Description
Error Code	Integer	Yes	For example, 0.
Error Message	String	Yes	For example, NULL.
Data	List	Yes	Returns all the properties of the tag.

The Write Tag API

Write Tag Data API enables you to create data for tags. You can write data to a tag for different data types such as integer, float, array, multifield and so on. Once created, you can view the data using other end points. Only REST API Administrator and users with write permission can perform this operation.

Method	POST
URI	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api /v1/datapoints/create`

SAMPLE URI	`https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api /v1/datapoints/create`
	
	`Payload`
	
	`{`
	`"TagName": "GDW14NV2E.Simulation00015",`
	`"samples": [`
	`{`

Method	POST
	`"TimeStamp": "2019-09-17T15:58:00.000Z", "Value": "1", "Quality": 3 } ] }`
SAMPLE RESPONSE	`{ "ErrorCode": 0, "ErrorMessage": "" }`
SAMPLE CURL COMMAND	`curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>" -d "{ \"TagName\":\"GDW14NV2E.Simulation00015\",\"samples\":[{\"TimeStamp\": \ "2019-09-17T15:58:00.000Z\",\"Value\": \"1\",\"Quality\": 3}]}" -X POST https://<Azure-Load-Balancer-IP-of-Proficy-Historian-for-Cloud>:9090/historian-rest-api/v1/datapoints/create`

Table 28. Query Parameters

Parameter	Description	Required?	Values
Payload	JSON format of Property Name and Property Value.	Yes	Multi-data types. It can have integer, float, array, multifield data types.

Table 29. Response Parameters

Parameter	Data Type	Required?	Description
Error Code	Integer	Yes	For example, ErrorCode = 0, which means the operation was successful.
Error Message	String	Yes	For example, NULL.

REST APIs for Managing Tag Data

This topic provides REST APIs that you can use to manage tags. You can add, access, modify, rename, and delete tags.

Before You Begin: Get an authorization token *(on page 108)*.

Table 30. Access Raw Data Using the GET Method

Parameter	Value
Method	GET
URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/raw/<tag names>/<start>/<end>/<direction>/<count>` **Tip:** To find the Azure Load Balancer IP: 1. Go to the Azure portal. 2. Go to the **Resource Group** that was specified during deployment. 3. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. 4. Select or copy the IP Address.
Authorization	Bearer *<token>*
Content type	application/json

Table 30. Access Raw Data Using the GET Method (continued)

Parameter	Value

Sample URI	• **Raw By Number:** Retrieves data samples up to a specified number.
	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/datapoints/raw/tagName1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/100`
	• **Raw By Time:** Retrieves data samples up to a specified time.
	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/datapoints/raw/tagName1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0`

Sample cURL commands	• **Raw By Number:**
	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/raw/<tag name>/<start time>/<end time>/<direction>/<count>`
	• **Raw By Time:**
	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/raw/<tag name>/<start time>/<end time>/<direction>/0`

Query parameters	• **Tag names:** The names of tags whose data you want to retrieve. • **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ).

Table 30. Access Raw Data Using the GET Method (continued)

Parameter	Value
	• **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. • **Count:** The count of the data samples within each calculation interval.

Table 31. Access Raw Data Using the POST Method

Parameter	Value
Method	POST
URI	https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/raw/<*start*>/<*end*>/<*direction*>/<*count*>
Authorization	Bearer <*token*>
Content type	application/json
Sample URI	• **Raw By Number:** Retrieves data samples up to a specified number. https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/100 • **Raw By Time:** Retrieves data samples up to a specified time. https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0
Sample cURL commands	• **Raw By Number:** curl X POST -i -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <*token*>-d {\tagNames\:\<*tag name*>;<*tag name*>\}http://<*Azure Load Balancer IP of Proficy Historian for Cloud*>/histori-

Table 31. Access Raw Data Using the POST Method (continued)

<table>
<tr><th>Parameter</th><th>Value</th></tr>
<tr><td></td><td><code>an-rest-api/v1/datapoints/raw/<tag name><start time>/<end time>/<direction>/<count></code>
• Raw By Time:
<code>curl X POST -i -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <token>-d {\tagNames\:\<tag name>;<tag name>\}http://<Azure Load Balancer IP of Proficy Historian for Cloud>/historian-rest-api/v1/ datapoints/raw?start=<start time>&end=<end time>&direction=<direction>&count=<count></code></td></tr>
<tr><td>Query parameters</td><td>• Tag names: The names of tags whose data you want to retrieve.
• Start time: The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ).
• End time: The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ).
• Direction: The direction (forward or backward) from the start time for which you want to retrieve data.
• Count: The count of the data samples within each calculation interval.</td></tr>
<tr><td>Body</td><td><pre>{
    "ErrorCode" : 0,
    "ErrorMessage": null,
    "Data":[{
     "DataType":"<value>",
     "ErrorCode":0,
     "TagName":"<value>",
     "Samples":
       [
         { "TimeStamp":"<value>","Value":"<value>","Quality":<value> },
         { "TimeStamp":"<value>","Value":"<value>","Quality":<value> },
         { "TimeStamp":"<value>","Value":"<value>","Quality":<value> }</pre></td></tr>
</table>

Table 31. Access Raw Data Using the POST Method (continued)

Parameter	Value

	`]`
	`}]`
	`}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }`
	`]`
	`}]`
	`}`

Table 32. Access Interpolated Data

Parameter	Value
Method	GET
URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/<start>/<end>/<count>/<interval in milliseonds>`
Authorization	Bearer <*token*>
Content type	application/json
Sample URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/tag-`

Table 32. Access Interpolated Data (continued)

Parameter	Value
	`Name1/2013-10-02T11:30:00.111111Z/2013-10-02T11:31:11.111Z/100/10000`
Sample cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/ historian-rest-api/v1/datapoints/interpolated/<tag name>/<start time>/<end time>/<count>/<interval in milliseconds>`
Query parameters	• **Tag names:** The names of tags whose data you want to retrieve. • **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **Interval:** The interval in milliseconds. • **Count:** The count of the data samples within each calculation interval.

Body	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"<value>",`
	`"ErrorCode":0,`
	`"TagName":"<value>",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }`
	`]`
	`}]`
	`}`

Table 32. Access Interpolated Data (continued)

Parameter	Value

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }`
	`]`
	`}]`
	`}`

Table 33. Update Interpolated Data

Parameter	Value
Method	POST
URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/<start>/<end>/<count>/<interval in milliseonds>`
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/2013-10-02T11:30:00.111111Z/2013-10-02T11:31:11.111Z/100/10000`

Table 33. Update Interpolated Data (continued)

Parameter	Value
Sample cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/ historian-rest-api/v1/datapoints/interpolated/<start time>/<end time>/<count>/<interval in milliseconds>`
Query parameters	• **Tag names:** The names of tags whose data you want to retrieve. • **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **Interval:** The interval in milliseconds. • **Count:** The count of the data samples within each calculation interval.

Body	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"<value>",`
	`"ErrorCode":0,`
	`"TagName":"<value>",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }`
	`]`
	`}]`
	`}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`

Table 33. Update Interpolated Data (continued)

Parameter	Value
	"Data":[{ "DataType":"DoubleFloat", "ErrorCode":0, "TagName":"TagName1", "Samples": [{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 }, { "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 }, { "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }] }] }

Table 34. Access Current Data

Parameter	Value
Method	GET
URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
Authorization	Bearer <*token*>
Content type	application/json
Sample URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue?tag-Names=tagName1`
Sample cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/ historian-rest-api/v1/datapoints/currentvalue/<tag name>`
Query parameters	**Tag names:** The names of tags whose data you want to retrieve.

Table 34. Access Current Data (continued)

Parameter	Value
Body	`{ "ErrorCode" : 0, "ErrorMessage": null, "Data":[{ "DataType":"DoubleFloat", "ErrorCode":0, "TagName":"TagName1", "Samples":[ { "TimeStamp":"<value>","Value":"<value>","Quality":<value> } ] }] }`
Example	`{ "ErrorCode" : 0, "ErrorMessage": null, "Data":[{ "DataType":"DoubleFloat", "ErrorCode":0, "TagName":"TagName1", "Samples":[ { "TimeStamp":"2014-01-01T12:00:00Z","Value":"34.26155","Quality":3 } ] }] }`

Table 35. Update Current Data

Parameter	Value
Method	POST
URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
Authorization	Bearer *<token>*
Content type	application/json

Table 35. Update Current Data (continued)

Parameter	Value
Sample URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
Sample cURL command	`curl -i X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <token>-d {\tagNames\:\<tag name>\}http://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
Query parameters	**Tag names:** The names of tags whose data you want to retrieve.

Body	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"<value>",`
	`"ErrorCode":0,`
	`"TagName":"<value>",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }`
	`]`
	`}]`
	`}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	`"Samples":`
	`[`

Table 35. Update Current Data (continued)

Parameter	Value
	{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 }, { "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 }, { "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }] }] }

Table 36. Access Sampled Data

Parameter	Value
Method	GET
URI	https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/sampled/<*tag names*>/<*start time*>/<*end time*>/<*direction*>/<*count*>/<interval>
Authorization	Bearer <*token*>
Content type	application/json
Sample URI	https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/sampled?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&samplingMode=1&calculationMode=1&direction=0&count=0&intervalMs=1000
Sample cURL commands	curl -i -H "Accept: application/json" -H "Authorization: Bearer <*token*>http://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:8843/historian-rest-api/v1/datapoints/sampled/<*tag name*>/<*start time*>/<*end time*>/<*direction*>/<*count*>/<interval>

Table 36. Access Sampled Data (continued)

<table>
<tr><th>Parameter</th><th>Value</th></tr>
<tr><td>Query parameters</td><td>Tag names: The names of tags whose data you want to retrieve.
Start time: The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ).
End time: The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ).
Interval: The interval in milliseconds.
Direction: The direction (forward or backward) from the start time for which you want to retrieve data.
Count: The count of the data samples within each calculation interval.</td></tr>
<tr><td>Body</td><td><pre>{
     "ErrorCode" : 0,
     "ErrorMessage": null,
     "Data":[{
      "DataType":"DoubleFloat",
      "ErrorCode":0,
      "TagName":"TagName1",

 "Samples":[{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }]
      }]
}</pre></td></tr>
<tr><td>Example</td><td><pre>{
     "ErrorCode" : 0,
     "ErrorMessage": null,
     "Data":[{
      "DataType":"DoubleFloat",
      "ErrorCode":0,
      "TagName":"TagName1",</pre></td></tr>
</table>

Table 36. Access Sampled Data (continued)

Parameter	Value
	"Samples":[{ "TimeStamp":"2014-01-01T12:00:00Z","Value":"34.26155","Quality":3 }] }] }

Table 37. Update Sampled Data

Parameter	Value
Method	POST
URI	https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/sampled/<*tag names*>/<*start time*>/<*end time*>/<*direction*>/<*count*>/<interval>
Authorization	Bearer <*token*>
Content type	application/json
Sample URI	https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/sampled?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&samplingMode=1&calculationMode=1&direction=0&count=0&intervalMs=1000
Sample cURL command	curl -i X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <*token*>-d { \tagNames\:\<*value*>\, \start\: \<*value*>\, \end\: \<*value*>\, \samplingMode\: <*value*>, \calculationMode\: <*value*>, \direction\: <*value*>, \count\: <*value*>, \returnDataFields\: <*value*>, \intervalMs\: <*value*>, \queryModifier\: <*value*>, \filterMode\: <*value*>, \filterExpression\: \<*value*>\}http://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/sampled

Table 37. Update Sampled Data (continued)

<table>
<tr><th>Parameter</th><th>Value</th></tr>
<tr><td>Query parameters</td><td>

- **Tag names:** The names of tags whose data you want to retrieve.
- **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ).
- **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ).
- **Interval:** The interval in milliseconds.
- **Sampling mode:** The sampling mode using which you want to retrieve data.
- **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data.
- **Count:** The count of the data samples within each calculation interval.

</td></tr>
<tr><td>Body</td><td><pre>{
     "ErrorCode" : 0,
     "ErrorMessage": null,
     "Data":[{
      "DataType":"<value>",
      "ErrorCode":0,
      "TagName":"<value>",
      "Samples":
        [
          { "TimeStamp":"<value>","Value":"<value>","Quality":<value> },
          { "TimeStamp":"<value>","Value":"<value>","Quality":<value> },
          { "TimeStamp":"<value>","Value":"<value>","Quality":<value> }
        ]
     }]
}</pre></td></tr>
<tr><td>Example</td><td><pre>{
     "ErrorCode" : 0,
     "ErrorMessage": null,</pre></td></tr>
</table>

Table 37. Update Sampled Data (continued)

Parameter	Value

	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }`
	`]`
	`}]`
	`}`

Table 38. Access Trend Data

Parameter	Value
Method	GET
URI	https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/trend/<*tag names*>/<*start time*>/<*end time*>/<*direction*>/<*count*>/<interval>
Authorization	Bearer <*token*>
Content type	application/json
Sample URI	https://<*Azure Load Balancer IP of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/trend?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&samplingMode=1&calculationMode=1&direction=0&count=0&intervalMs=1000
Sample cURL commands	curl -i -H "Accept: application/json" -H "Authorization: Bearer <*token*>http://<*Azure Load Balancer IP of Proficy Histori-*

Table 38. Access Trend Data (continued)

Parameter	Value
	`an for Cloud>:8843/historian-rest-api/v1/datapoints/trend/<tag name>/<start time>/<end time>/<direction>/<count>/<interval>`
Query parameters	• **Tag names:** The names of tags whose data you want to retrieve. • **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **Sampling mode:** The sampling mode that you want to use to retrieve data. • **Calculation mode:** The calculation mode that you want to use to retrieve data. • **Interval:** The interval in milliseconds. • **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. • **Count:** The count of the data samples within each calculation interval.
Body	`{ "ErrorCode" : 0, "ErrorMessage": null, "Data":[{ "DataType":"DoubleFloat", "ErrorCode":0, "TagName":"TagName1", "Trend":[ { "TimeStamp":"<value>","Value":"<value>","Quality":<value> } ] }] }`
Example	`{ "ErrorCode" : 0, "ErrorMessage": null,`

Table 38. Access Trend Data (continued)

Parameter	Value

	"Data":[{
	"DataType":"DoubleFloat",
	"ErrorCode":0,
	"TagName":"TagName1",
	
	"Trend":[{ "TimeStamp":"2014-01-01T12:00:00Z","Value":"34.26155","Quality":3 }]
	}]
	}

Table 39. Update Sampled Data

Parameter	Value
Method	POST
URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/trend/<tag names>/<start time>/<end time>/<direction>/<count>/<interval>`
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/trend?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&samplingMode=1&calculationMode=1&direction=0&count=0&intervalMs=1000`
Sample cURL command	`curl -i X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <token>-d { \tagNames\:\<value>\, \start\: \<value>\, \end\: \<value>\, \trend\: <value>, \calculationMode\: <value>, \direction\: <value>, \count\: <value>, \returnDataFields\: <value>, \intervalMs\: <value>, \queryModifier\: <value>, \filterMode\: <value>, \filterExpression\: \<value>\}http://<Azure Load Balancer IP of`

Table 39. Update Sampled Data (continued)

Parameter	Value
	Proficy Historian for Cloud>:9090/historian-rest-api/v1/data-points/sampled
Query parameters	• **Tag names:** The names of tags whose data you want to retrieve. • **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **Interval:** The interval in milliseconds. • **Sampling mode:** The sampling mode using which you want to retrieve data. • **Calculation mode:** The calculation mode using which you want to retrieve data. • **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. • **Count:** The count of the data samples within each calculation interval.

Body	{
	"ErrorCode" : 0,
	"ErrorMessage": null,
	"Data":[{
	"DataType":"<*value*>",
	"ErrorCode":0,
	"TagName":"<*value*>",
	"Samples":
	[
	{ "TimeStamp":"<*value*>","Value":"<*value*>","Quality":<*value*> },
	{ "TimeStamp":"<*value*>","Value":"<*value*>","Quality":<*value*> },
	{ "TimeStamp":"<*value*>","Value":"<*value*>","Quality":<*value*> }
	]

Table 39. Update Sampled Data (continued)

Parameter	Value
	`}]` `}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }`
	`]`
	`}]`
	`}`

Table 40. Limit Query Results

Parameter	Value
Method	GET
URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/configuration/maxDataQueryResultSize=<value>` The minimum number of query result size to enter is 1000.
Authorization	Bearer *<token>*
Content type	application/json

Table 40. Limit Query Results (continued)

Parameter	Value
Sample URI	`https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/configuration/maxDataQueryResultSize=6000`
Sample cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>https://<Azure Load Balancer IP of Proficy Historian for Cloud>:9090/ historian-rest-api/v1/datapoints/configuration?maxDataQueryResultSize=<value>`
Query parameters	**maxDataQueryResultSize:** The maximum number of query results that you want to retrieve.

Chapter 7. Using The Excel Add-In for Historian

About The Excel Add-In for Historian

The Excel Add-In for Historian enhances the power and benefits of using the Historian data archiving and retrieval system.

Features:

- You can add tags to Historian by generating a tag worksheet using the standard Excel tools, editing the parameters, and then importing the information in bulk directly into Historian.
- You can export tag parameters from Excel, make bulk changes using similar techniques, and then import the changes back into Historian.
- You can retrieve selected data from any archive file and include it in a customized report.
- You can plot the data in any of the standard chart formats.
- You can calculate derived variables from raw data values.
- You can perform mathematical functions to smooth or characterize data.
- You can import, export, and modify tags and data — all with familiar Excel commands, macros, and computational techniques.
- You can create dynamic reports that you can share among users.

Excel Add-In Conventions

The Excel Add-In uses several conventions that allow you to take full advantage of the features of the Historian Excel Add-In:

- You can select tags and times either by cell references or by manually entering the values.
- You can select multiple statistics or attributes.
- Specifying an output cell is optional. If you do not specify an output cell, the active cell is used as the starting point for output. When you specify an output cell, that cell is used as the starting point for output. If you select a range for an output cell, the top left cell in the range is used as the starting point for output.
- Specifying an output range determines how many data points are retrieved from a given query. It is important for these functions to specify whether you want the data points to be sorted in ascending or descending order by selecting the appropriate option.
- When you specify an output range or an output cell, ensure that the active cells are not the same cells that you specified with tag name cell references. Otherwise, it will lead to circular cell referencing and incorrect values.

- Specifying data retrieval into rows or columns determines how multiple attributes or statistics are displayed in the worksheet.
- Specifying data retrieval into rows or columns only applies when the window inserts a single function into the worksheet. When you select a multi-cell output range, the orientation of that range determines whether the requested data is returned into rows or columns.
- If no parameters in an Excel formula change, the formula does not recalculate unless you edit the formula. For example, if you change a Hi Scale value from 100 to 50 and then import a tag, the Hi Scale field will still display 100 when looking at the tag information.
- When retrieving data, leave at least one blank line at the top of the output display for the column header labels. If you do not, the header labels will not appear.
- When you retrieve data for more than one tag, if you choose to display the timestamp in the output, then the timestamp will be displayed only once and the parameter values of the selected tags will be shown based on the orientation selected.
- In several fields, an underscore appears at the right side of the field. If you select the underscore, the window instantly changes to a minimized display. You can return to the original display by selecting the box again. The purpose of this feature is to allow you to see an unobstructed view of your worksheet or other windows as you work your way through the window and to allow you to select a cell or range of cells in the worksheet.

Installation

Install the Historian Excel Add-in Using the Installer

Install one of the following 32-bit or 64-bit Microsoft® Excel® applications:

- Microsoft® Excel® 2021 (32 & 64 bit)
- Microsoft® Excel® 2019 (32 & 64 bit)
- Microsoft® Excel® 2016 (32 & 64 bit)
- Microsoft® Excel® 2013 (32 & 64 bit)

This topic describes how to install Excel Add-In using the installer. You can also install it at a command prompt *(on page 37)*.

1. Run the `InstallLauncher.exe` file. Contact the Azure support team for the installer.
2. Select **Historian Excel Add-in**.
   The installer runs through the installation steps.

> **Note:**
> If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Excel Add-In is installed.

Activate Excel Add-In *(on page 186)*.

Install the Historian Excel Add-in at a Command Prompt

1. Install one of the following 32-bit or 64-bit Microsoft® Excel® applications:
    - Microsoft® Excel® 2019
    - Microsoft® Excel® 2016
2. Install Excel Add-in using the installer *(on page 36)* on a machine. When you do so, a template file named `setup.iss` is created at `C:\Windows`. This file stores the installation options that you have provided during the installation. You can then use this template to install Excel Add-in at a command prompt on other machines.

This topic describes how to install the Excel Addin for Historian at a command prompt. You can also install it using the installer *(on page 36)*.

1. Copy the `setup.iss` file to each machine on which you want to install Excel Add-in at a command prompt.
2. In the folder that contains the `setup.iss` file, run the following command: `setup.exe /s /sms`
   The installer runs through the installation steps.

> **Note:**
> If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Excel Add-In is installed.

Activate Excel Add-In *(on page 186)*.

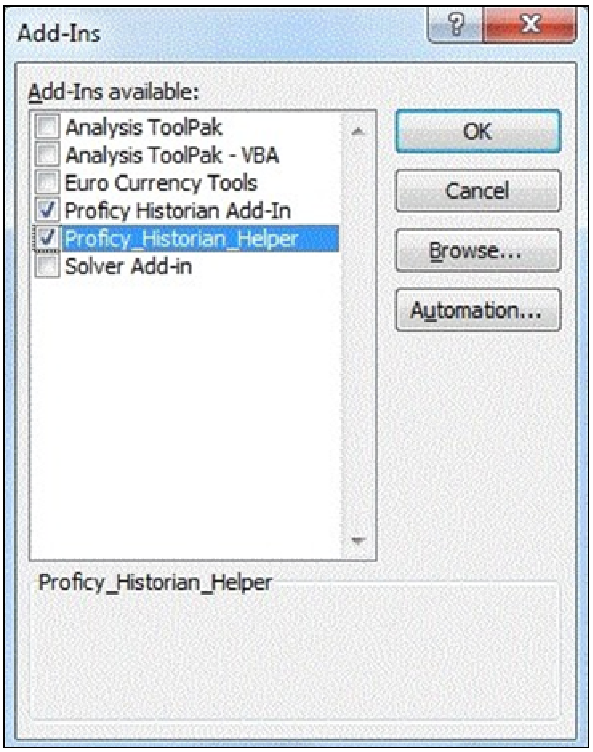
Activate Excel Add-In

Install Excel Add-In *(on page 36)*.

1. Open a new Microsoft Excel worksheet.
2. Select **File > Options**.

   The **Excel Options** window appears.
3. Select **Add-Ins**.
4. In the **Manage** box, select **Excel Add-ins**, and then select **Go**.

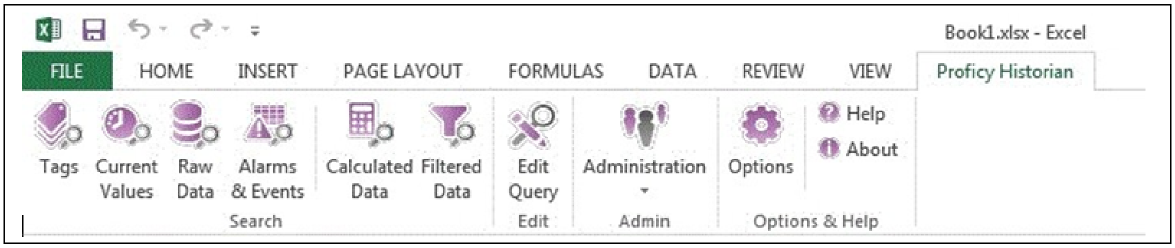
   The **Add-Ins** window appears.

5. Select the **Proficy Historian Add-In** and **Proficy_Historian_Helper** check boxes, and then select **OK**.

   If the **Proficy Historian Add-In** and **Proficy_Historian_Helper** check boxes do not appear, select **Browse** to locate the `Historian.xla` file for the check boxes to appear. This file is created if you have installed Microsoft Excel after installing Excel Add-In. By default, the `Historian.xla` file is located in the `C:\Program Files\Proficy\Historian or C:\Program Files (x86)\Proficy\Historian` folder.

Excel Add-In is now ready to use and the **Proficy Historian** menu is now available in the Microsoft Excel toolbar.



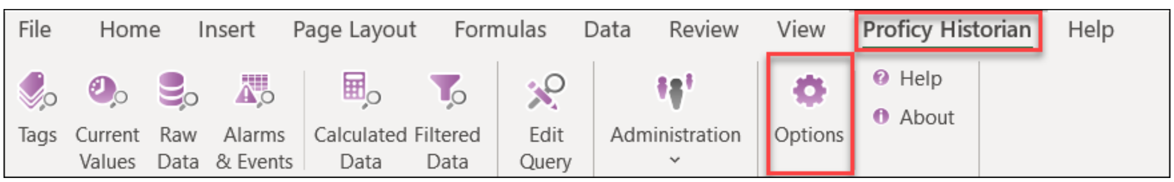
Connect the Excel Add-in with a Historian server *(on page 187)*.

Connect the Excel Add-in with the Historian Server

Activate the Excel Add-in for Historian *(on page 186)*.

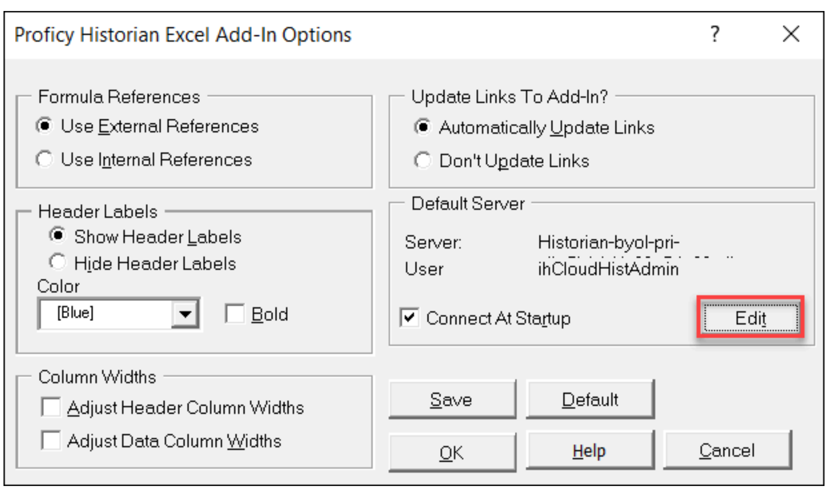
This topic describes how to add an Azure Load Balancer IP in the Excel Add-in for Historian. You can add multiple Azure Load Balancers; however, you can connect with a single Azure Load Balancer at a time.

1. Open an Excel worksheet.
2. Select **Proficy Historian > Options**.

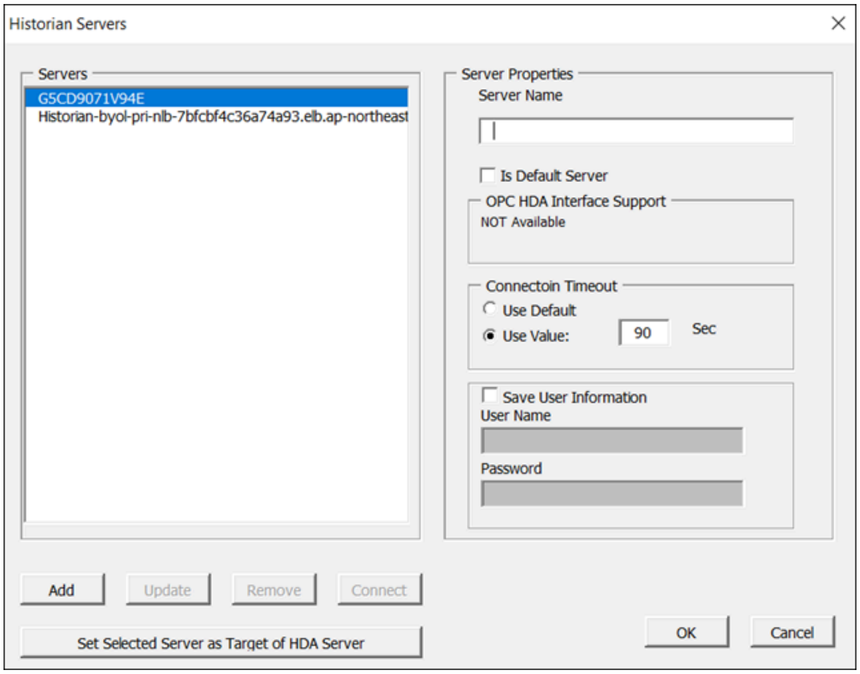


The **Proficy Historian Excel Add-in Options** window appears.

3. In the **Default Server** section, select **Edit**.



The **Historian Servers** window appears.



4. Enter values as described in the following table.

Field	Description
Server Name	Enter the Azure Load Balancer IP. **Tip:** To find the Azure Load Balancer IP: a. Go to the Azure portal. b. Go to the **Resource Group** that was specified during deployment. c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**. d. Select or copy the IP Address.
Is Default Server	Select this check box if you want to set this Historian server as the default one.
Connection Timeout	Specify the time after which the connection will time out.
User Name	Enter the username to connect to Proficy Historian for Azure Cloud.

Field	Description
Password	Enter the password to connect to Proficy Historian for Azure Cloud. **Tip:** This is the value you entered in the **Password** field at the time of deployment.

5. Select **OK**.
The Excel Addin is connected to the Historian server. You can now query data or manage tags.

Querying Data

Query Current Values

You can query the following types of data using the add-in:

- **Current values:** Retrieves the most recently updated value of one or more tags or process variables.

> **Note:**
> If you attempt to perform a query with two worksheets open, the add-in may become unstable and unresponsive. This is a known Microsoft Excel issue. To avoid this issue, work with only one Excel spreadsheet at a time.

- **Raw data:** Raw data values are the values actually stored in the archive, after applying collector and archive compression, but before applying any interpolation, smoothing, or other signal processing calculations. Querying raw data retrieves these values for a selected tag.

In addition, you can query filtered data *(on page 191)* and calculated data *(on page 193)*.

1. Open an Excel worksheet.
2. If you want to query current values, select **Historian > Query Current Value**. If you want to query raw data, select **Historian > Query Raw Data**.
The **Historian Current Value Query** or the **Historian Raw Data Query** window appears.
3. Select the Historian server from the drop-down list box. If you do not specify a server, the default server is considered.

> **Tip:**
> To set the selected server as default, ensure that the **Set Server to Default** option is enabled.

4. Select a tag on your worksheet, and then place the cursor in the **Tag Name** field.
   Optionally, you can select the tag from the **Advance Tag Search** window. For more information, refer to Advanced Tag Search *(on page 203)*.
   The tag name is automatically entered. You can also enter a tag name manually in the **Tag Name** field.
5. Enter values as described in the following table.

Field	Description
Query Type	Select the type of data search: ◦ **By Time**: Using this option, you can search for data values between a start time and an end time. You can also use relative time entries to this field. ◦ **By Number Forward**: Using this option, you can search for a number of values after a specified time. Enter values into the **After Time** and **Number of Values** fields. ◦ **By Number Backward**: Using this option, you can search for a number of values before a specified time. Enter values in the **Values Before Time** and **Number of Values** fields.
Query Criteria String	Enter the query criteria along with the # symbol. For example, if the query criteria string is to retrieve only good data quality values, enter #ONLYGOOD. For more information, see Query Modifiers *(on page 195)*.
Output Display	Select one or more parameters for the output.
Output Range	Select a range of cells in a single row or column to determine where the returned data is placed.
Rows or **Columns**	Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees.

Field	Description
Ascending or **Descending**	Specify the order of the retrieved data.

6. Select **OK**.
   The query returns a number of data points based on the number of rows or columns specified in the output range. If all the data points do not appear, select enough rows or columns to display all the data.

Query Filtered Data

You can filter tag data based on a specific batch ID, lot number, or product code. You can also filter data that meets certain limits (for example, all the data points in which the temperature exceeds a certain value).

When querying filtered data, you can use a **Filter Expression** instead of **FilterTag**, **FilterMode**, and **FilterValue** parameters. You can use multiple filter conditions in the filter expression. For more information and examples on filter expression, refer to *Advanced Topics*.

> **Note:**
> Do not use the **Desc** option for the **Output Range** in the **Filtered Data Query** window. Using this option may cause the Excel Add-In to become unstable. If you use this option and find that Excel is unstable, try minimizing the Excel application window, expose the **Filtered Data Query** window, and close the window. Excel should then function normally.

1. Open an Excel worksheet.
2. Select **Historian > Query Filtered Data**.
   The **Historian Filtered Data Query** window appears.
3. Select the Historian server from the drop-down list box. If you do not specify a server, the default server is considered.

   > **Tip:**
   > To set the selected server as default, ensure that the **Set Server to Default** option is enabled.

4. Select a tag on your worksheet, and then place the cursor in the **Tag Name** field.
   If entering multiple tag names manually, separate each tag name with a colon. If your tag name has a colon within it, then select the tag names via cell references only.

Do not use wildcards in this field. If you use a tag mask instead of a tagname, Historian only returns the first possible match.

Optionally, you can select the tag from the **Advance Tag Search** window. For more information, refer to Advanced Tag Search *(on page 203)*.
The tag name is automatically entered. You can also enter a tag name manually in the **Tag Name** field.

5. Enter values as specified in the following table.

Field	Description
Query Time	Enter the start time and end time for the query. You can also use relative time entries.
Query Criteria String	Enter the query criteria along with the # symbol. For example, if the query criteria string is to retrieve only good data quality values, enter #ONLYGOOD. For more information, see Query Modifiers *(on page 195)*.
Sampling Type	Select the sampling type.
Calculation Field	Select a calculation algorithm. This field is enabled only if you select **Calculated Sampling** in the **Sampling Type** field.
Sampling Interval	Select one of the following options: ◦ **By Interval**: Using this option, you can query the data for a specific interval. For example, if you want to query the data for 10-minute intervals, enter 10 in the **Interval** field, and select **Minutes** in the **Time Unit** field. ◦ **By Samples**: Using this option, you can query the data for a specific number of samples. For example, to query 100 samples, enter 100 in the **Number of Samples** field.
State Value	Enter the state value. This field is enabled only if you selected **Calculated** in the **Sampling Type** field and if you selected **State Count** or **State Time** in the **Calculation Field** field.
Output Display	Select one or more parameters for the output.
Filter Definition	Enter the filter parameters in the available fields.
Output Range	Select a range of cells in a single row or column to determine where the returned data is placed.

Field	Description
Rows or **Columns**	Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees.
Ascending or **Descending**	Specify the order of the retrieved data.

6. Select **OK**.
   The query returns a number of data points based on the number of rows or columns specified in the output range. If all the data points do not appear, select enough rows or columns to display all the data.

Query Calculated Data

You can query data that is the result of performing calculations on raw data.

> **Note:**
> If you attempt to perform a query with two worksheets open, the add-in may become unstable and unresponsive. This is a known Microsoft Excel issue. To avoid this issue, work with only one Excel spreadsheet at a time.

1. Open an Excel worksheet.
2. Select **Historian > Query Calculated Value**.
   The **Historian Calculated Query** window appears.
3. Select the Historian server from the drop-down list box. If you do not specify a server, the default server is considered.

   > **Tip:**
   > To set the selected server as default, ensure that the **Set Server to Default** option is enabled.

4. Select a tag on your worksheet, and then place the cursor in the **Tag Name** field.
   Optionally, you can select the tag from the **Advance Tag Search** window. For more information, refer to Advanced Tag Search *(on page 203)*.
   The tag name is automatically entered. You can also enter a tag name manually in the **Tag Name** field.
5. Enter values as described in the following table.

Field	Description
Query Time	Enter the start time and end time for the query. You can also use relative time entries.
Query Criteria String	Enter the query criteria along with the # symbol. For example, if the query criteria string is to retrieve only good data quality values, enter #ONLYGOOD. For more information, see Query Modifiers *(on page 195)*.
Sampling Type	Select the sampling type.
Calculation	Select a calculation algorithm. This field is enabled only if you select **Calculated Sampling** in the **Sampling Type** field.
Sampling Interval	Select one of the following options: ◦ **By Interval**: Using this option, you can query the data for a specific interval. option displays two entry fields, and . Enter values in both. For example, if you want to query the data for 10-minute intervals, enter 10 in the **Interval** field, and select **Minutes** in the **Time Unit** field. ◦ **By Samples**: Using this option, you can query the data for a specific number of samples. For example, to query 100 samples, enter 100 in the **Number of Samples** field.
State Value	Enter the state value. This field is enabled only if you selected **Calculated** in the **Sampling Type** field and if you selected **State Count** or **State Time** in the **Calculation Field** field.
Output Display	Select one or more parameters for the output.
Output Range	Select a range of cells in a single row or column to determine where the returned data is placed.
Rows or **Columns**	Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees.
Ascending or **Descending**	Specify the order of the retrieved data.

6. Select **OK**.
   The query returns a number of data points based on the number of rows or columns specified in the output range. If all the data points do not appear, select enough rows or columns to display all the data.

Modify a Query

You can change query parameters such as tag name, start time, end time, and so on. You cannot, however, narrow down the output range. For example, you cannot reduce the number in the **NumberOfSamples** field, or you cannot change the **Output Orientation** to values that result in fewer rows or columns.

1. Open an Excel worksheet.
2. Access the query that you want to modify.
3. In the **Add-In** drop-down list box, select **Edit Query** or  icon. Or you can double-select any cell that has the query formula.
   The **Edit Query** window appears.
4. Modify the query, and then select **OK**.

Query Modifiers

Query modifiers are used to retrieve data that has been stored in the archive. They are used along with sampling and calculation modes to get a specific set of data.

If you want to use a query modifier, when you create or modify a query, in the **Query Criteria String** field, enter #, and then enter the query modifier. For example, if you want to retrieve only good data quality values, enter #ONLYGOOD.

Query Modifier	Results
`ONLYGOOD`	The `ONLYGOOD` modifier excludes bad and uncertain data quality values from retrieval and calculations. Although you can use this modifier with any sampling or calculation mode, it is most useful with raw and current Value queries. All the calculation modes such as minimum or average exclude bad values by default, so this modifier is not required with those cases.
`INCLUDEREPLACED`	Normally,when you query raw data, any values that have been replaced with a different value for the same timestamp are not returned.

Query Modifier	Results
	The `INCLUDEREPLACED` modifier is used to specify that you want replaced values to be returned, in addition to the updated values. However, you cannot query only the replaced data and the retrievable values that have replaced the other values. You can query all currently visible data and get the data that has been replaced. This modifier is only useful with the rawbytime or rawbynumber retrieval. Do not use it with any other sampling or calculation mode.
`INCLUDEDELETED`	The `INCLUDEDELETED` modifier retrieves the value that was previously deleted. Data that has been deleted from the archiver is never actually removed but is marked as hidden. Use the `INCLUDEDELETED` modifier to retrieve the values that were deleted, in addition to the current values. This modifier is only useful with the rawbytime or rawbynumber retrieval. Do not use it with any other sampling or calculation mode.
`ONLYIFCONNECTED/ONLYIFUPTODATE`	The `ONLYIFCONNECTED` and `ONLYIFUPTODATE` modifiers can be used on any sampling or calculation mode to retrieve bad data if the collector is not currently connected and sending data to the archiver. The bad data is not stored in the IHA file but is only returned in the query. If the collector reconnects and flushes data and you run the query again, the actual stored data is returned in the following situations: • Collector loses connection to the archiver • Collector crashes • Collector compression is used and no value exceeds the deadband
`ONLYRAW`	The `ONLYRAW` modifier retrieves only the raw samples. It does not add interpolated or lab sampled values at the beginning of each interval during calculated retrieval such as average, minimum, or maximum. Normally, a data query for minimum value will interpolate a value at the start of each interval and use that together with any raw samples to determine the minimum value in the interval. Interpolation is necessary because some intervals may not have any raw samples stored. Use this query modifier with calculation modes only, not with raw or sampled retrieval like interpolated modes.

Query Modifier	Results
`LABSAMPLING`	The `LABSAMPLING` modifier affects the calculation modes that interpolate a value at the start of each interval. Instead of using interpolation, lab sampling is used. When querying highly compressed data you may have intervals with no raw samples stored. For example, an average from 2 pm to 6 pm on a one-hour interval will interpolate a value at 2 pm, 3 pm, 4 pm, and 5 pm, and uses those in addition to any stored samples to compute averages. When you specify `LABSAMPLING`, the lab sampling mode is used instead of the interpolated sampling mode to determine these hourly values. A lab sampled average is used when querying a tag that never ramps up but changes in a step pattern such as a state value or a set point. Use this query modifier with calculation modes only, not raw or sampled retrieval like interpolated modes.
`INCLUDEBAD`	Normally, when you query calculated data from Historian, only good data quality raw samples are considered. `INCLUDEBAD` modifier includes bad data quality values in calculations. You can use `INCLUDEBAD` with any sampling or calculation mode.
`FILTERINCLUDEBAD`	Normally, while filtering, we use only good data quality values. When we use `FILTERINCLUDEBAD`, the bad data quality values are considered when filtering to determine time ranges. This query modifier is not always recommended.
`USEMASTERFIELDTIME`	The `USEMASTERFIELDTIME` query modifier is used only for the MultiField tags. It returns the value of all the fields at the same timestamp of the master field time, in each interval returned.
`HONORENDTIME`	Normally, a query keeps searching through archives until the required number of samples has been located, or until it gets to the first or last archive. However, there are cases where you would want to specify a time limit as well. For example, you may want to output the returned data for a `RawByNumber` query in a trend page, in which case there is no need to return data that would be offpage. If you want to specify a time limit, provide an end time in your `RawByNumber` query and include the `HONORENDTIME` query modifier. Since `RawByNumber` has direction (backwards or forwards), the end time must be older than the start time for a backwards direction or later than the start time for a forwards direction. Use this query modifier only with the `RawByNumber` sampling mode.

Query Modifier	Results
`EXAMINE-FEW`	Queries using calculation modes normally loop through every raw sample, between the given start time and end time, to compute the calculated values. When using `FirstRawValue`, `FirstRawTime`, `LastRawValue`, and `LastRawTime` calculation modes, we can use only the raw sample near each interval boundary and achieve the same result. The `EXAMINEFEW` query modifier enables this. If you are using one of these calculation modes, you may experience better read performance using the `EXAMINEFEW` query modifier. Using this query modifier is recommended when: • The time interval is great than 1 minute. • The collection interval is greater than 1 second. • The data node size is greater than the default 1400 bytes. • The data type of the tags is String or Blob. Query performance varies depending on all of the above factors. Use this query modifier only with `FirstRawValue`, `FirstRawTime`, `LastRawValue`, and `LastRawTime` calculation modes.

Export Data

The Export Data function allows you to move values from the Historian Server to your Excel worksheet or to another system in the same way you move tag information with Export Tags.

> **Note:**
> Before importing or exporting tags or data, you should be aware of a convention used with the Historian application. The Server is the reference point for all import and export functions. If you want to move tag information from the Server into your worksheet, you must use the **Export Tags** command. Conversely, if you want to move data from your worksheet to the server, you must use the **Import Data** command.

1. Select **Administration > Export Raw Data** from the **Historian** menu.
   The **Export Data from Historian** window appears.
2. If you want to specify a server, select a server from the drop down list. If you do not specify a server, the Add-In uses the default server.
3. Select a tag on your worksheet or enter the tag names manually.

> **Note:**
> If your tag name has a colon within it, then you should select the tag names via cell references only.

4. Optionally, you can select the tag name from the **Advance Tag Search** window.
   See Search for a Tag (Advanced) *(on page 203)*
5. In the **Query Criteria String**, enter the query criteria along with the # symbol.
   For example, if the query criteria string is to retrieve only good data quality values, then you should specify `#ONLYGOOD` as the **Query Criteria String**. See Query Modifiers *(on page 195)*.
6. In the **Query Time** section enter values of time in the **Start Time** and **End Time** fields.
   You can also use relative time entries to this field. See Relative Time Entries *(on page 223)*.
7. In the **Sampling Type** section, select a type from the drop-down list.
8. The **Calculation** field is active only after you select **Calculated Sampling** as the **Sample Type**.
   Select a **Calculation Algorithm** type from the drop-down list.
9. In the **Sampling Interval** section, select either the **By Interval** or **By Samples** option.
   The **By Interval** option displays two entry fields, **Interval** and **Time Unit**. Enter values in both. For example, to sample at 10 minute intervals, enter 10 in the interval field and select Minutes in the Time Unit field. The **By Samples** option displays a **Number of Samples** field.

   To specify a number of samples for the data query, enter a number in this field. For example, to query 100 samples, enter 100 in this field.
10. In the **Filter Definition** section, enter filter parameters in the fields for **Filter Tag**, **Filter Comparison**, **Include Date Where Value Is Equal To**, and **Include Times**.
    These fields are optional. If you do not enter any values, the query returns all values without filtering.
11. In the **Fields To Export** section, select one or more fields.
    To select multiple individual tags, press the **Control** key and select the tagnames. To select a sequence of tags, press the **Shift** key and select the first and last tagname of the sequence.
12. In the **Export Options** section, select one of three options:
    - **To New Worksheet**
    - **To CSV File** or
    - **To XML File**
13. If you select **To CSV File** or **To XML File**, you must enter a file name and path for the new file in the **File Name** field.
14. Select **OK** to initiate the export. Select **Cancel** to abort the operation and close the window.

Import Data

The **Import Data** command is the converse of the **Export Data** command. It moves selected information from your current worksheet into the specified Server in the same way the **Import Tags** command functions.

> **Note:**
> If you use the **Active Hours** setting while importing data using the Excel Add-In, note that if the first tags imported are not within the **Active Hours** settings, no subsequent tags will be returned on that import (even if they are within the set active hours).

1. Select **Administration** and then select **Import Data** from the **Historian** menu.
   A message box appears.
2. Select **Yes** to initiate the operation. If successful, a window appears confirming the completion of the import function.
   Select **OK** to close the window. If errors occur on the import, a window appears detailing the issues encountered in the import. If an error occurs in any line of the import, the whole import is aborted.

Access Archive Statistics

You can access a list of selected statistics about an archive file. You can specify the server, the archive file name, and the type of information you want to access (such as start time, end time, file name, target file size, current file size, and current or read-only status). You can also specify a range of cells for the display.

1. Open an Excel worksheet.
2. Select **Historian > Administration > List Archives**.
   The **Historian Archive List** window appears.
3. Select a server from the drop-down list. If you do not specify a server, the default server is considered.
4. Enter values as described in the following table.

Field	Description
Archive Name	Enter a archive name. Do not use wildcards in this field. **Tip:** To return details for more than one item, specify a sub-string in the **Archive Name** field that exists in each

Field	Description
	archive you want listed. For example, if you have archive files named from `Hero5_Archive001` to `Hero5_Archive010`, enter `Hero5_Archive` to return the details for all those archives.
Output Display	Select one or more parameters for the output display.
Output Range	Select a range of cells in a single row or column to determine where the returned data is placed.

5. Select **Asc** or **Desc** to sort the archives in ascending or descending order.
6. Select either **Columns** or **Rows** for the output display.

> **Note:**
> When selecting multiple tags, the orientation of the return data is based on the orientation of the selected tags and the Row/Col selection is ignored.

7. Select **OK**.
   The statistics of the selected archives appear.

Access Collector Statistics

You can access a list of selected statistics of a collector instance. You can specify the server, the collector instance, and the type of information you want to access. You can also specify the range of cells for the display.

1. Open an Excel worksheet.
2. Select **Historian > Administration > List Collectors**.
   The **Historian Collector List** window appears.
3. Select a server from the drop-down list. If you do not specify a server, the default server is considered.
4. Enter values as described in the following table.

Field	Description
Collector Name	Enter a collector instance name. Do not use wildcards in this field.

Field	Description
	Tip: To return details for more than one item, specify a substring in the **Collector Name** field that exists in each collector you want listed. For example, if you have collectors named from `Hero5_Collector0001` to `Hero5_Collector010`, enter `Hero5_Collector` to return the details for all those collectors.
Output Display	Select one or more parameters for the output display.
Output Range	Select a range of cells in a single row or column to determine where the returned data is placed.

5. Select either **Columns** or **Rows** for the output display.

> **Note:**
> When selecting multiple tags, the orientation of the return data is based on the orientation of the selected tags and the Row/Col selection is ignored.

6. Select **OK**.
   The statistics of the selected collector instances appear.

Managing Tags

Search for a Tag (Basic)

You can search for tags and perform actions on them.

This topic describes how to perform a basic search of tags. You can also perform an advanced search *(on page 203)*.

1. Open an Excel worksheet.
2. Select **Historian > Search Tags**.
   The **Historian Tag Search** window appears.
3. In the **Server** field, select a server from the drop-down list. If you do not specify a server, the default server is considered.
4. In the **Tag Mask**, enter a wildcard character to search for tags (for example, *).

5. Select **Search**.
   The **Historian Tag Search** window is populated with a tag list.
6. Move tags from the left section to the right section to add them to the search query.
7. Use the **Search Display** section to choose whether you want to display tag names or tag description. It also displays the number of tags returned.
8. Use the **Output With** to choose whether the output shows the names of the selected tags or the cell computation formulas.
   You can use the **Output with Formula** to place a dynamic formula in the worksheet instead of just copying the selected tag names. When you do so, the list of tags returned are dynamic based on the tag mask criteria. This is useful when selecting a cell reference for the tag mask as opposed to typing in a tag mask directly in the window.
9. Use the **Output Range** field to determine where in the worksheet the output data must appear.
10. Use the **Output Display** section to select the type of data to be displayed.
11. Select **OK** to apply your choices and initiate the query.
    A list of tags appears based on your search criteria.

Search for a Tag (Advanced)

You can search for tags and perform actions on them.

This topic describes how to perform a advanced search of tags. You can also perform a basic search *(on page 202)*.

When you perform an advanced search, the most recently used search criteria are saved in a file named `DefaultSearchCriteria.xml` in `c:\user- s\<username>\AppData`. These criteria are automatically loaded into the window the next time you access the Excel worksheet. You can reuse or modify the criteria rather than entering them each time. If you want to reset your criteria, delete the XML file.

While performing an advanced search, you can:

- Add multiple search criteria.
- Modify the existing criteria.
- Delete the unwanted search criteria from the list.
- Save the criteria to a file and reuse it.
- View the details of a tag in the search results.

1. Open an Excel worksheet.
2. Select **Historian > Search Tags > Advanced Tag Search**.
3. In the **Tag Criteria** field, specify one or more tag criteria *(on page 230)*.
4. Provide values in the **Tag Criteria Value** field.

5. Select **Add Criteria**.

   The criteria are listed in the **Search Criteria** section.
6. Select **Search**.

   All tags that satisfy the query criteria are displayed in the **Available** section.
7. Move tags from the **Available List** section.
8. To modify the **Tag Criteria Value** already entered:
   a. Double-click the criteria from the list.
   b. Change the **Tag Criteria Value**.
   c. Select **Update Criteria**. The criteria value is updated with the new value.
9. To delete the search criteria from the list, select the criteria from the list, and then select **Delete**.
10. To save a search criteria list to be reused:

    a. Create your search criteria list.

    b. Select **Save**.

       The **Save As** window appears.

    c. Enter the file name, and select **Save**.

       Your criteria list is saved.

11. To load an existing criteria list:

    a. Select **Load**.

       The **Open** window appears.

    b. Choose the XML file you saved earlier, and then select **Open**.

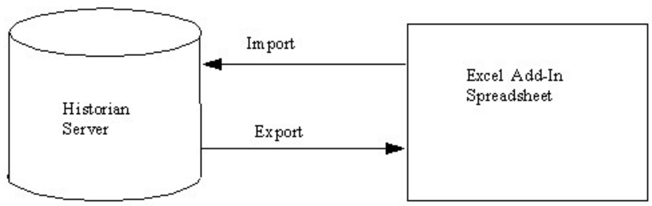
       The criteria list is loaded to the **Advanced Tag Search** window.

12. To view the tag attributes, double-click the tag from the available section or from the selected section.

    The **Tag Attributes** window appears with the attribute details.
13. Select **OK**.

Export Tags

You can export tags from a Historian server into an Excel worksheet or to another system (either local or remote). After you export tags into an Excel worksheet, you can add/modify tags *(on page 206)* in bulk, and then import them *(on page 206)*.



> **Note:**
> You cannot enter more than 32,767 characters in a single cell in an Excel worksheet.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Export Tags**.
   The **Export Tags from Historian** window appears.
3. Select a server from the drop-down list. If you do not select a server, the add-in uses the default server.
4. Enter values as described in the following table.

Field	Description
Filter Criteria	Enter the name or description of the tag you want to export. You can use a tag mask to select a group of tags. To select a tag, use cell references instead of manually typing them. **Note:** You cannot export multiple tags when tagnames are read from multiple cells. If you specify a range of tag-names to read from multiple cells in the **Tag Mask** or **Tag Name(s)** fields, only the first tag in the range will be exported.
Collector	Enter the collector name.
Data Type	Enter the data type.

5. Select one or more field names from the list in the right hand window. Always include tag names in the list of fields to export.

6. In the **Export Options** section, specify whether you want to export tags into a new Excel worksheet, a CSV file, or an XML file. If you select CSV or XML, you must also enter a path and file name for the destination file.
7. Select **OK**.
   The data is exported.

Add/Modify Tags

Using the add-in, you can add tags to Historian or modify existing tags. To do so, include the tags in an Excel worksheet either automatically or manually, and then import them in bulk into Historian.

This can be a very convenient mechanism when you are working with large numbers of tags. If any conflicting names or parameters occur, an error occurs; you can then resolve the conflict and try again.

1. Create a tags worksheet in Excel either manually or automatically (using macros or any other tools).
   Since Historian requires information about each tag that varies with the type of the tag, ensure that you have included all the required information in the worksheet before attempting to import it into Historian. To determine what specific tag information is required, refer to the documentation provided with your SCADA application.
2. Import the tags into Historian *(on page 206)*.

> **Note:**
> If any errors on the import occur, a window appears, specifying the issues encountered during the import. If an error occurs with any line of the import, the whole import is aborted.

Import Tags

In an Excel worksheet, add/modify the tags that you want to import *(on page 206)*.

Using the add-in, you can add tags to Historian or modify existing tags. To do so, include the tags in an Excel worksheet either automatically or manually, and then import them in bulk into Historian (either local or remote).

> **Note:**
> Do not add or update the spare configurations as the data may get corrupted or overwritten. For example, the **Spare 5** field is used by the Server-to-Server collector for internal purposes.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Import Tags**.
   A message appears.
3. Select **Yes** to initiate the operation.
   A message appears, confirming that the import is complete.
4. Select **OK**.
   If errors occur, a window appears detailing the issues encountered during the import. If an error occurs with any line of the import, the whole import operation is aborted.
   f you export all the fields and attempt to import the read-only fields **LastModified** and **LastModifiedUser**, you may receive the following error message: Import failed, Error with Import Header. To avoid this issue, export the tags without selecting the read-only fields, and then import the tags.

Rename Tags

To rename a tag, you must be a member of the administrator's group with tag-level security.

When you rename a tag, you can choose between the following options:

- **Rename using an alias:** In this case, the old name is called the tag alias. You can retrieve tag data using the tag alias as well. When you copy a tag, the tag alias is captured as well to aid in an audit trail.
- **Rename permanently:** In this case, the old name is no longer captured. Therefore, you can create another tag with this old name. You cannot store and forward data using the old name. This implies that data for the tag is collected separately for the new name.

1. Export the tags *(on page 204)* that you want to rename.

   > **Important:**
   > You must only include tag name in the list of fields to export.

2. In the Excel worksheet, to the right of the **Tagname** column, insert a column named New Tagname.
3. For each tag that you want to rename, enter the new name in the **New Tagname** column.

   > **Important:**
   > You must specify a tag name in all the rows of the **New Tagname** column. If you do not want to rename any of those exported tags, you must delete that row.

4. If you want to rename the tags permanently, to the right of the **New Tagname** column, insert a column named **Permanent Rename**.
5. For each tag that you want to rename permanently, enter TRUE in the **Permanent Rename** column. For the remaining tags, enter FALSE.
6. Select **Historian > Administration > Rename Tags**.
   A message appears, asking you to confirm that you want to rename the tags.
7. Select **Yes**.
   The tags are renamed.

Reference

Excel Add-In Options

Field	Description
Internal vs. External References	Choosing **Use External References** allows your application to reference cells in other worksheets and workbooks in addition to the current one. If you choose **Use Internal References** instead, you can only access cells in the current worksheet. The default setting is **Use External References**.
Automatically Update Links to Add- In (Yes/ No)	Add-In functions are maintained as worksheet links. If users who share worksheets do not have Microsoft Office installed the same way, it is necessary to turn this feature on. When on, this feature automatically re-establishes any formula links that may be broken due to differences among users in Microsoft Office installation. The default setting enables this feature. The Auto Update feature allows sharing of worksheets. You must, however, install the Excel Add-In in the exact same Microsoft Office Library Path as the other worksheets if you want to use the sharing feature. When opening a worksheet with links to another worksheet, you may receive a message prompting you to update all linked information in the workbook (Yes) or keep the existing information (No). It is recommended that you select No and keep the existing information. The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated.
Show/Hide Header Labels	This option lets you display or suppress the column header labels that are automatically placed in the worksheet when entering formulas throughout the Historian windows. The default setting is **Show Labels**.

Field	Description
Color	Allows you to select the header name color from the drop-down list: black, blue, red, green, magenta, cyan, or yellow.
Assign Default Server	This window shows the current server assignment. You can modify the setting by selecting the **Edit** button and accessing the **Historian Server Managers** window. This window allows you to save user connection information, add or connect to a new server, delete a server, and modify the default server.
Adjust Column Widths	This option lets you automatically adjust the width of columns in your worksheet as formulas are inserted by Historian windows. Select **Adjust Header Column Width** to modify the width of header labels; select **Adjust Data Column Width** to modify the data column widths to accommodate the data values. Enabling these options usually makes the worksheet much more readable. However, doing so can sometimes make the worksheet calculate too much when building a large report. In such cases, disable the automatic feature and adjust individual columns manually.
Save/Default/Cancel	These action buttons let you apply your choices of options. Select **Save** to apply the settings you entered, select **Default** to select default settings for all options, and select **Cancel** to close the window.

Reports

You can generate a wide range of custom reports. You can use all the standard, familiar Excel tools and techniques to access the Historian archives and build reports and charts of all types to fit your specific needs. You can use the sample reports included with Historian almost as is — just change the tags to fit your application. As an alternative, use the setup worksheets as a starting point and adapt them to your particular situation.

Defining Reports

You can define a report so that Excel recalculates the worksheet whenever the contents of specific cells, such as start times or dates, change. In this way, the report generates a dynamic snapshot of process performance, updated regularly in real time. You can also manually initiate recalculation at anytime.

Building Dynamic Reports

The primary rule to follow in building a dynamic report is to use formulas with cell references that contain variable information rather than fixed data, so that recalculation produces new data each time it occurs. You then initiate recalculation by changing certain inputs manually or automatically.

Sharing Reports

You can share any Excel reports you develop with the Historian Excel Add-In as you would any other Excel workbook. For each client using the worksheets, set up the Excel Add-In for Historian.

Using the Sample Reports

The Historian application includes three typical sample reports that demonstrate the power and ease-of-use of the Excel Add-In. Use them directly in your application or modify them to fit your requirements.

The three sample Excel reports are built using tags from the Simulation collector. You must create an instance of the Simulation collector and start it in order for these reports to work. The `Historian Batch Report Sample.xls` file also uses Batch ID and Product ID tags from the Simulation collector. These are Simulation Collector points that are configured to store string data types.

To ensure that the sample reports work correctly, you must add the string tags. These are the last five tags in the tag collector list. Add the string tags by browsing the Simulation collector and adding all of the tags by selecting the **Add All Tags** check box. Alternatively, you can run the `Add Tags to Simulation Collector.bat` batch file in the `Historian\Server` directory of the machine that has the Simulation collector.

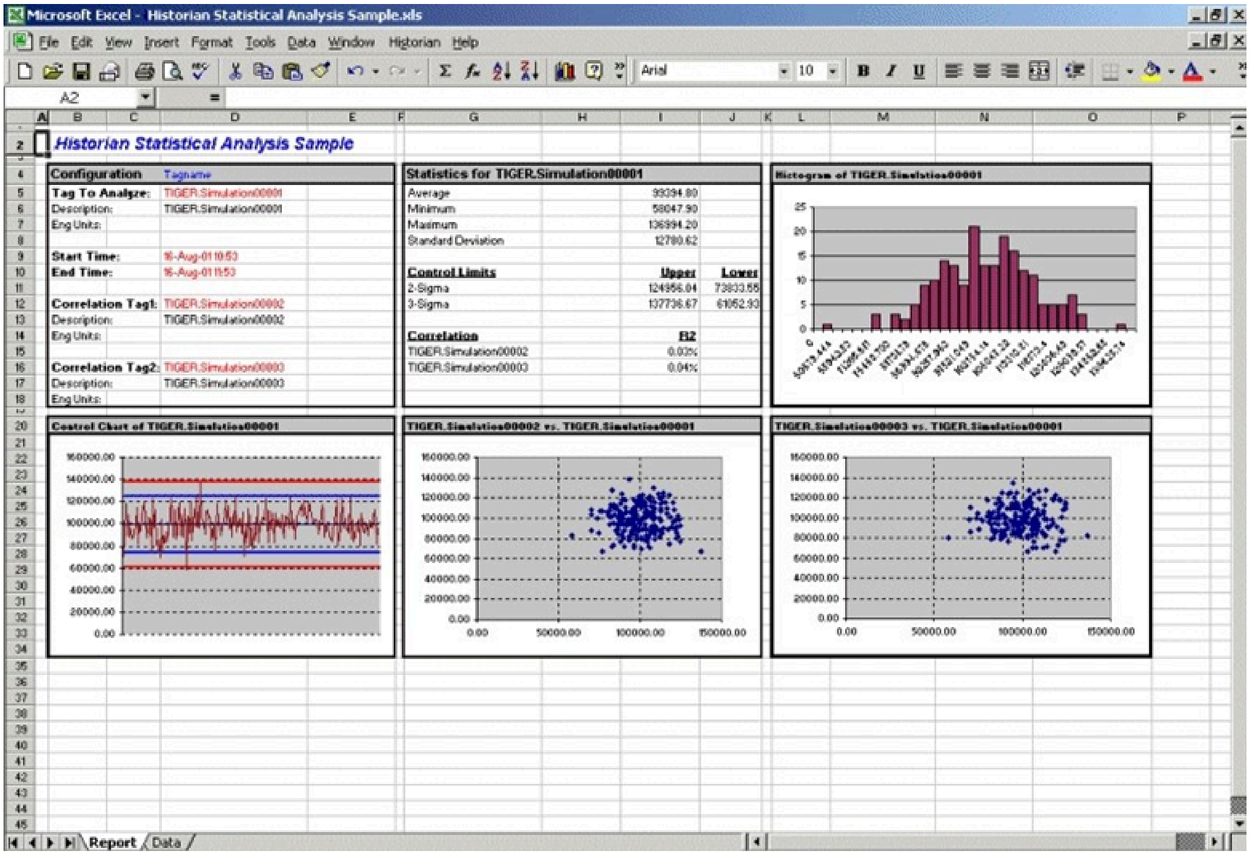
In addition, when you create an instance of the Simulation collector, it prompts you for the number of simulation tags it should create (but you must still add the tags for collection using one of the two methods above). The default is 1000. Do not enter a value less than 30.

When opening a sample Excel report, you may receive a message prompting you to update all linked information in the workbook or keep the existing information. It is recommended that you select No (that is, keep the existing information). The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated.

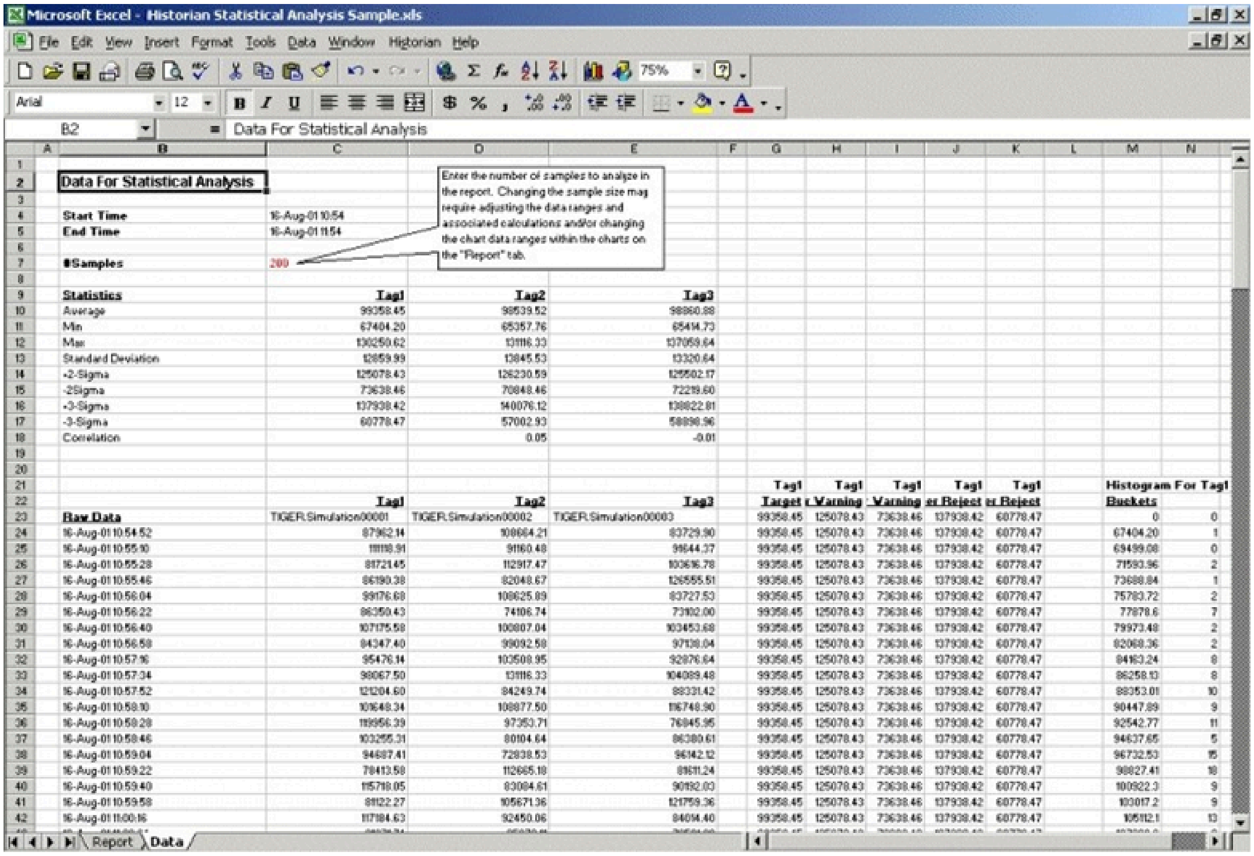
Historian Statistical Analysis Sample Report

For a specific duration, this report calculates a number of statistical properties of a tag, such as the average, maximum, minimum, standard deviation, 2 sigma and 3 sigma control limits, and correlation coefficients for other tags. It displays charts of various types for several of these variables.

The chart at the lower left is a plot of the main variable vs. time with sigma control limits indicated by the straight lines. The two charts to the right are scatter diagrams that show the correlation between the main variable and two other variables. The chart at the top right is a histogram of data values of the main variable that shows how the data points are distributed.

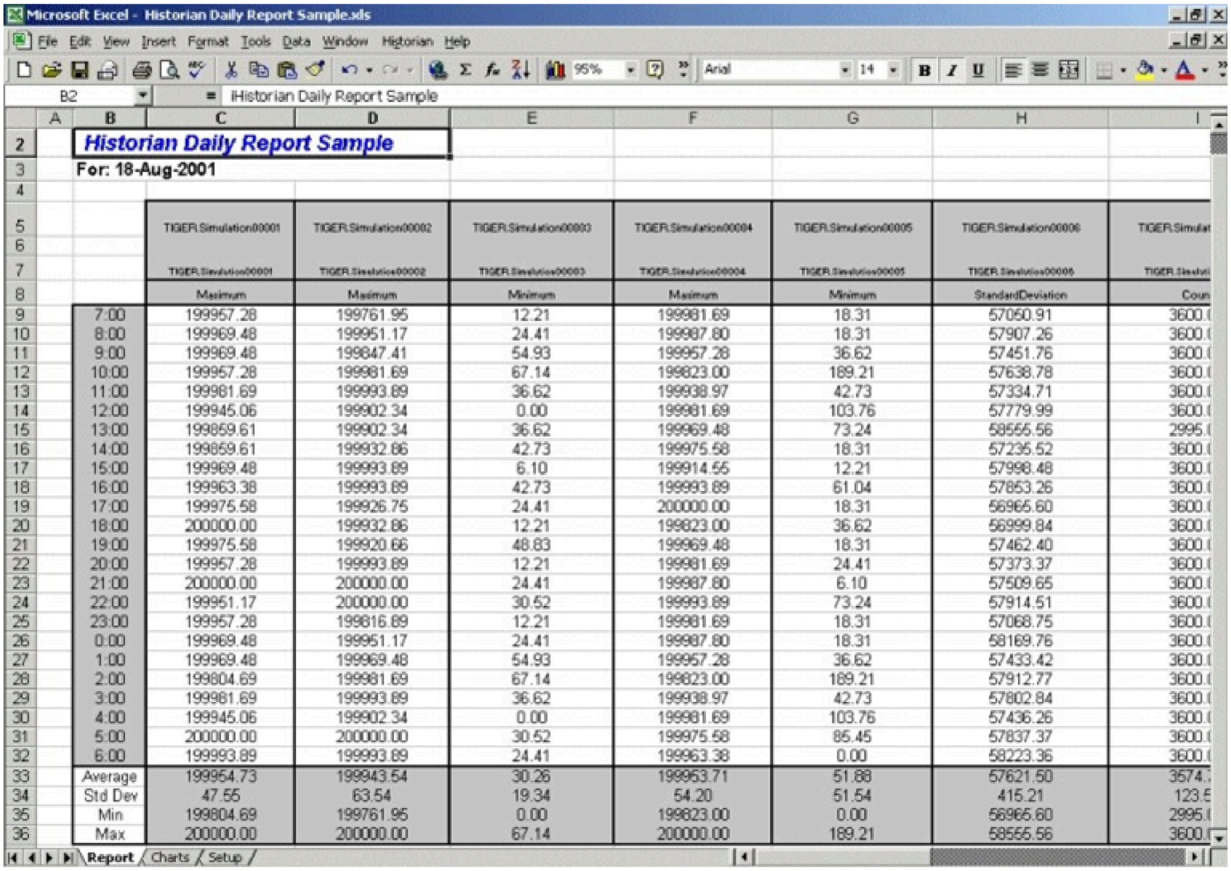


The following figure shows the worksheet associated with the sample report that contains the data used to generate the report.



Daily Performance Sample Report

This sample report shows how the measured values and selected statistical properties of specified tags have varied in the last 24 hours. This sample is an example of a typical daily performance report in an industrial plant.

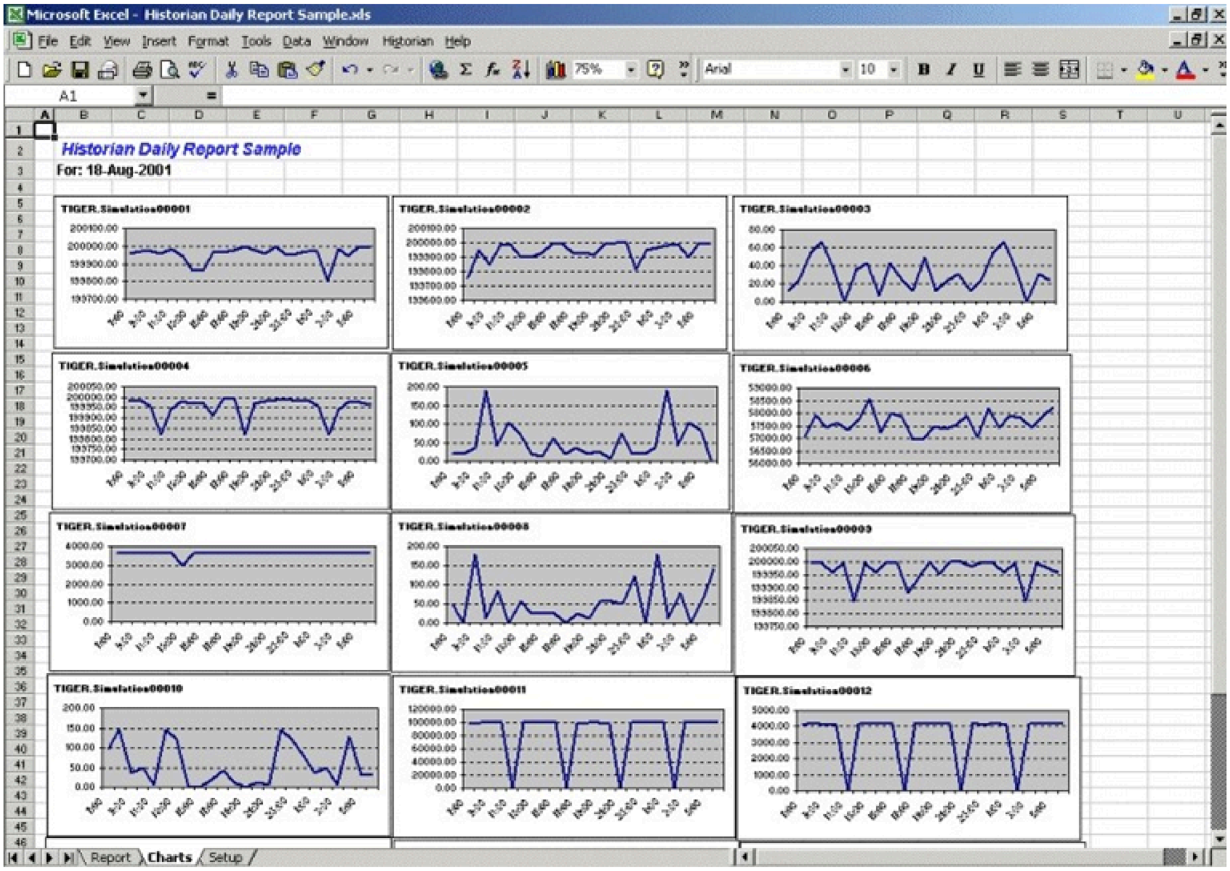


Historian Daily Report Sample

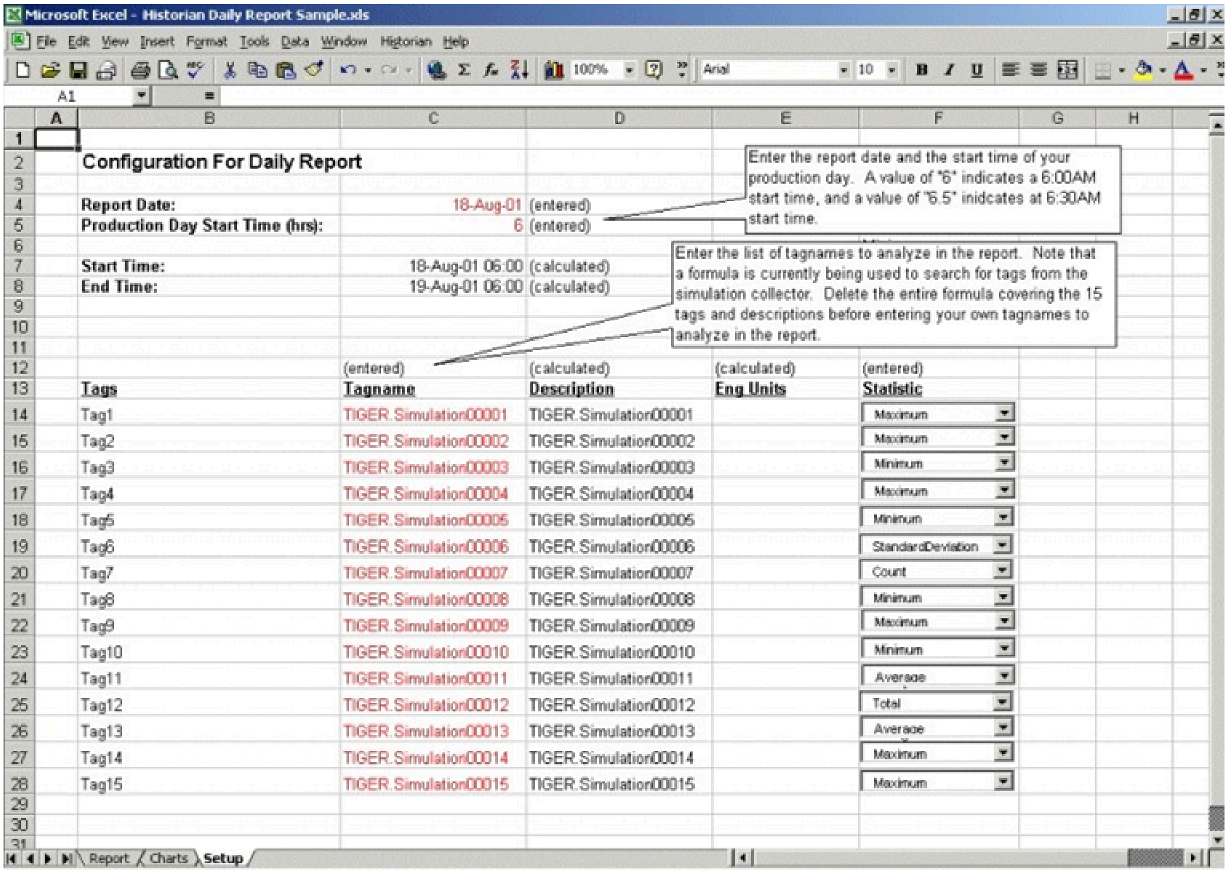
For: 18-Aug-2001

	TIGER.Simulation00001	TIGER.Simulation00002	TIGER.Simulation00003	TIGER.Simulation00004	TIGER.Simulation00005	TIGER.Simulation00006	TIGER.Simulat
	TIGER.Simulation00001	TIGER.Simulation00002	TIGER.Simulation00003	TIGER.Simulation00004	TIGER.Simulation00005	TIGER.Simulation00006	TIGER.Simulati
	Maximum	Maximum	Minimum	Maximum	Minimum	StandardDeviation	Coun
7:00	199957.28	199761.95	12.21	199981.69	18.31	57050.91	3600.
8:00	199969.48	199951.17	24.41	199987.80	18.31	57907.26	3600.
9:00	199969.48	199847.41	54.93	199957.28	36.62	57451.76	3600.
10:00	199957.28	199981.69	67.14	199823.00	189.21	57638.78	3600.
11:00	199981.69	199993.89	36.62	199938.97	42.73	57334.71	3600.
12:00	199945.06	199902.34	0.00	199981.69	103.76	57779.99	3600.
13:00	199859.61	199902.34	36.62	199969.48	73.24	58555.56	2995.
14:00	199859.61	199932.86	42.73	199975.58	18.31	57235.52	3600.
15:00	199969.48	199993.89	6.10	199914.55	12.21	57998.48	3600.
16:00	199963.38	199993.89	42.73	199993.89	61.04	57853.26	3600.
17:00	199975.58	199926.75	24.41	200000.00	18.31	56965.60	3600.
18:00	200000.00	199932.86	12.21	199823.00	36.62	56999.84	3600.
19:00	199975.58	199920.66	48.83	199969.48	18.31	57462.40	3600.
20:00	199957.28	199993.89	12.21	199981.69	24.41	57373.37	3600.
21:00	200000.00	200000.00	24.41	199987.80	6.10	57509.65	3600.
22:00	199951.17	200000.00	30.52	199993.89	73.24	57914.51	3600.
23:00	199957.28	199816.89	12.21	199981.69	18.31	57068.75	3600.
0:00	199969.48	199951.17	24.41	199987.80	18.31	58169.76	3600.
1:00	199969.48	199969.48	54.93	199957.28	36.62	57433.42	3600.
2:00	199804.69	199981.69	67.14	199823.00	189.21	57912.77	3600.
3:00	199981.69	199993.89	36.62	199938.97	42.73	57802.84	3600.
4:00	199945.06	199902.34	0.00	199981.69	103.76	57436.26	3600.
5:00	200000.00	200000.00	30.52	199975.58	85.45	57837.37	3600.
6:00	199993.89	199993.89	24.41	199963.38	0.00	58223.36	3600.
Average	199954.73	199943.54	30.26	199963.71	51.88	57621.50	3574.
Std Dev	47.55	63.54	19.34	54.20	51.54	415.21	123.5
Min	199804.69	199761.95	0.00	199823.00	0.00	56965.60	2995.
Max	200000.00	200000.00	67.14	200000.00	189.21	58555.56	3600.

The report shown in the following image is a collection of chart plots of the data displayed in the report of the previous image.

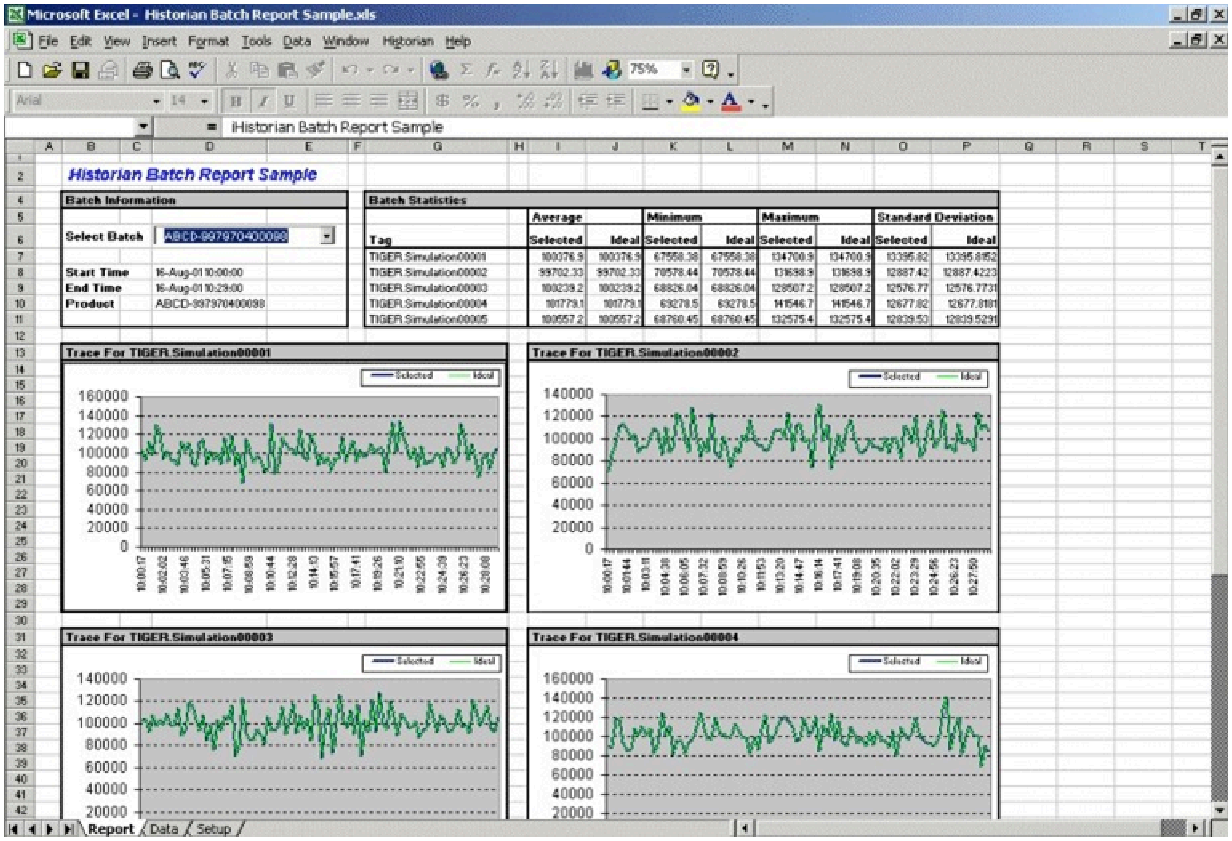


The following figure shows the worksheet used to set up the Daily Sample Report. Edit the worksheet to adapt this report to your application.

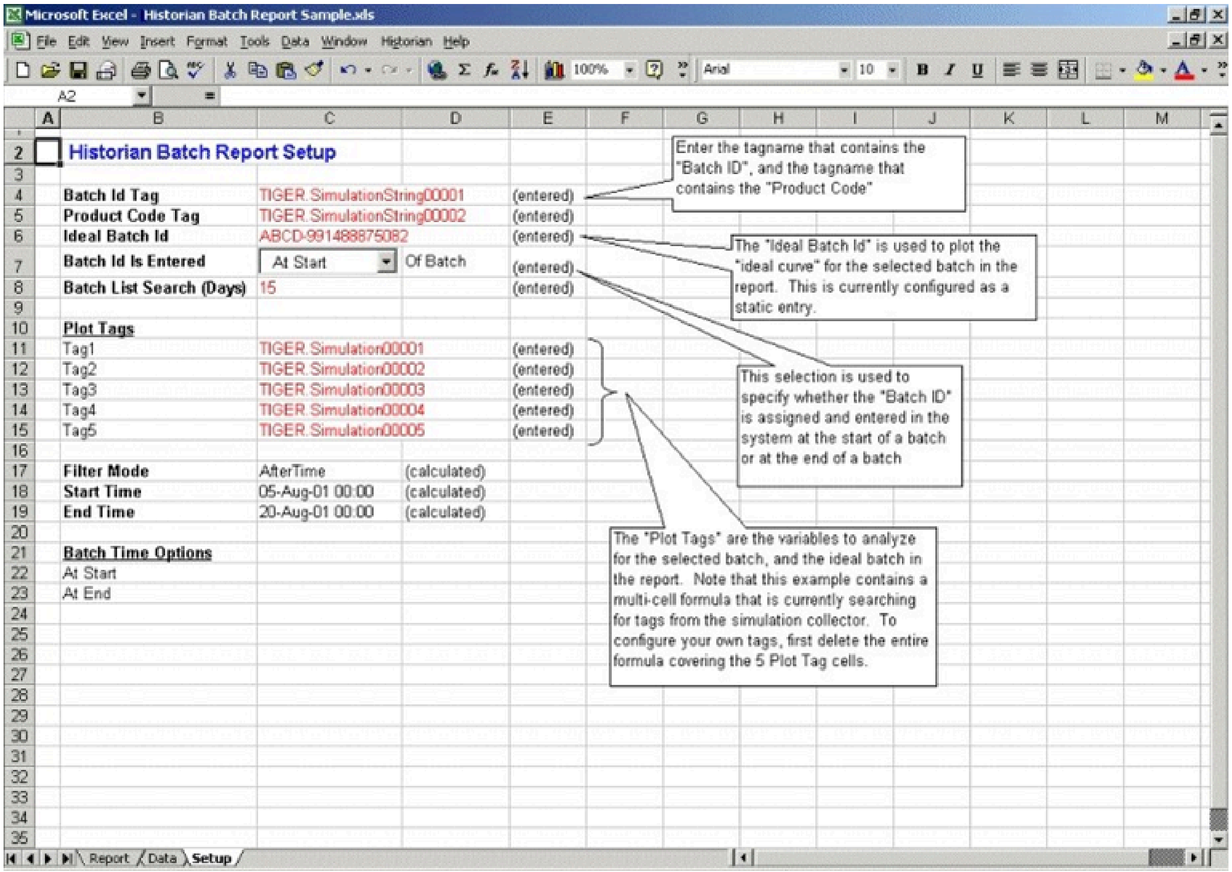


Batch Sample Report

This is an example of a report that might be used with a batch type of industrial process. The table at the top of the report shows the batch identification, the start and end times, product name, and computed statistics for several process variables. The charts show how selected process parameters varied during the batch cycle.



This is the configuration worksheet used to generate the report shown in the previous image. Modify this worksheet to adapt it to your requirements.



Troubleshooting the Excel Add-In Sample Reports

If you follow the recommended installation procedures, you should not have any difficulty in running the sample reports. If you do encounter any problems, they are likely to relate to the locations of files and the links to those files.

When opening a sample Excel report, you may receive a message prompting you to update all linked information in the workbook or keep the existing information. We recommend that you select No (that is, keep the existing information). The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated.

For problems in the worksheets themselves, refer to Excel online Help for assistance.

Running a Report Using Visual Basic

The following Visual Basic example shows you how to create a hidden instance of Microsoft Excel, open a preconfigured Historian report in that instance, and then print the report to the default printer. To use the example, you must modify the path of the `.XLA` and `.XLS` files. The paths that you need to edit are in bold font in the following example.

To use this example, you must have the privileges to run the collector as a Windows service and a default printer must be installed. If Historian security is enabled, you must be a member of the iH Readers group. Tag-level security can override this privilege.

You can trigger this example to run on an event basis or on a polled basis. Most likely, you would run this example on an event basis. However, you can run it on a polled basis using Windows Task Scheduler.

```
Sub CreateExcelObjects()

Dim xlApp As Excel.Application Dim wkbNewBook As Excel.Workbook Dim wksSheet As Excel.Worksheet Dim strBookName As
 String

' Create new hidden instance of Excel. Set xlApp = New Excel.Application

' Open the preconfigured Historian Excel Add-in report.

Workbooks.Open "C:\Program Files\Microsoft Office\Office11\Library\iHistorian.xla"

Set wkbNewBook = Workbooks.Open("c:\testih.xls", 0, False)

'xlApp.Visible = True

With wkbNewBook

For Each wksSheet In .Worksheets

Select Case wksSheet.Name Case "tag1" wksSheet.Select

.RefreshAll

.PrintOut End Select Next wksSheet

.Close False

End With

Set wkbNewBook = Nothing xlApp.Quit

Set xlApp = Nothing

End Sub
```

Array Formulas for the Historian Excel Add-In

In Excel, an array formula is a data request that inputs a set of parameters and returns results. The Historian Excel Add-In uses the following array formulas:

```
ihSearchTags

(pServer,pTagMask,pDescriptionMask,pCollector,pArraySize,pSort,pRowCol,Parameters())


ihQueryData

(pServer-,pTagName,pStartTime,pEndTime,pSamplingMode,pCalculationMode,pSamplingInterval,pNumberOfSamples,pDirection,pFi
lterTag,pFilterMode,pFilterComparisonMo ())


ihQueryData3
```

```
(pServer,pTagName,pStartTime,pEndTime,pSamplingMode,pCalculationMode,pSamplingInterval,pNumberOfSamples,pDirection,pFil
terTag,pFilterMode,pFilterComparisonMo ())

ihQueryMessages
(pServer,pTopic,pStartTime,pEndTime,pSearchText,pArraySize,pSort,pRowCol,Parameters())

ihListArchives
(pServer,pArchiveNameMask,pArraySize,pSort,pRowCol,Parameters())

ihListCollectors
(pServer,pCollectorNameMask,pArraySize,pSort,pRowCol,Parameters())
```

When inserting an array formula, you cannot overwrite part of the range of another array formula in your worksheet. The range includes cells without data displayed. An error message appears if you try to do so. Reselect a different output range to insert the formula.

Array Formula Parameters

The following table describes the parameters for the array formulas for the add-in.

Parameter	Description
`pArchiveNameMask`	A search mask you can use to browse the archivers. Use standard Windows wildcard characters.
`pArraySize`	The number of cells that the array spans.
`pCalculationMode`	The type of the calculation mode.
`pCollector`	The collector or collector mask that you want to query.
`pCollectorNameMask`	A search mask for browsing collectors. Use standard Windows wildcard characters.

Parameter	Description
`pDescriptionMask`	A search mask for browsing tag descriptions. Use standard Windows wildcard characters.
`pDirection`	The direction (forward/backward from the start time) of data sampling from the archive.
`pEndTime`	The end time used to refine your query.
`pFilterComparisonMode`	The type of comparison to be made on the filter comparison value: • `Equal`: Filter condition is True when the `FilterTag` is equal to the comparison value. • `EqualFirst`: Filter condition is True when the `FilterTag` is equal to the first comparison value. • `EqualLast`: Filter condition is True when the `FilterTag` is equal to the last comparison value. • `NotEqual`: Filter condition is True when the `FilterTag` is NOT equal to the comparison value. • `LessThan`: Filter condition is True when the `FilterTag` is less than the comparison value. • `GreaterThan`: Filter condition is True when the `FilterTag` is greater than the comparison value. • `LessThanEqual`: Filter condition is True when the `FilterTag` is less than or equal to the comparison value. • `GreaterThanEqual`: Filter condition is True when the `FilterTag` is greater than or equal to the comparison value. • `AllBitsSet`: Filter condition is True when the binary value of the `FilterTag` is equal to all the bits in the condition. It is represented as `^` to be used in Filter Expression. • `AnyBitSet`: Filter condition is True when the binary value of the `FilterTag` is equal to any of the bits in the condition. It is represented as `~` to be used in Filter Expression. • `AnyBitNotSet`: Filter condition is True when the binary value of the `FilterTag` is not equal to any one of the bits in the condition. It is represented as `!~` to be used in Filter Expression. • `AllBitsNotSet`: Filter condition is True when the binary value of theFilterTag is not equal to all the bits in the condition. It is represented as `!^` to be used in Filter Expression.

Parameter	Description
`pFilterComparisonValue`	The value to compare the filter tag with when applying the appropriate filter to the DataRecordset query (to determine the appropriate filter times).
`pFilterExpression`	An expression that includes multiple filter conditions. The type of conditions used are: • `AND` condition • `OR` condition • Combination of both `AND` and `OR` You can use a filter expression instead of `FilterTag`, `FilterComparisonMode` and `FilterValue` parameters. While using `FilterExpression`, the expression is passed within single quotes, and for complex expressions, enclose the conditions in parentheses. There is no maximum length for a filter expression.
`pFilterMode`	The type of the time filter: • `ExactTime`: Retrieves data for the exact times that the filter condition is True (only True). • `BeforeTime`: Retrieves data from the time of the last False filter condition up until the time of the True condition (False until True). • `AfterTime`: Retrieves data from the time of the True filter condition up until the time of the next False condition (True until False). • `BeforeAndAfterTime`: Retrieves data from the time of the last False filter condition up until the time of next False condition (While True). • The `FilterMode`: Defines how time periods before and after transitions in the filter condition should be handled. For example, `AfterTime` indicates that the filter condition should be True starting at the timestamp of the archive value that triggered the True condition and leading up to the timestamp of the archive value that triggered the False condition.
`pFilterTag`	The single tagname used when applying the filter criteria.

Parameter	Description
`pNumberOfSamples`	Number of samples from the archive to retrieve. Samples will be evenly spaced within the time range defined by start time and end time for most sampling modes. For the `RawByNumber` sampling mode, the `NumberOfSamples` column determines the maximum number of values to retrieve. For the `RawByTime` sampling mode, the `NumberOfSamples` is ignored.
`pRowCol`	The sorting criteria used: 0 for columns and 1 for rows.
`pSamplingInterval`	For non-raw sampled data, this column represents a positive integer for the time interval (in milliseconds) between returned samples.
`pSamplingMode`	The type of the sampling mode used by the query.
`pSearchText`	The text or mask that you want to search for in the message.
`pServer`	Name of the server from which you are retrieving data. If you are running Excel on the same server from which you are retrieving data, you need not enter a string, as the default server is used.
`pSort`	The sorting criteria used for the rows or columns: 0 for descending and 1 for ascending.
`pStartTime`	The start time used to refine your query.
`pTagMask`	A search mask for browsing tagnames. Use standard Windows wildcard characters.
`pTagName`	The tagname or tagname mask that you want to query.
`pTopic`	The message topic: • Connections • Configuration

Parameter	Description
	• General • Services • Performance • Security
`Parameters()`	Output display of the array formula. This field can include be one or more parameters.

Relative Time Entries

When entering the Start and End times for Excel Add-in queries and exports, you can already enter them as exact literal dates and times such as *"`1/28/14 09:00:00`"* in the query windows like **Query Calculated Data**, or you can use a cell reference to an exact time, or use an Excel function such as `=Now()` or `=Today()`. Apart from the mentioned ways, you can use relative time entries using a base value and an offset value as described in the following tables.

For example, you can use `Yesterday+8H` for 8am yesterday or `Now-15m` for 15 minutes before the current time. The typical use of a relative time entry, is to type the time values using a base and an offset into the start and end time of the **Query** window or the **Export** window, instead of having to put `=Now()` or `=Today()` in a cell and making a cell reference to that, or use the base `Monday` to produce weekly reports.

Base Values

Base Value	Description
`Now`	The current date and time.
`Today`	The current date at midnight.
`Yesterday`	The previous day at midnight.
`Sunday`	Today or the most recent Sunday at midnight.
`Monday`	Today or the most recent Monday at midnight.
`Tuesday`	Today or the most recent Tuesday at midnight.
`Wednesday`	Today or the most recent Wednesday at midnight.

Base Value	Description
`Thursday`	Today or the most recent Thursday at midnight.
`Friday`	Today or the most recent Friday at midnight.
`Saturday`	Today or the most recent Saturday at midnight.

Offset Values

Offset Value	Description
`d`	One 24 hour day
`h`	One hour
`m`	One minute
`s`	One second

Filter Parameters for Data Queries

Parameters	Description
`Filter Tag`	The single tag name used when applying the filter criteria. **Note:** You can enter your filter conditions using Filter tag, Filter Comparison Mode, and Filter Comparison Value or you can put that all that information in a single Filter Expression.
`Filter Expression`	An expression that includes one or more filter conditions. The types of conditions used are: • AND Condition • OR Condition • Combination of both AND and OR FilterExpression can be used instead of FilterTag, FilterComparisonMode and FilterValue parameters. There is no maximum length for a filter expression.
`Filter Mode`	The type of time filter:

Parameters	Description
	• **ExactTime** — Retrieves data for the exact times that the filter condition is True (only True). • **BeforeTime** — Retrieves data from the time of the last False filter condition up until the time of the True condition (False until True). • **AfterTime** — Retrieves data from the time of the True filter condition up until the time of next False condition (True until False). • **BeforeAndAfterTime** — Retrieves data from the time of the last False filter condition up until the time of next False condition (While True). The Filter Mode defines how time periods before and after transitions in the filter condition should be handled. For example, AfterTime indicates that the filter condition should be True starting at the timestamp of the archive value that triggered the True condition and leading up to the timestamp of the archive value that triggered the False condition.
`Filter Comparison Mode`	The type of comparison to be made on the filter comparison value: • **Equal** — Filter condition is True when the Filter Tag is equal to the comparison value. • **EqualFirst** — Filter condition is True when the Filter Tag is equal to the first comparison value. • **EqualLast** — Filter condition is True when the Filter Tag is equal to the last comparison value. • **NotEqual** — Filter condition is True when the Filter Tag is NOT equal to the comparison value. • **LessThan** — Filter condition is True when the Filter Tag is less than the comparison value. • **GreaterThan** — Filter condition is True when the Filter Tag is greater than the comparison value. • **LessThanEqual** — Filter condition is True when the Filter Tag is less than or equal to the comparison value. • **GreaterThanEqual** — Filter condition is True when the Filter Tag is greater than or equal to the comparison value. • **AllBitsSet** — Filter condition is True when the binary value of the Filter Tag is equal to all the bits in the condition. It is represented as `^` to be used in Filter Expression.

Parameters	Description
	• **AnyBitSet** — Filter condition is True when the binary value of the Filter Tag is equal to any of the bits in the condition. It is represented as `~` to be used in Filter Expression. • **AnyBitNotSet** — Filter condition is True when the binary value of the Filter Tag is not equal to any one of the bits in the condition. It is represented as `!~` to be used in Filter Expression. • **AllBitsNotSet** — Filter condition is True when the binary value of the Filter Tag is not equal to all the bits in the condition. It is represented as `!^` to be used in Filter Expression. • **Alarm Condition** — Specifies an alarm condition to filter data by. For example, Level. • **Alarm SubCondition** — Specifies an alarm sub-condition to filter data by. For example, HIHI. The Filter Comparison Mode defines how archive values for the Filter Tag should be compared to the Filter Value to establish the state of the filter condition. If a Filter Tag and Filter Comparison Value are supplied, time periods are filtered from the results where the filter condition is False. **Note:** Filter Comparison Mode is only used if Filter Tag is filled in.
`FilterComparisonValue`	The value to compare the filter tag with when applying the appropriate filter to the data record set query (to determine the appropriate filter times). **Note:** Filter Comparison Value is only used if Filter Tag is filled in.

Batch IDs

If you had a `BatchID` going into a Historian tag, that `BatchID` will either have a timestamp at the beginning of the batch or at the end of the batch. Different batch systems report the `BatchID` as the batch is started, and other systems do not report the `BatchID` until the batch is finished.

If your `BatchID` is reported at the beginning of a batch, you would need to use the **AfterTime** option because you would want to include all data for a particular `BatchID` after the time the BatchID was reported up until the next `BatchID` was reported. If your `BatchID` was being reported at the end of the batch, you would want to use the **BeforeTime** option because you would want to include all data for a particular `Batch ID` before the time the `Batch ID` was reported back to the previous `BatchID` being reported.

Sampling Types

Interpolated Sampling

Calculates values between two data points using a linear interpolation algorithm.

Calculated Sampling

Computes values using an algorithm selected in the Calculation field.

Lab Sampling

Computes intermediate values between two data points by using the last actual value. This type of sampling displays as a stair step type of curve.

Trend Sampling

Returns the raw minimum and raw maximum value for each specified interval. Use the Trend sampling mode to maximize performance when retrieving data points for plotting. For the Trend sampling mode, if the sampling period does not evenly divide by the interval length, **Historian** ignores any leftover values at the end, rather than putting them into a smaller interval.

InterpolatedtoRaw Sampling

Provides raw data in place of interpolated data when the number of samples fall lesser than the available samples.

TrendtoRaw Sampling

The TrendtoRaw sampling mode almost always produces the same results as the Trend sampling mode. The exception is that, when more samples are requested than there are raw data points, the TrendtoRaw sampling mode returns all of the available raw data points with no further processing. TrendtoRaw is therefore used rather than Trend when the number of actual data samples are fewer than the requested number of samples.

LabtoRaw Sampling

Provides raw data for the selected calculated data over the plot, when the number of samples fall lesser than the available samples.

RawByFilterToggle Sampling

Returns filtered time ranges with values 0 and 1. If the value is 1, then the filter condition is true and 0 means false. This sampling mode is used with the time range and filter tag conditions. The result starts with a starting time stamp and ends with an ending timestamp.

Trend2 Sampling

Returns the raw minimum and raw maximum value for each specified interval. Use the Trend2 sampling mode to maximize performance when retrieving data points for plotting. Also, if the sampling period does not evenly divide by the interval length, Historian creates as many intervals of the interval length as will fit into the sampling period, and then creates a remainder interval from whatever time is left. Trend2 sampling mode is more suitable than Trend sampling mode for analysis of minutes and maxes and for plotting programs that can handle unevenly spaced data.

TrendtoRaw2 Sampling

The TrendtoRaw2 sampling mode almost always produces the same results as the Trend2 sampling mode. The exception is that, when more samples are requested than there are raw data points, the TrendtoRaw2 sampling mode returns all of the available raw data points with no further processing. TrendtoRaw2 is therefore used rather than Trend2 when the number of actual data samples are fewer than the requested number of samples.

Calculation Algorithm Types

Average

A time weighted arithmetic mean.

Minimum

The lowest value in the group.

Maximum

The highest value in the group.

Standard Deviation

The square root of the arithmetic mean of deviations from the time- weighted arithmetic mean of all values in the group.

Total

The time-weighted total of all values in the group. Note that Engineering Units are assumed to be in Units/Day. If your Engineering Units were not measured in Units/Day, you must scale your total to the actual time units of the measurement. For example, if the measurement were in Units/Minute (such as GPM), you would multiply the total number by 1440 (minutes in a day) to scale the value into the correct time units.

Count

The total number of values in the group.

Raw Average

The unweighted arithmetic mean of all values in the group.

Raw Standard Deviation

The square root of the arithmetic mean of deviations from the unweighted arithmetic mean of all values in the group.

Raw Total

The unweighted total of all values in the group.

Time of Minimum Value

The time at which the minimum value occurred. l Time of Maximum Value - the time at which the maximum value occurred.

Time Good

The amount of time (in milliseconds) during the interval when the data quality is good.

State Count

Displays the number of times a tag has transitioned to another state from a previous state. A state transition is counted when the previous good sample is not equal to the state value and the next good sample is equal to state value.

State Time

Displays the duration that a tag was in a given state within an interval.

First Raw Value

Returns the first good raw sample value in the given time interval.

First Raw Time

Returns the time stamp of the first good raw sample in the given time interval.

Last Raw Value

Returns the last good raw sample value in the given time interval.

Last Raw Time

Returns the time stamp of the last good raw sample in the given time interval.

TagStats

Returns the values of multiple calculation modes in a single query.

Tag Criteria List

The following table outlines the tag criteria available:

Criteria	Description
Tagname	Tagname or tag mask property of the tag.
Description	User description of the tag.
Data Type	The data type of the tag.
Collector Name	Name of the collector responsible for collecting data for the specified tag.
Collector Type	The type of collector responsible for collecting data for the tag. **Note:** Do not use wildcards in this field.
Collection Type	Type of collection used to acquire data for the tag.
Data Store Name	Indicates the name of the data store to which the tag belongs to.
EGU Description	Indicates the engineering units assigned to the tag. **Note:** Do not use wildcards in this field.
Comment	Comments that is applied to the tag. **Note:** Do not use wildcards in this field.
Source Address	The address for the selected tag in the data store.

Criteria	Description
	Note: Do not use wildcards in this field.
Collection Interval	The time interval between the readings of data. The value entered is in milliseconds.
Collector Compression	Whether or not collector compression is enabled as a default setting.
Archive Compression	Indicates the current effect of archive data compression.
Last Modified User	The name of the person who last modified the tag configuration parameters.

Troubleshooting Issues with the Add-In

Troubleshooting General Imports

- Review the `HistorianSDKErrors.log` file. This file is usually located in the `LogFiles` folder in your Historian program folder. Historian records additional information for some errors in this file. Sometimes, by reviewing this file, you can determine the cause of the error.
- If using Historian security, verify that the user has the appropriate security rights. If the rights are incorrect, log in as a user with the correct privileges or change the rights for the current user.
- Verify that there are no empty rows between valid rows in your spreadsheet. These empty rows can cause issues.
- Note if any errors occur. If an error occurs with any line of the import, Historian aborts the whole import.

Troubleshooting Tag Imports

- If you remove or add Historian servers, and then if you attempt to search for tags, the add-in may not recognize the default server, and may display a message, stating that the default server has not been set. To avoid this issue, close and reopen the **Search Tags** window.

- Make sure that you are not trying to import the Calculation Execution Time, Last Modified, or Last Modified User fields for each tag. These fields are read-only. As such, you can export them but cannot import them.
- Verify that your collector does not contain any duplicate tagnames.
- Verify that the number of tags that you want to import does not exceed the maximum licensed tag count. If it does, you will not be able to import the tags.

Troubleshooting Data Imports

- Ensure that the time stamps of any online archives are not prior to the start time of the oldest online archive.
- Ensure that the time stamps are not for a time greater than 15 minutes ahead of the system time on the Historian server.
- Ensure that the tags are valid Historian tags. To do this, import your tags before importing their associated data.

Troubleshooting Data or Tag Exports

You cannot export data or tags to a remote path using the add-in.

You can export a 64-bit tag, include it in a report and perform calculations on it. However, there will be a minor precision loss while retrieving the data due to a Visual Basic limitation.

Importing Tags Fails

Description: If you export all the fields and attempt to import the read-only fields **LastModified** and **LastModifiedUser**, you may receive an error message.

Error Message: Import failed, Error with Import Header.

Workaround: Export the tags without selecting the read-only fields, and then import the tags.

Unable to Run Sample Reports

Description: If you follow the recommended installation procedures, you should not have any difficulty in running the sample reports. If you do encounter any problems, they are likely to relate to the locations of files and the links to those files.

Workaround: When opening a sample Excel report, you may receive a message prompting you to update all linked information in the workbook or keep the existing information. We recommend that you select No (that is, keep the existing information). The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated.

For problems in the worksheets themselves, refer to Excel online Help for assistance.

Error Occurs While Inserting an Array Formula

When inserting an array formula, you cannot overwrite part of the range of another array formula in your worksheet. The range includes cells without data displayed. An error message appears if you try to do so. Reselect a different output range to insert the formula.

Chapter 8. Using the OLE DB Provider

Overview of the OLE DB Provider

OLE DB is a collection of standard COM-based interfaces defined by Microsoft that abstract standard SQL commands into native API access for any data source. OLE DB adds tremendous value to Historian by providing simple access to data from within the SQL environment, without the need for complex scripting.

The Historian OLE DB provider is a data access mechanism that allows you to query Historian data using SQL statements or other client reporting tools.

Supported Applications: Using the OLE DB provider, you can create reports and integrate Historian with the following applications:

- Microsoft Power BI
- Seagate Crystal Reports v8.0, and above (v11.0 or above required for use with Historian Alarms and Events)
- VisiconX with iFIX v4.0 and later
- Microsoft Excel 2003 and later
- Visual Basic v6.0, Service Pack 5
- Visual Basic for Applications (VBA) v6.0
- Microsoft SQL Server v7, Service Pack 3
- Microsoft SQL Server 2008, or SQL Server Express 2008
- Oracle 8.x and above
- Dream Report™

> **Note:**
> Other OLE DB clients are likely to work with the OLE DB provider, but have not been tested.

Limitations: The OLE DB provider has read-only access. You cannot insert, update, or delete data in archives using the OLE DB provider.

Setting Up

Install Client Tools

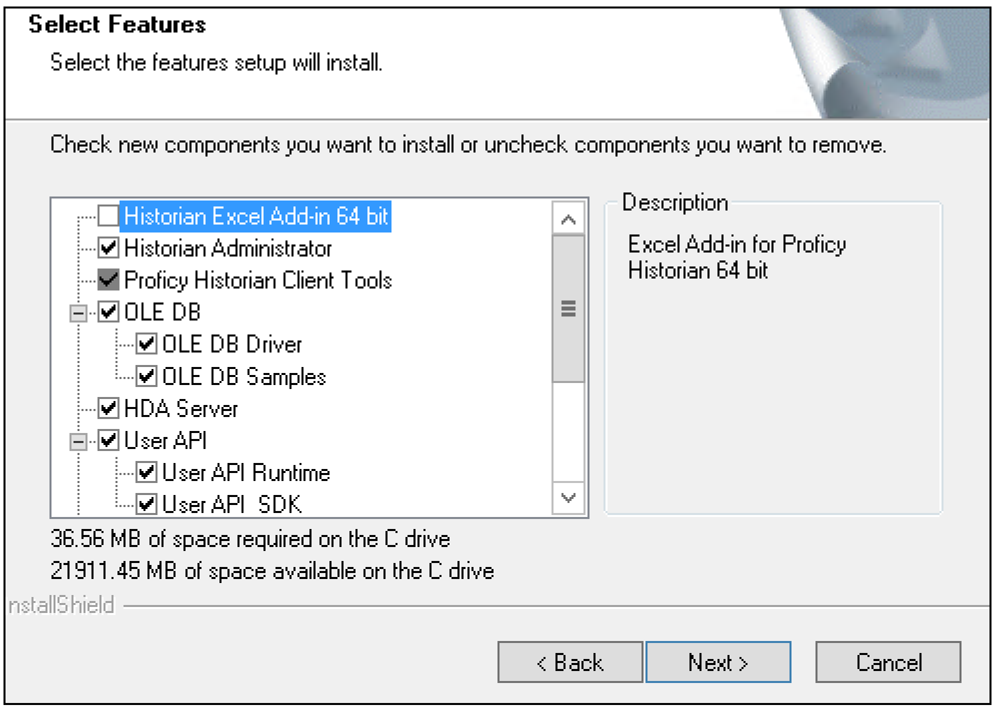
When you install Client Tools, the following components are installed by default:

- Client Tools
- Historian Administrator
- OLE DB provider (driver and samples)
- The OPC Classic HDA server
- User API and SDK
- Historian Client Access API
- Collector Toolkit

This topic describes how to install Client Tools using the installer. You can also install it at a command prompt *(on page 237)*.

1. Reach out to the GE Digital Support team for the installer.
2. From the installer, select **Install Client Tools**.

   The **Select Features** page appears, displaying a list of components that you can install with Client Tools.

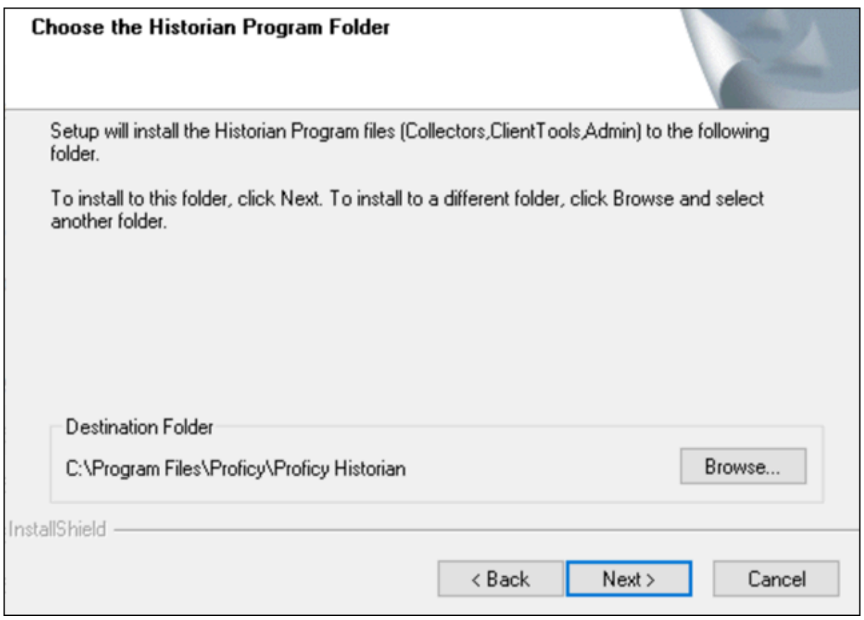
   

   By default, the check boxes for components such as **Historian Administrator**, **HDA Server**, **OLE DB**, and **User API and SDK** are selected. If you do not want to install them at this time, clear the check boxes. You cannot, however, clear the **Proficy Historian Client Tools** check box.

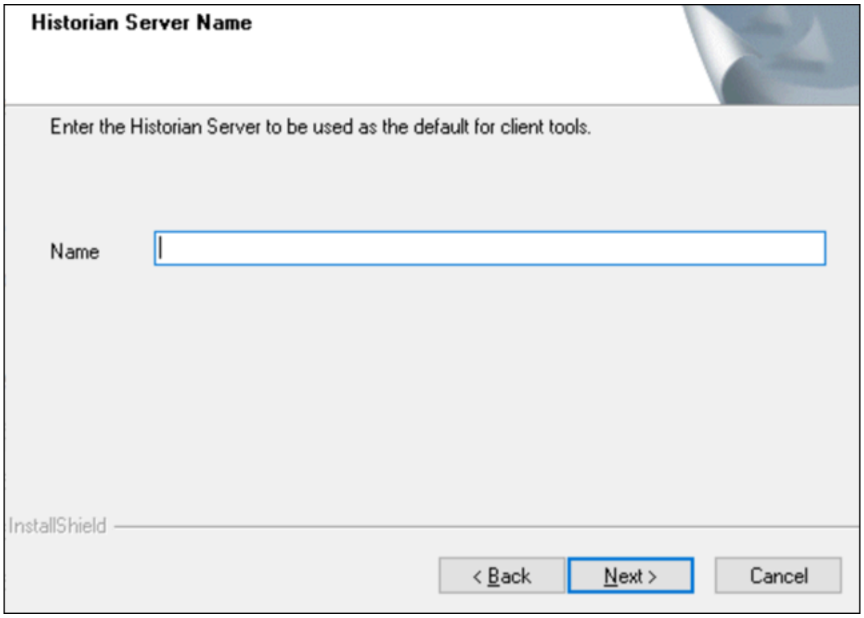
> **Important:**
> If you are reinstalling, you must select all of the previously installed components. If you do not do so, the component will be uninstalled.

By default, the **Historian Excel Add-in 64-bit** check box is cleared. If you want to install Excel Add-In along with Client Tools installation, select the check box.

> **Note:**
> If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message while installing Excel Add-In, stating that some of the DLL files are not registered. You can ignore these messages.

3. Select **Next**.
   The **Choose the Historian Program Folder** page appears.
   

4. As needed, change the destination folder of Client Tools, or leave the default folder, and then select **Next**.
   The **Historian Server Name** page appears.



5. Enter the Azure Load Balancer IP.

> **Tip:**
> To find the Azure Load Balancer IP:
> a. Go to the Azure portal.
> b. Go to the **Resource Group** that was specified during deployment.
> c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
> d. Select or copy the IP Address.

6. When you are asked to reboot your system, select **Yes**.

Client Tools, along with the selected components, are installed in the following folder: <*installation drive*>:\Program Files\Proficy\Proficy Historian\x86\<*tool name*>. If you have selected HDA Server, Microsoft .NET Framework 4.5 and the OPC Core Components 3.00 redistributable are installed as well.

Install Client Tools at a Command Prompt

1. If you want to install Excel Add-In for Historian, install one of the following 32-bit or 64-bit Microsoft® Excel® applications:
    - Microsoft® Excel® 2019
    - Microsoft® Excel® 2016

  - Microsoft® Excel® 2013
  - Microsoft® Excel® 2010
2. Install Client Tools using the installer *(on page 234)* on a machine. When you do so, a template file named `setup.iss` is created at `C:\Windows`. This file stores the installation options that you have provided. You can then use this template to install Client Tools at a command prompt on other machines.

When you install Client Tools, the following components are installed by default:

- Client Tools
- Historian Administrator
- OLE DB driver and samples
- The OPC Classic HDA server
- User API and SDK
- Historian Client Access API
- Collector Toolkit

1. Copy the `setup.iss` file to the machine on which you want to install Client Tools at a command prompt.
2. In the folder in which you have copied the file, run the following command: `setup.exe /s /sms` The installer runs through the installation steps.

   Note:
   If using certain versions of Windows (such as Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Client Tools are installed.

If you have installed Excel Add-in, activate it *(on page 186)*.

Connect to a Historian Server

1. Install Client Tools *(on page 234)*, which will automatically install the OLE DB provider.
2. Initialize the COM library on the machine on which you have installed the OLE DB provider.

This topic provides basic steps to connect the OLE DB provider to a Historian server so that you can import the data. For instructions specific to a client, refer to:

- Import Historian Data into Power BI Desktop *(on page 239)*
- Import Historian Data into Crystal Reports *(on page 246)*
- Import Historian data into Microsoft Excel *(on page 251)*

Run the following command:

```
Provider=iHOLEDB.iHistorian.1;PersistSecurity Info=False;

USER ID=[<Historian server username>];

Password=[<Historian server password>];

Data Source=[<Azure Load Balancer IP>]
```

Working with Clients

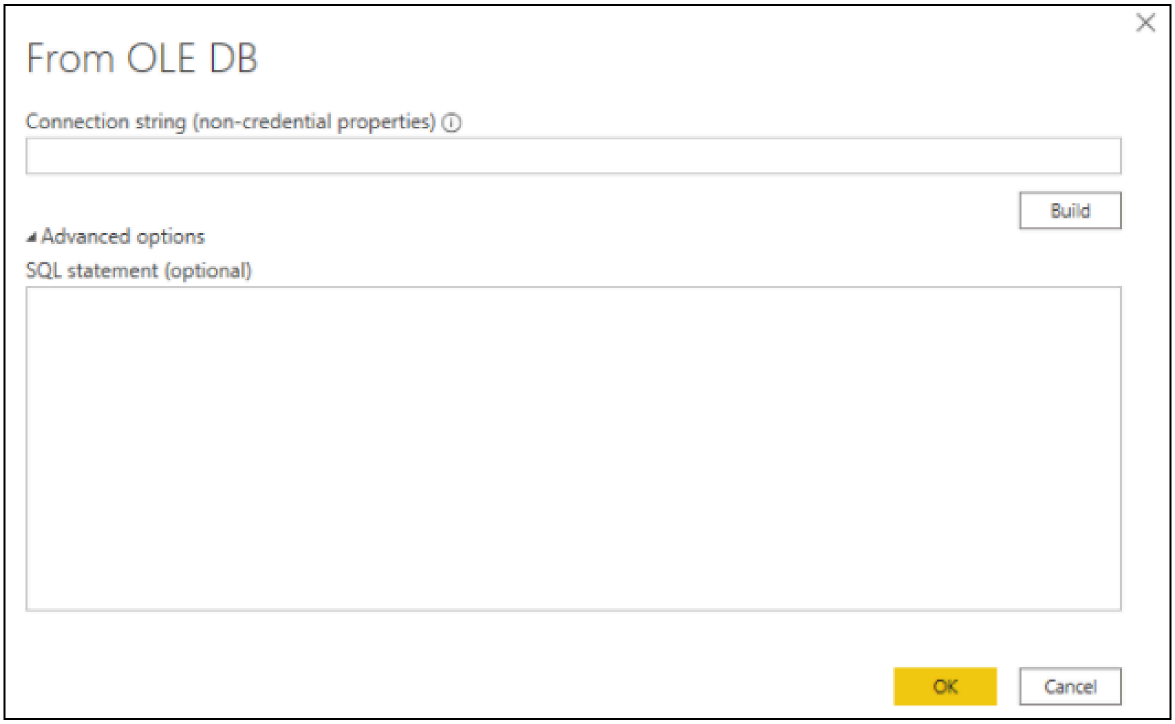
Power BI Desktop

Import Historian Data into Power BI Desktop

Microsoft Power BI Desktop is an application that transforms and visualizes data. Using this application, you can connect to multiple data sources and combine the data into a data model.

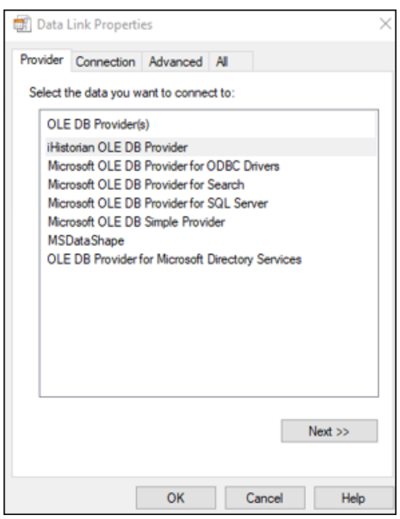
This topic describes how to import Historian data into Power BI Desktop.

1. Access Power BI Desktop.
2. Select **Get Data > Other > OLE DB**, and then select **Connect**.
   The **From OLE DB** window appears.



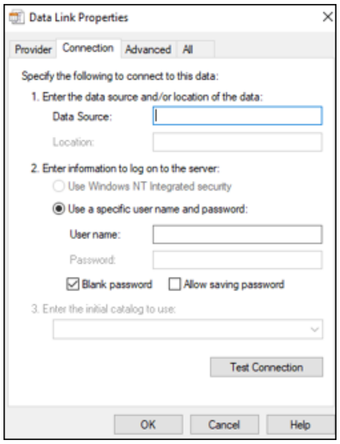
3. Select **Build**.

   The **Data Link Properties** window appears, displaying a list of the Historian OLE DB providers in the **Provider** section.



4. Select **Next**.

   The **Connection** section appears.



5. Leave the default values as is, and select **Test Connection.**

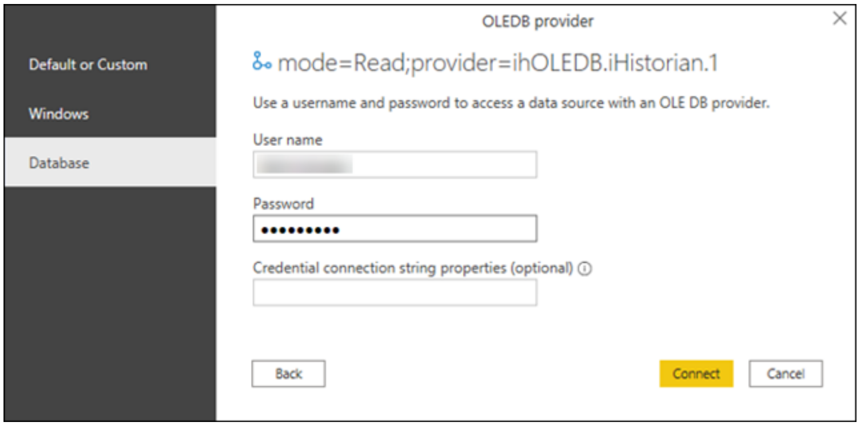
   After the connection succeeds, the connection string is populated in the **From OLE DB** window.

> **Important:**
> Do not use the connection string that is populated. If you do so, an error occurs.

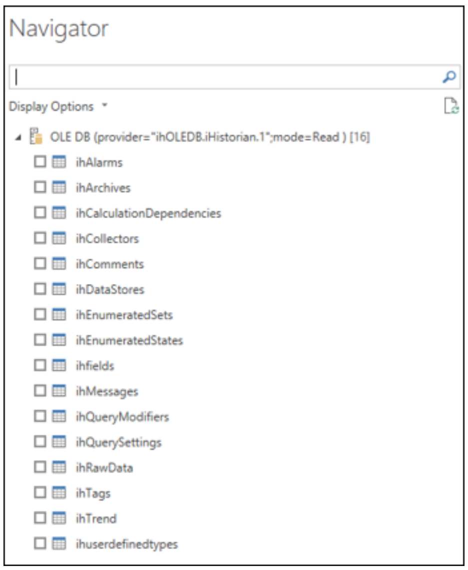
6. Change the connection string to `Provider=iHOLEDB.iHistorian.1;PersistSecurity Info=False; USER ID=[<Historian server username>]; Password=[<Historian server password>]; Data Source=[<Azure Load Balancer IP>]`

7. Select **OK**.

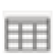
   The **OLE DB Provider** window appears.
8. In the **Database** section, enter `ihCloudHistAdmin` as the username, and enter the password that you provided at the time of deployment, and then select **Connect**.

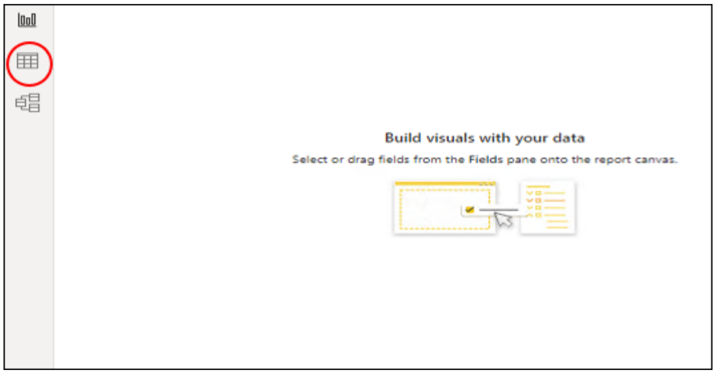


A list of Historian tables appears in the **Navigator** section.



If an error message appears after entering the credentials, restart your machine.

9. Select the table whose data you want to import, and then select **Load**.
10. Select .



Data from the selected Historian table appears.

You can now create a Power BI report and then publish it.

Working with VisiconX

Using the OLE DB provider with VisiconX, you can:

- Use tables or SQL queries.
- Insert multiple controls into a picture to the same or different servers.
- Provide a username and password or be prompted when opening a picture.

1. Access the Historian OLE DB provider from VisiconX. For instructions, refer to Using VisiconX.
2. To make all the VisiconX controls use synchronous (SYNC) executes:

    a. Access the `FixUserPreferences.ini` file in the `Dynamics/Local` folder.

    b. Add the following lines to the end of the file:

    ```
    [VisiconX]

    RUNASYNC=FALSE
    ```

    c. Save the file, and restart the collector.

Access the iFIX Sample Picture

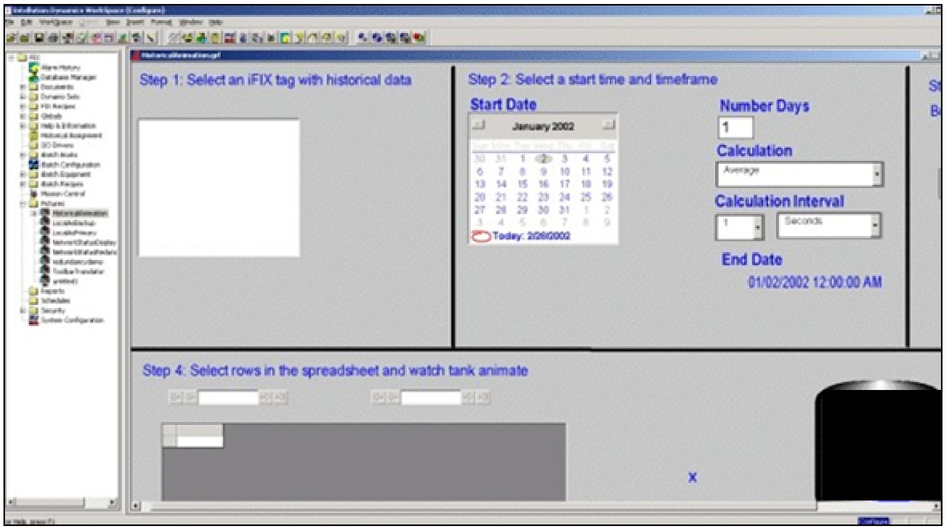
To use iFIX with VisiconX, edit the `FixUserPreferences.ini` configuration file.

The `HistoricalAnimation.grf` file contains an iFIX sample picture with the VisiconX controls. It is located in the `Historian\Samples\iFIX` folder.

1. Copy the `HistoricalAnimation.grf` file to your `Dynamics/Pic` folder.
2. Start iFIX.

3. Open **iFIX WorkSpace**.
4. Double-click the **Pictures** folder.
5. Double-click the **HistoricalAnimation** picture.

   The picture appears in the workspace.



You can now perform the following tasks:

- Modify the picture.
- Switch between the configure and run modes.
- Follow the steps on the picture.
- View the properties of the VisiconX controls.
- Change the properties.

> **Note:**
> You can have multiple VisiconX controls that each link to different Historian servers.

Create a Background Schedule to Run Crystal Reports

In iFIX, you can create a background schedule that runs Crystal reports. This topic contains a sample Visual Basic code to create a background schedule:

```
Private ReportFileName

Private CrystalReport


Private Sub KKTimer_OnTimeOut(ByVal lTimerId As Long)


Set CrystalApplication = CreateObject("Crystal.CRPE.Application")
```

```
Set CrystalReport = CrystalApplication.OpenReport("C:\Program Files (x86)\GE\iFIX\APP\RTtemplate.rpt")
 CrystalReport.Printout False


Set CrystalReport = Nothing
Set CrystalApplication = Nothing
End Sub
```

Working with Oracle

You can import Historian data into Oracle by using an ADO program. A sample program is provided in the `Historian/Samples/Oracle` folder.

Use SQL WorkSheet to test that Oracle imported the data and created the tables properly.

Crystal Reports

Crystal Reports allows you to create reports easily through its experts and wizards. When working with Crystal Reports,remember that:

- Crystal does not support the `SET` command. You must use a `WHERE` clause in a `SELECT` statement to specify query parameters.
- A single report can only retrieve data from one server, but you can create subreports from different servers within a report.
- The Crystal Reports application does not display milliseconds in timestamps.
- IIf you want to create a report on numeric data in the Value or Quality column in the `ihRawData` table, you may want to convert all Variant data types to Float data types so that Crystal displays them correctly in the report. Refer to Format Decimal Point Precision *(on page 248)* for instructions.
- Analysis of the ihTrend and ihAlarm tables in Crystal Reports is not supported.

Table 41. Crystal Reports Samples

File Name	Description
`SimpleCrystal80Report.rpt`	Contains values cast from **Variant** to **Float**.
`MultipleServers-Subreport.rpt`	Contains data from two servers by using a subreport.
`iFIX1_CHART_OLEDB.rpt`	Contains data from iFIX Sample System converted from the iFIX Historical ODBC driver to OLE DB.

Table 41. Crystal Reports Samples (continued)

File Name	Description
`iFIX1_CROSSTAB_OLEDB.rpt`	Contains data from iFIX Sample System converted from the iFIX Historical ODBC driver to OLE DB.
`iFIX1_DAILY_OLEDB.rpt`	Contains data from iFIX Sample System converted from the iFIX Historical ODBC driver to OLE DB.

Connect to the Historian Server

1. Open the report file in Crystal Reports.
2. Select the **Database** menu.
3. If the **Database** menu does not appear automatically:

    a. Wait for approximately 90 seconds for the connection timeout to occur.
    After the 90-second timeout, the **Data Link Properties** window appears. Although it may appear as if Crystal Reports has stopped working or is frozen before the timeout occurs, this functionality is as expected.

    b. In the **Data Source** field, enter the Azure Load Balancer IP.

    > **Tip:**
    > To find the Azure Load Balancer IP:
    > i. Go to the Azure portal.
    > ii. Go to the **Resource Group** that was specified during deployment.
    > iii. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
    > iv. Select or copy the IP Address.

    c. Select **OK**.

    d. Skip the next step.

4. If the **Database** menu appears automatically:

    a. Select **Database > Set Location**.
    The **Set Location** window appears.

    b. Select **Set Location**.
    The **Data Explorer** window appears.

c. Select a source, and then select **Set**.

d. Select **Done**.

The Historian server is connected with Crystal Reports.

Create a Crystal Report

Ensure that Crystal Reports is integrated with the Historian server whose data you want to analyze. For instructions, refer to Connect to the Historian Server *(on page 246)*.

This topic describes how to import Historian table data into Crystal Reports and create a report.

1. In Crystal Reports, select **File > New**.
   The **Crystal Reports Gallery** window appears.
2. Select **Using the Report Expert > Standard Report Expert**, and then select **OK**.
   The **Standard Report Expert** appears.
3. Select **Database**.
   The **Data Explorer** appears.
4. Open the `More Data Sources` folder, and then open the `OLE DB` folder.
5. Select **Make New Connection > Add**.
   The **Data Link Properties** window appears.
6. Select **Historian OLE DB Provider > Next**.
   The **Connection** section appears.
7. Leave these fields empty to use the default server and currently logged-in user. Otherwise, do the following:

   a. In the **Data Source** field, enter the Azure Load Balancer IP.

   > **Tip:**
   > To find the Azure Load Balancer IP:
   > i. Go to the Azure portal.
   > ii. Go to the **Resource Group** that was specified during deployment.
   > iii. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
   > iv. Select or copy the IP Address.

   b. Clear the **Blank Password** check box.

   c. Enter a Windows username and password.

8. Select **OK**.
   The Historian OLE DB provider tables appear in the **Data Explorer**.
9. Select the table that you want to query, select **Add**, and then select **Close** to exit the **Data Explorer** window.
10. In the **Fields** section of the **Standard Report Explorer** window, select a field that you want to report on, and then select **Add** to move the field into the **Fields to display** list.

    > **Note:**
    > If you want to create a report on numeric data in the Value or Quality column in the `ihRawData` table, you may want to convert all Variant data types to Float data types so that Crystal displays them correctly in the report. Refer to Format Decimal Point Precision *(on page 248)* for instructions.

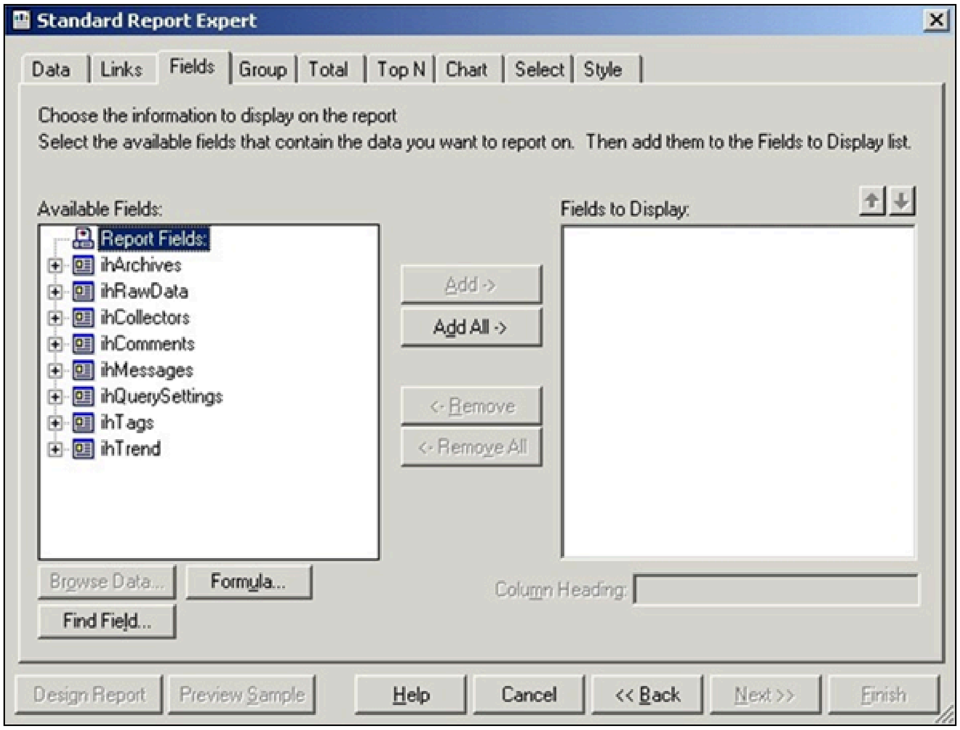
11. Repeat the previous step for each field that you want to add, and then select **Finish**.
    The Crystal Report is generated.

Format Decimal Point Precision

Connect to the OLE DB provider, and add the Historian database tables.

To format decimal point precision in your reports, you must convert Variant data types to Float data types in Crystal Reports. For instance, if retrieving the Value column from the ihRawData table, you must convert the values to Float. You need not perform these steps if you are working with strings.

1. Access **Standard Report Expert**, and then select **Fields**.
   The **Fields** section appears.



2. Select **Formula**.

   The **Formula Name** window appears.

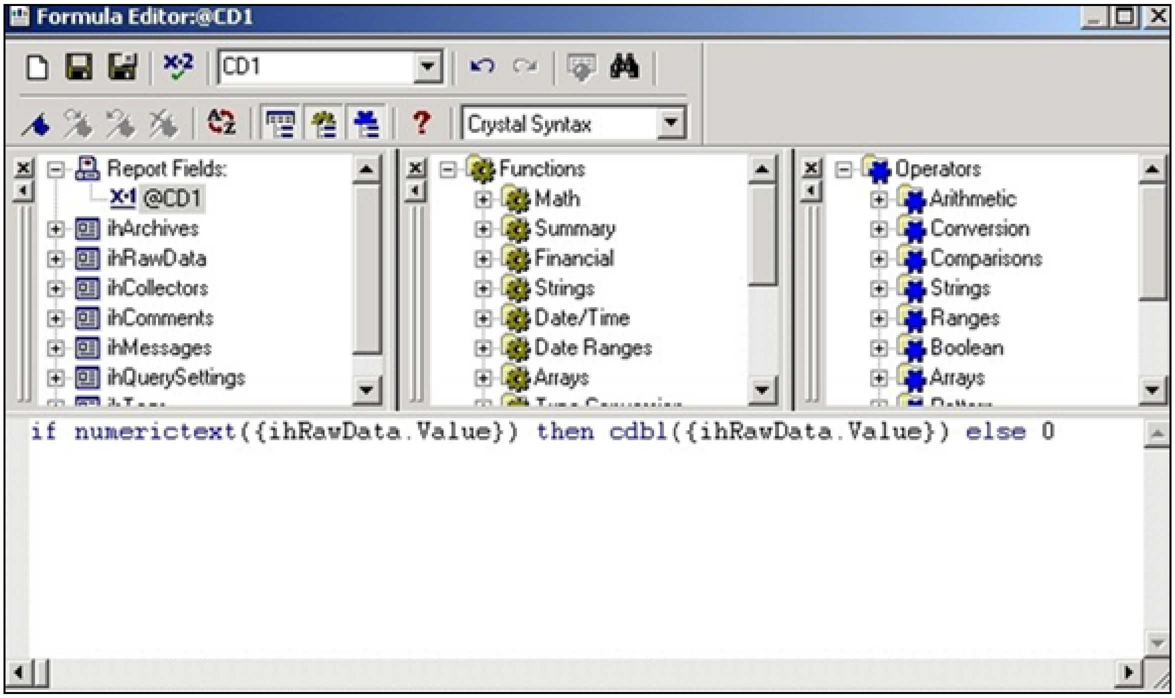
3. Enter a name for the formula.

   The **Formula Editor** section appears.

> **Tip:**
> You can also access the **Formula Editor** section by selecting **Insert > Field Object**. Right-click the formula fields, and then select **New**.

4. In the **Formula** field, enter the following text :

```
if numerictext({ihRawData.Value}) then cdbl({ihRawData.Value}) else

                        0
```



5. Select **Save**.

   You can now use the formula as a normal numeric column instead of the **Value** column in the report.

Change the Date and Time Format

This topic describes how to format the date/time column of Historian tables in Crystal Reports. When formatting timestamps, note that milliseconds do not appear in Crystal Reports.

1. Select a field in a column that contains timestamps.
2. Right-click the field, and then select **Format Field**.

   The **Format Editor** window appears.
3. Select **Date/Time**, and specify the date format that you want to use.
4. Select **OK**.

   The timestamps are updated to display the new format.

Microsoft Excel

With Excel, you can import a snapshot of Historian data at a single point in time. You can choose Historian as a data source in Excel. You can specify the connection settings manually *(on page 251)* or using a UDL file *(on page 252)*.

After you import the data, you can create and edit SQL queries in Excel.

When to Use Excel Instead of the Historian Excel Add-In

Use the Excel Add-In when you want to get data into Microsoft Office 2003, 2007 or 2010 (32-bit/64-bit). Use the Historian OLE DB provider with Excel, instead of the Excel Add-In, when you want to do any of the following:

- Perform advanced filtering, sorting, and joining of data.
- Obtain detailed information from the ihTrend table.
- Run calculations using the SQL aggregate functions.
- Perform advanced summaries.

Table 42. Microsoft Excel Samples

File Name	Description
`ihOLEDB_ LASTHOUR.XLS`	One-sheet report that uses auto-refresh to display the last hour of data using relative shortcuts.
`ihTags.odc`	Data source file that retrieves the `ihTags` table from the default server.
`iHistorian.udl`	Sample universal data link (`.UDL`) file that connects to the default Historian server with the currently logged-in user.

These sample are found in the following folder: `Historian\Samples\Excel`

Import Historian Data Into Excel Manually

This topic describes how to import Historian data into Excel by providing the connection details manually. You can also import the data by creating a UDL file *(on page 252)* or by using the sample UDL file *(on page 254)*.

1. Open an Excel worksheet.
2. Select **Data > Import External Data > Import Data**.
   The **Select Data Source** window appears.
3. Select **My DataSources > +Connect to New Data Source.odc > Open**.
   The **Data Connection Wizard** appears.
4. Select **Other/Advanced** from the list of data sources to which you can connect, and then select **Next**.
   The **Data Link Properties** window appears.
5. Select **Historian OLE DB Provider** from the **OLE DB Provider** list, and then select **Next**.
   The **Connection** section appears in the **Data Link Properties** window.

6. Leave these fields empty to use the default server and the currently logged-in user. Otherwise, do the following:

    a. In the **Data Source** field, enter the Azure Load Balancer IP..

    > **Tip:**
    > To find the Azure Load Balancer IP:
    > i. Go to the Azure portal.
    > ii. Go to the **Resource Group** that was specified during deployment.
    > iii. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
    > iv. Select or copy the IP Address.

    b. Clear the **Blank Password** check box.

    c. Enter a Windows username and password.

    d. Select the **Allow Saving Password** check box if applicable.

7. Select **Test Connection** to confirm that the data source, username, and password provide a successful connection, and then select **OK**.
   The **Select Database and Table** page appears in the wizard.
8. Select the table that you want to query, and then select **Next**.
   The **Save Data Connection File and Finish** page appears in the wizard.
9. Accept the default settings, and select **Finish**.
   The **Import Data** window opens.

   > **Note:**
   > If you want to run a specific SQL command instead of the default table command setting, refer to Edit SQL Queries in Excel *(on page 254)*.

10. Select **OK** to import the column data from the selected table.
    Historian data populates the current spreadsheet.

Import Historian Data Into Excel by Creating a UDL File

This topic describes how to create a UDL file with connection information and then import Historian data into Excel using the UDL file. You can also provide the connection details manually *(on page 251)* or using the sample UDL file *(on page 254)*.

1. Create a UDL file with connection details:

    a. Create a text document.

       We recommend that you use the **My Data Sources** folder in the **My Documents** folder.

    b. Rename the file extension .UDL.

    c. Double-click the `.UDL` file.
       The **Data Link Properties** window appears.

    d. Select **Provider > Historian OLE DB Provider > Next**.
       The **Connection** section appears in the **Data Link Properties** window.

2. Leave these fields empty to use the default server and the currently logged-in user. Otherwise, do the following:

    a. In the **Data Source** field, enter the Azure Load Balancer IP..

       > **Tip:**
       > To find the Azure Load Balancer IP:
       > i. Go to the Azure portal.
       > ii. Go to the **Resource Group** that was specified during deployment.
       > iii. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
       > iv. Select or copy the IP Address.

    b. Clear the **Blank Password** check box.

    c. Enter a Windows username and password.

    d. Select the **Allow Saving Password** check box if applicable.

3. Select **Test Connection** to confirm that the data source, username, and password provide a successful connection, and then select **OK**.
   The **Select Database and Table** page appears in the wizard.
4. Select **Data > Import External Data > Import Data**.
   The **Select Data Source** window appears.
5. Select the `.UDL` file that you have created, and then select **Open**.
   The **Select Table** window appears.
6. Select the table that you want to query, and then select **OK**.
   The **Import Data** window appears.

> **Note:**
> If you want to run a SQL command instead of the default table command setting, refer to Edit SQL Queries in Excel *(on page 254)*.

7. Select **OK** to import the column data from the selected table.
   Historian data is imported into the spreadsheet.

Import Historian Data into Excel Using the Sample UDL File

With the sample universal data link (`.UDL`) file, you can specify the connection information so that Excel can connect to the tables in the OLE DB provider and import data using the default server and the currently logged-in user.

This topic describes how to import Historian data into an Excel spreadsheet using the sample .UDL file. You can also import Historian data by providing the connection details manually *(on page 251)* or by creating a .UDL file *(on page 252)*.

1. Open an Excel spreadsheet.
2. Select **Data > Import External Data > Import Data**.
   The **Select Data Source** window appears.
3. Select the `Historian.udl` file in the `Historian\Samples\Excel` folder, and then select **Open**.
   The **Select Table** window appears.
4. Select the table that you want to query, and then select **OK**.
   The **Import Data** window appears.

> **Note:**
> If you want to run a SQL command instead of the default table command setting, refer to Edit SQL Queries in Excel *(on page 254)*.

5. Select **OK**.
   Historian data appears in the spreadsheet.

Edit SQL Queries in Excel

By default, data import functionality in Excel selects all columns from the specified Historian table using the default query parameters. This command is the equivalent of running the SQL command `SELECT * FROM` *`TABLE_NAME`*, where *TABLE_NAME* is the name of the table that you want to query.

You can change the query by issuing a different SQL query if you are familiar with SQL syntax. Refer to the Microsoft Excel documentation for more information.

If you are unsure if the SQL syntax is correct, you can test your SQL query outside of Excel using the Historian Interactive SQL application. See Historian Interactive SQL Application *(on page 265)* for more details.

Format Date and Time

This topic describes how to format the date/time column for Historian tables in Excel if you need to display a specific date format. For more specific information on formatting spreadsheets, refer to the Microsoft Excel online Help.

1. Right-click the heading of the column that you want to format.
2. Select **Format Cells > Number**.
3. Select **Date**.
4. In the **Type** field, select the date format that you want to use.
   To display milliseconds, instead of selecting the **Date** category, select **Custom**, and then enter `dd-mmm-yyyy hh:mm:ss.000` in the **Type** field.
5. Select **OK**.
   The date and time format is set.

Refresh Data

After you import Historian data into an Excel worksheet, you can refresh it to get the most updated data. This feature is most useful when using relative start times, such as `Now - 2h`. You can also set a refresh interval to refresh data automatically.

1. Open the Excel worksheet into which you have imported the Historian data.
2. Select **External Data > Refresh Data**

   Tip:
   If the **External Data** toolbar is not available, select **View > Toolbars**.

   The data is refreshed.
3. To automatically set refresh intervals, select **Data Range Properties**, and provide the interval at which you want to refresh data automatically.
   Data is refreshed automatically at the interval that you have specified.

Visual Basic and ADO

You can access the OLE DB provider using Microsoft ActiveX Data Objects (ADO). This approach is more generic than using the Historian SDK.

Visual Basic supports asynchronous (ASYNC) connections. You can open multiple ADO connections to the same data source from within a Visual Basic program. You are limited to one server per connection, and one username and password. A different user can make another connection to the same server, however, by using a different username and password.

We recommend that you use client-side cursors instead of server-side cursors in Visual Basic. If you use a server-side cursor, the `RowCount` property on the `recordset` object will always be `-1` instead of the actual row count.

Table 43. Visual Basic and ADO Samples

File Name	Description
`SimpleADOExample.vbp`	Visual Basic project file that uses a simple ADO example with a connect string.
`modSimpleADOExample.bas`	File that is part of the `SimpleADOExample.vbp` project file.
`iholedb_databound-grid.vbp`	Visual Basic project file that displays a data-bound grid example that fetches data from the `ihRawData` table.
`frmMain.frm`	File that is part of the `iholedb_databoundgrid.vbp` project file.
`frmMain.frx`	File that is part of the `iholedb_databoundgrid.vbp` project file.

These samples are available in the following folder: `Historian\Samples\VB`

Retrieve Milliseconds

Use the following code to retrieve timestamps to a resolution of milliseconds.

```
Public Function Time_To_String_With_Milliseconds(TheTime As Double) As String

Dim Temp As String

Dim TimeFraction As Double

Dim Msc As Long

Dim TempTime As Date


On Error GoTo errc
```

```
If TheTime = 0 Then

Time_To_String_With_Milliseconds = ""

Exit Function

End If


TimeFraction = TheTime * 86400#

TimeFraction = TimeFraction - Fix(TimeFraction)


Msc = CLng(TimeFraction * 1000)


TempTime = TheTime - (TimeFraction / 86400#)

If Msc = 1000 Then

Msc = 0

TempTime = DateAdd("s", 1, TempTime)

End If


Time_To_String_With_Milliseconds = LCase(Format$(TempTime, "dd-mmm-yyyy hh:nn:ss") + "." + Format$(Msc, "000"))


errc:

End Function
```

Set a Maximum Limit to Records

Use the following example code to set a maximum limit to the number of rows returned in your query:

```
SET rstTitles = New ADODB.Recordset

rstTitles.MaxRecords = 10

strSQLTitles = "SELECT Tagname FROM ihTags"

rstTitles.Open strSQLTitles, strCnxn, adOpenStatic, adLockReadOnly, adCmdText
```

Use Parameterized Queries

Use the following example code to use parameterized queries:

```
Private Sub SampleParameterizedQuery()

    Dim ihConnectString As String

    Dim ihRecordSet     As ADODB.Recordset

    Dim ihConnection    As ADODB.Connection

    Dim ihParameter     As ADODB.Parameter

    Dim ihCommand       As ADODB.Command
```

```
    'Set Up the Historian Connect String...
    Set ihConnectString = "Provider=ihOLEDB.iHistorian.1;User Id=;Password="

    'Create Our Other Objects...
    Set ihConnection = CreateObject("ADODB.Connection")
    Set ihRecordSet = CreateObject("ADODB.Recordset")
    Set ihCommand = CreateObject("ADODB.Command")

    'Open the Connection to the Historian Archiver...
    ihConnection.ConnectionString = ihConnectString
    ihConnection.Open
    'Set up the Command Object
    With ihCommand

        'Set the Active Connection to the Historian Connection Opened Above..
        .ActiveConnection = ihConnection

        'Set the Command Text to a Parameterized Sql Statement....
        .CommandText = "select * from ihTags where datatype = ?"

        'Set the Type of the Command...
        .CommandType = adCmdText

        'Refresh Our Parameter List...
        .Parameters.Refresh
    End With

    'Create a Single Parameter Object...
    Set ihParameter = ihCommand.CreateParameter("Temp", adChar, adParamInput, 100)
    'Set the Parameters Value...
    ihParameter.Value = "SingleFloat"
    'Add the Parameter to the Command Object...
    ihCommand.Parameters.Append ihParameter
    'Run the Command!
    Set ihRecordSet = ihCommand.Execute
End Sub
```

For more information, refer to Parameterized SQL Queries *(on page 295)*.

Proficy Real-Time Information Portal

Proficy Real-Time Information Portal is a web-based tool for accessing, analyzing, and visualizing production information. It has sophisticated trending and reporting capabilities that take advantage of the vast archival and retrieval capabilities of Historian.

In Proficy Real Time Information Portal, parameters are used to build SQL queries that you can reuse with different values. In the place of a constant value in a SQL query, you can use a parameter, which takes a dynamic value at execution time. Parameterized SQL queries are driven by Proficy Real Time Information Portal components such as list boxes, combo boxes, or grids.

The SQL Query Builder application in Proficy Real Time Information Portal is used to define a parameterized query.

To define a parameterized query: In the **Specify Selected Item Wizard** or **Specify Criterion Wizard**, in the **Parameter** field, enter the name of the parameter.

The following conditions apply when you define a parameterized query:

- Parameter names must be unique.
- A question mark (?) is appended to the parameter name, and the parameter is enclosed in parentheses. For example, the parameter `temperature` becomes `{temperature?}`.
- You can specify a default value for the parameter.
- You can also select a data type for the parameter. By default, the data type is set to `char`. However, you can select `int`, `date`, `num`, or `char` as the type of database column.

Linked Servers in Microsoft SQL Server

If you want to relate Historian data with other data in SQL Server tables such as batch events, iFIX Alarms and Events collector, iDownTime data, and any other information that is available in a relational database, you can use the OLE DB provider as a linked server in Microsoft SQL Server. You can also use the OLE DB provider as a linked server if you do not want to duplicate data with an import.

With linked servers, when you query data from Historian, the SQL server fetches the requested data from Historian at the time the query is executed. Data is not duplicated because nothing is imported or stored in the SQL server. The data is simply returned as part of a query, just as any other query on a SQL Server database would return data.

Another advantage of using the OLE DB provider as a linked server is that you do not need to install Historian in the client machines. For example, a client tool such as Microsoft Query Analyzer can be used to retrieve Historian product data over the network on a computer with no Historian software installed.

Configure the OLE DB Provider as a Linked Server Manually

The following steps are necessary in order to access a linked server via the `OPENQUERY` statement.

This topic describes how to configure the OLE DB provider as a linked server manually. You can also configure it automatically *(on page 261)*.

1. From the **Start** menu, open the **SQL Server Enterprise Manager**.
2. Select an SQL server, and open the `Security` folder.
3. Right-click the `Linked Servers` folder, and select **New Linked Server**.
   The **Linked Server Properties** window appears.
4. Enter a name for the linked server, such as `iHist`.
5. In the **Provider Name** field, select **Historian OLE DB Provider**.
6. In the **Data Source** field, enter the Azure Load Balancer IP, and then select **Provider Options**.

   > **Tip:**
   > To find the Azure Load Balancer IP:
   > a. Go to the Azure portal.
   > b. Go to the **Resource Group** that was specified during deployment.
   > c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
   > d. Select or copy the IP Address.

   The **Provider Options** window appears.

   > **Note:**
   > - Select the **Level Zero Only** option only if using older versions of SQL server. For better performance while executing small queries, select the **Allow in Process** option. Clear the option if larger queries are to be executed.
   > - For configuring the Historian 64-bit OLE DB provider as a linked server, the **Allow in Process** option is mandatory.

7. Select **OK**.
8. If Historian security is enabled, enter a Historian username and password.
9. For SQL Server 2008 (32-bit/64-bit), follow these steps:

a. Select **Security**.
b. Select the **Be made using this security context** option.
c. Enter a Historian username and password in the **Remote Login** and **With Password** fields.

10. Select **OK**.

The linked server is created.

Configure the OLE DB Provider as a Linked Server Automatically

Configure a linked server and options using **Enterprise Manager**, as described in Configuring the Historian OLE DB provider as a Linked Server *(on page 260)*. Then, since the options **Allow In Process** and **Level Zero Only** apply to all linked servers that use the provider, you can create additional linked server definitions to other Historian servers using the `sp_addlinkedserver` stored procedure.

This topic describes how to configure the OLE DB provider as a linked server automatically using the `sp_addlinkedserver` system stored procedure from Microsoft SQL Server. You can also configure it manually *(on page 260)*.

1. To configure a linked server definition, use the following example code:

```
EXEC sp_addlinkedserver @server='MYSERVER_LS', @srvproduct='',

@provider='iHOLEDB.iHistorian.1', @datasrc='MY_SERVER'
```

2. To search for linked server definitions, use the following example code:

```
EXEC sp_linkedservers
```

3. To delete linked server definitions, use the following example code:

```
EXEC sp_dropserver 'MYSERVER_LS', 'droplogins'
```

Access a Linked Server

Configure a linked server and options using **Enterprise Manager**, as described in Configuring the Historian OLE DB provider as a Linked Server *(on page 260)*.

This topic describes how to access the OLE DB provider as a linked server in an SQL server using the following methods:

- **`OPENQUERY`:** This is the recommended method of accessing data by means of a linked server. To use this method, you must first configure a linked server definition. You can then use that linked server name in the `OPENQUERY` command.
- **Four-Part Name Syntax:** To use this method, you must first configure a linked server definition. You can then use that linked server name in the four-part name syntax.

- **`OPENROWSET` and `OPENDATASOURCE`:** These methods are considered adhoc methods of accessing data. They are recommended only for infrequently accessed data. When using either method, you must specify the data source, username, and password in each query instead of configuring it once in a linked server definition. If you want to limit the number of users to a defined set of servers and usernames, you can disable all methods of adhoc access by selecting the **Disallow Adhoc Accesses** option in the **Provider Options** window.

> **Note:**
> You cannot use `OPENQUERY` to access the `ihTrend` table. Use four-part name syntax to access the `ihTrend` table.

1. To fetch a list of Historian tags, run the following query:

```
SELECT * FROM OPENQUERY(iHist,'SELECT * FROM ihTags')
```

2. To fetch tag values from Historian, use the following example code:

```
SELECT TagName, TimeStamp, Value, Quality FROM OPENQUERY (iHist,'
SET
StartTime=yesterday-12Day, EndTime=Today, IntervalMilliseconds=1Hour, SamplingMode=Calculated,
 CalculationMode=Maximum
SELECT * FROM ihRawData WHERE TagName LIKE *simulation00001')
```

3. To access the ihTrend table from a linked server, run the following query:

```
SELECT * FROM iHist...[SELECT timestamp, *.value FROM ihTrend]
```

   Although the four-part name syntax works with all tables, it is only necessary to use it with the ihTrend table, because the ihTrend table does not work with `OPENQUERY`.

4. To use OPENROWSET with an SQL query, use the following example code:

```
SELECT * FROM OPENROWSET('ihOLEDB.iHistorian.1',
'MY_SERVER';'';'','SET starttime="2002-01-30 10:00:00", endtime="2002
```

> **Note:**
> This example uses double quotes around date and time because single quotes do not work inside the overall single-quoted query. It is important for you to use double quotes in this scenario.

5. To access a table, use the following example code:

```
SELECT * FROM OPENDATASOURCE('iHOLEDB.iHistorian.1', 'Data Source=MY_SERVER')...ihTags
```

6. To use OPENDATASOURCE with an SQL query and security, use the following example code:

```
SELECT * FROM OPENDATASOURCE('iHOLEDB.iHistorian.1',

'Data Source=MY_SERVER;User ID=user1;Password=thepassword')...[SE
```

7. To join Historian data with iFIX data logged with AlarmODBC, use the following example code, which determines the last date and time a specific analog tag was raised as an alarm. The date and time are then used to collect the data from the previous hour leading up to the alarm. You can use this example to determin if the value spiked into the alarm or slowly approached the alarm limit.

```
declare @var1 as varchar(300)

declare @iHistServer as varchar(10)

declare @Tagname as varchar(40)

declare @HistTagname as varchar(50)

declare @AlarmStatus as varchar(10)

declare @Node as varchar(8)

declare @StartDt as varchar(30)

declare @EndDt as varchar(30)

declare @queryDt as varchar(30)

SET @iHistServer = 'iHistMY_SERVER'

SET @Node = 'MY_SCADA'

SET @Tagname = 'Simulation00001'

SET @HistTagname = 'MY_SERVER.' + @Tagname

SET @AlarmStatus = 'HIHI'

SET @queryDt= DATEADD(day, -1, CURRENT_TIMESTAMP)

SET @EndDt = (SELECT TOP 1 DateTimeLast FROM AlarmODBC WHERE AlarmStatus = @AlarmStatus AND Node = @Node and

 Tagname =

SET @StartDt = DATEADD(hour, -1, @EndDt)

set @var1 = 'SELECT * FROM OPENQUERY

('+ @iHistServer +',''SET StartTime="'+ @StartDt +'", EndTime="'+ @Enddt +'"

SELECT Tagname, TimeStamp, Value, Quality FROM ihRawData WHERE TagName = '+ @HistTagname +''')' exec (@var1)
```

8. To access linked server data using a stored procedure, use the following example code, which interfaces with the alarm's ODBC table to get the last alarm time for a specified tag in the past 24 hours. It then uses this time to retrieve data for the tag from one hour leading up to the time the alarm occurred.
   The input parameters are the linked Historian server name, tag name, alarm status, and SCADA node name on which the alarm was created. This example uses a sim tag in the Historian database

rather than setting up a collector to an iFIX SCADA node. Preferably, an iFIX tag name must be concatenated with the node and field (`node.tagname.fieldname`).

a. To execute a stored procedure, use the following example code:

```
EXEC alarmhist 'iHistMY_SERVER', 'simulation00001', 'HIHI', 'MY_SCADA'
```

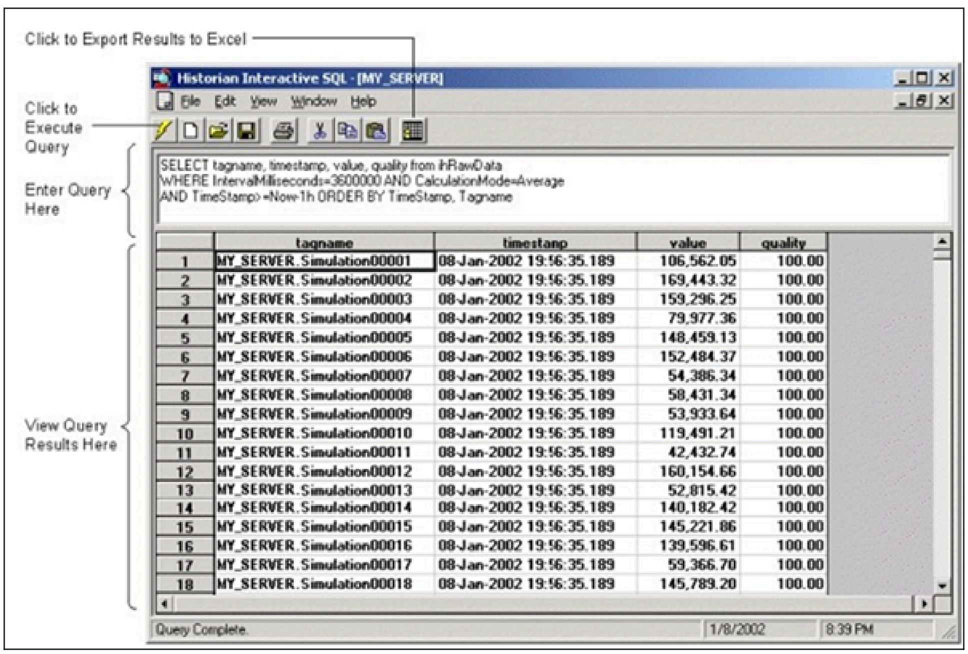
b. When you create the stored procedure in **Enterprise Manager**, include the following lines before the create procedure command to avoid an error:

```
SET ANSI_NULLS ON
GO
(@iHistServer varchar(10),
@Tagname varchar(40),
@AlarmStatus varchar(10),
@Node varchar(8))
AS
declare @var1 as varchar(400)
declare @HistTagname as varchar(50)
declare @StartDt as varchar(30)
declare @EndDt as varchar(30)
declare @queryDt as varchar(30)
declare @count as int
declare @CalculationMode as varchar(20)
SET @HistTagname = 'MY_SERVER.' + @Tagname
SET @queryDt= DATEADD(day, -1, CURRENT_TIMESTAMP)
SET @count = (SELECT COUNT(*) FROM AlarmODBC WHERE AlarmStatus = @AlarmStatus AND Node = @Node AND
 Tagname = @Tagname
If @count > 0
BEGIN
If @AlarmStatus = 'HIHI' or @AlarmStatus = 'HI'
BEGIN
SET @CalculationMode = 'Maximum'
END
ELSE
BEGIN
SET @CalculationMode = 'Minimum'
END
```

```
SET @EndDt = (SELECT TOP 1 DateTimeLast FROM AlarmODBC WHERE AlarmStatus = @AlarmStatus AND Node =
 @Node AND Tagname =
SET @StartDt = DATEADD(hour, -1, @EndDt)
SET @var1 = 'SELECT * FROM OPENQUERY
('+ @iHistServer +',''SET StartTime="'+ @StartDt +'",
EndTime="'+ @EndDt +'", IntervalMilliseconds=60000,
SamplingMode=Calculated,CalculationMode='+ @CalculationMode +'
SELECT Tagname, TimeStamp, Value, Quality FROM ihRawData WHERE TagName = '+ @HistTagname +''')'
print (@var1)
exec (@var1)
END
GO
```

About Working with Queries

Using the Historian Interactive SQL application (`ihSQL.exe`), you can run an SQL query and display the results of the query in the same window. It is useful if you want to test a query using the OLE DB provider.



It can open and save SQL queries and can show multiple windows, each containing a query request to the same server or different servers. For instance, you might want to open more than one window to compare two different time periods on the same server, or the same time period on different servers.

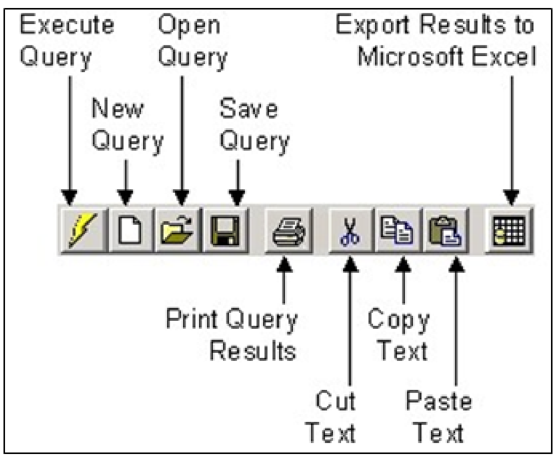
The Historian Interactive SQL application allows you to access data quickly and efficiently. Using this application, you can:

- Test SQL syntax before using it in an application.
- Troubleshoot OLE DB connections or Historian errors.
- Perform more complex searching or filtering of data than you can in the Historian SDK and administration applications.
- Retrieve data from any available Historian server.
- Save and access queries.
- Export query results to Microsoft Excel.

The Historian Interactive SQL application toolbar provides quick access to common functions such as:

- Executing queries
- Switching to a new Historian server
- Exporting query results to Microsoft Excel
- Saving a query
- Printing query results

The following figure shows the toolbar for the Historian Interactive SQL application, outlining what each button does.



Access the Historian Interactive SQL Application

When you start the application, you can log in to the default server or another Historian server.

1. From the **Start** menu, select **Programs > Historian > Historian Interactive SQL**.

> **Important:**
> The first time you use `ihSQL.exe`, you may need to select **Run As Administrator**. Otherwise, you may not be able to log in.

The **Historian Interactive SQL Login** window appears.

2. In the **Server** field, enter the Azure Load Balancer IP.

> **Tip:**
> To find the Azure Load Balancer IP:
>
> a. Go to the Azure portal.
> b. Go to the **Resource Group** that was specified during deployment.
> c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
> d. Select or copy the IP Address.

3. Enter the username and password to connect to the server. Leave the Domain field blank.
4. Select **OK**.

   A new session of the Historian Interactive SQL application appears, and it is connected to the server that you have specified. The session begins with the default values for SET variables *(on page 289)*.

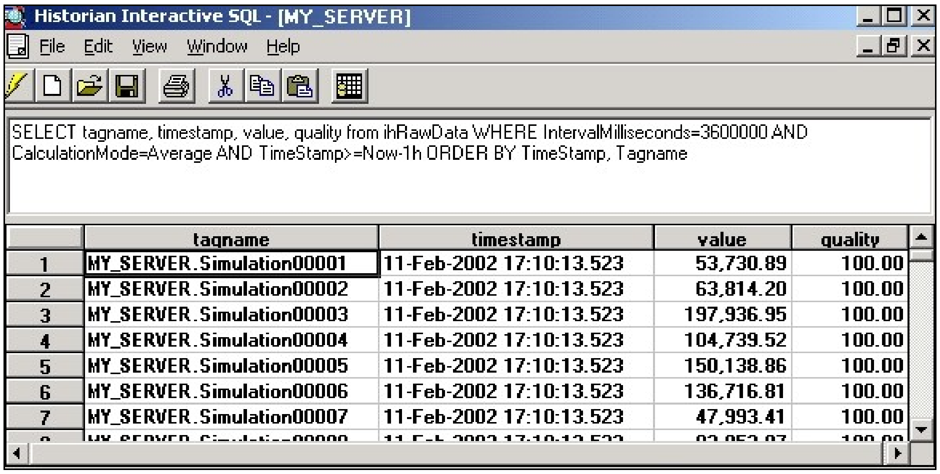
> **Note:**
> If modifications or additions are made to the list of available Historian servers using any of the Historian clients (Excel, non-web Administrator, or iFIX WorkSpace: Expression Builder and iFIX Migration Tools), those settings are global for any Historian clients running on that computer.

Run a Query

You can run a query against the data that is contained in the Historian database tables. A query is a `SET` or `SELECT` statement, or a combination of both of these SQL statements. When you execute a `SELECT` or

`SET` statement in the Historian Interactive SQL application, you can execute only one `SET` and one `SELECT` statement per query.

1. Access the Historian Interactive SQL application *(on page 266)*.
2. If you want to run a saved query, select **File > Open**, and then select the query that you want to run.
3. If you want to run a new query, enter your query in the **Query Entry** field.



4. Select  or press Ctrl+E.

   The query results appear.

Connect to a Server

The Historian Interactive SQL application allows you to make multiple connections to the same server or different servers. This allows you to look at data from different servers.

1. Access the Historian Interactive SQL application *(on page 266)*.
2. Select **File > New**.

   The **Historian Interactive SQL Login** window appears.

3. In the **Server** field, enter the Azure Load Balancer IP.

> **Tip:**
>
> To find the Azure Load Balancer IP:
>
> a. Go to the Azure portal.
>
> b. Go to the **Resource Group** that was specified during deployment.
>
> c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
>
> d. Select or copy the IP Address.

4. Enter the username and password to connect to the server. Leave the Domain field blank.
5. Select **OK**.

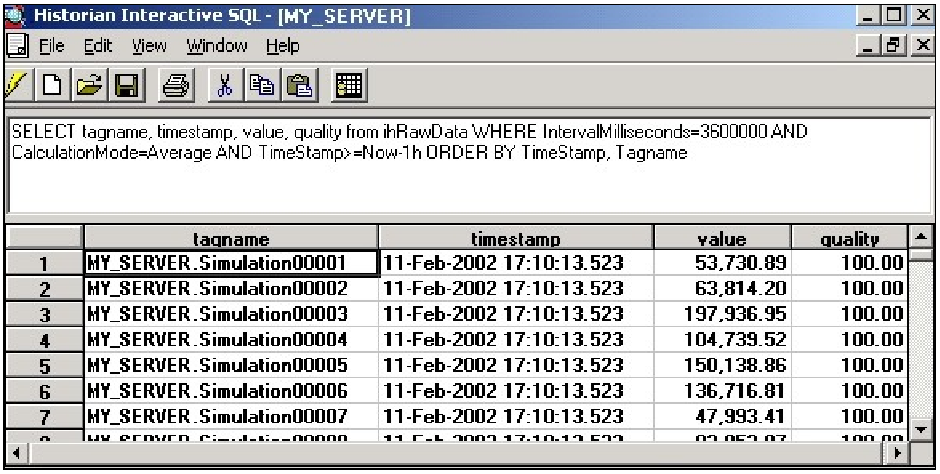
   A new session of the Historian Interactive SQL application appears, and it is connected to the server that you have specified. The session begins with the default values for SET variables *(on page 289)*.

> **Note:**
>
> If modifications or additions are made to the list of available Historian servers using any of the Historian clients (Excel, non-web Administrator, or iFIX WorkSpace: Expression Builder and iFIX Migration Tools), those settings are global for any Historian clients running on that computer.

Save a Query

When you save a query, it is saved as an `.SQL` file in the current working directory. You can later open the query in the Historian Interactive SQL application or in other client applications.

1. Access the Historian Interactive SQL application *(on page 266)*.
2. Enter your query into the **Query Entry** field.



3. Select **File > Save**.

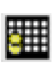
   The **Save Query to File** window appears.
4. Enter a name for the query.

> **Important:**
>
> Use the `.SQL` file extension.

5. Select .

   The query is saved in the working directory.

Export Query Results to Excel

1. Run the query that you want to export *(on page 267)*.
2. Select .

   The query results are exported to an Excel spreadsheet.

Format the date and time *(on page 255)* so that they appear correctly.

Optimize the Query Performance

To optimize query performance, follow these guidelines:

- Perform `GROUP BY` on the server whenever available. For instance, Crystal Reports gives you the option to group on the server as opposed to the client.
- Use `DISTINCT` to eliminate duplicate rows.
- Be specific when specifying tag names. For instance, when using wildcards, be as specific as possible.
- Limit the duration between start and end times.
- Get as precise a data type as possible to improve storage efficiency and allow reporting tools such as Power BI or Crystal Reports to properly format the data in reports.
- Do not rely on `TOP` or `ROWCOUNT` to optimize performance because they do not change the load on the archive or network but instead they just limit what is returned to the caller.

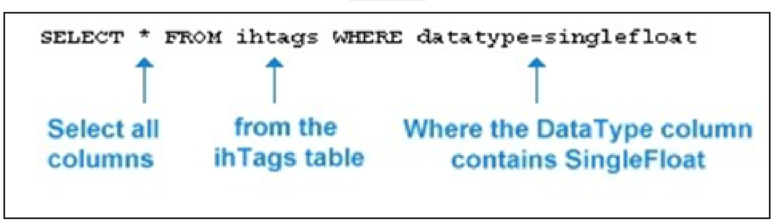
Supported SQL Syntax

The OLE DB provider supports the `SET` and `SELECT` statements in SQL queries. The following conditions apply for the supported SQL syntax:

- The supported statements follow the standard SQL-92 conventions.
- Adhering to SQL standards, these statements are not case-sensitive.
- The OLE DB provider does not allow SQL inserts, updates, deletes, or commits; therefore, there is no event notification. You can only retrieve and analyze data.
- String data types are not supported.

Some reporting packages, such as Crystal Reports, hide the SQL syntax by allowing you to use experts and wizards. However, familiarity with SQL syntax may help you in troubleshooting and tuning your SQL commands.

The following figure shows a `SELECT` statement.



With a `SELECT` statement, you can specify the Historian table and columns from which you want to retrieve data. The OLE DB provider establishes the server name at connection time. You can filter the data returned from `SELECT` by specifying a filter option in the `WHERE` clause.

Supported SELECT Statements Syntax

`SELECT` statements allow you to retrieve data from the Historian database for reporting and analysis. The `SELECT` statements that the OLE DB provider supports follow standard SQL-92 conventions. You can use `SELECT` statements to retrieve information from any of the columns in any of the Historian tables. The `SELECT` statement returns a snapshot of data at the given time of the query.

The order that you specify the columns in the `SELECT` statement controls how the data is returned. For more information on the tables and each of the columns in each table, refer to Historian Database Tables *(on page 306)* .

> **Note:**
> To query tag names with spaces in them, you must enclose the full tag name in double quotes. For example, to query the `Copy of 5vkn391s.Simulation00001` tag from the `ihTrend` table, use the following query: `SELECT "Copy of 5vkn391s.Simulation00001" from ihTrend`.

WHERE Clauses

You can use a `WHERE` clause to specify search conditions in a `SELECT` statement. You can specify a condition for any column in the table using the `WHERE` clause.

For example, you can search all rows of data in the `ihTags` table, where the `DataType` column equals `SingleFloat`. In another instance, you can find all tags that belong to a particular collector. Or, you can search for all tags with a certain poll rate, or range of poll rates, or ones with polling disabled.

You can provide maximum 200 conditions in a `SELECT` statement.

For more information on the columns for each individual Historian table, refer to Historian Database Tables *(on page 306)*.

Example 1: Search for All Single Float Tags

```
SELECT* FROM ihtags WHERE datatype=singlefloat
```

Example 2: Specify Query Parameters to Obtain String Data

```
SELECT* FROM ihrawdata WHERE tagname=SimulationString00001

AND samplingmode=interpolated

AND IntervalMilliseconds=1H
```

In this example, you change the `SamplingMode` column from the default value of `Calculated` to `Interpolated` in order to retrieve string data.

Example 3: Use a WHERE Clause to Specify a Time Range

```
SELECT* FROM ihmessages WHERE timestamp>bom
```

Example 4: Use a Complex WHERE Clause to Find All Tags With a Specific Name and Description Pattern

```
SELECT* FROM ihtags

WHERE(tagname LIKE '*001*' AND description LIKE '*sim*')

OR (tagname LIKE '*02*'

AND (description LIKE '*sec*' OR description LIKE '*sim*'))

AND (timestamptype=source OR timestamptype=collector)
```

For more information on building complex `WHERE` clauses, see Logical Operators and Parenthetical Expressions.

ORDER BY

If you do not specify `ORDER BY`, the output of the row order cannot be assumed. For example, if you want to order the rows returned from the `ihCollectors` table by the `CollectorName` column, you must include that column name in `ORDER BY`.

As a more common example, when requesting timestamps with data, use the `Timestamp` column with `ORDER BY` to ensure that the samples are sorted in order by time.

`ORDER BY` sorts the returned records by one or more specified columns in either ascending or descending order. By default, the ascending order is considered. You can order results by one or more columns. If you sort by multiple columns, the sorting priority begins with the first column listed in the query, and then the next column, and so on.

Abbreviation	Description
`ASC`	Specifies that the values must be sorted in ascending order, from lowest value to highest value.
`DESC`	Specifies that the values must be sorted in descending order, from highest value to lowest value.

The OLE DB provider treats `Null` values as the lowest possible values. It processes `ORDER BY` before it performs any `RowCount` truncation.

Example 1: Retrieve Collectors in Descending Order Sorted by the Collectorname Column

```
SELECT * FROM ihcollectors ORDER BY collectorname DESC
```

Example 2: Retrieve Messages in Ascending Order Sorted by Timestamp and Other Columns

```
SELECT * FROM ihmessages

WHERE timestamp>='5-oct-2001 00:00:00'

AND timestamp<='18-jan-2002 00:00:00'

ORDER BY timestamp, topic, username, messagenumber, messagestring
```

TOP

With the `TOP` predicate, you can limit the number of rows returned to a specified number or percentage of rows. And then, enter the rest of the query. Typically, you include `ORDER BY` in the query to sort the rows in a specified order.

When you select the top number or top percentage of rows, the returned value is limited by the `RowCount`. For instance, suppose you want the top 30 percent of rows from a query that can return a possible 10,000 rows, but the `RowCount` is set to 1000. The percentage logic processes the 3000 rows first, then it reduces the number to 1000 rows, as specified by `RowCount`. The final result returns 1000 rows, even though the top 30 percent is processed first. Use a `SET` statement or `WHERE` clause to change or disable the `RowCount` behavior.

Example 1: Return the Top 40 Tags in Alphabetical Order

```
SELECT TOP 40 * FROM ihtags ORDER BY Tagname
```

Example 2: Return the Top 10 Most Recent Messages

```
SELECT TOP 10 timestamp, topic, username, messagestring FROM

ihmessages WHERE timestamp<Now ORDER BY timestamp DESC
```

Example 3: Return the Top 10 Percent, RowCount Disabled

```
SET rowcount=0

SELECT TOP 10 PERCENT timestamp, topic, username, messagestring

FROM ihmessages WHERE timestamp<Now

ORDER BY timestamp DESC
```

LIKE

Use the `LIKE` expression when searching for column data similar to a specified text string. By using wildcards, you can specify the text strings that you want to search. You can use the wildcard before and/or after the text that you want to search for. Use an asterisk (*) for multiple unknown characters in a search string. Use a question mark (?) for a single unknown character.

> **Note:**
> You can also use a percentage (%) to select all tags that contain a specific string in the tag name and an underscore (_) to select all tags when you are unsure of only one character in the tag name. You must enclose these wildcard characters in single quotes (for example, `'%'` or `'_'`) when you use them in Historian tag names, but do not use single quotes if you want them to be treated as wildcards in SQL.

Example 1: Use LIKE With Multiple Character Replacement

```
SELECT * FROM ihtags WHERE tagname LIKE *.Simulation*

ORDER BY tagname

SELECT * FROM ihtags WHERE tagname LIKE %.Simulation%
```

Example 2: Use LIKE With Single Character Replacement

```
SELECT * FROM ihtags WHERE tagname LIKE MYSERVER.Simulation0000?

ORDER BY tagname

SELECT * FROM ihtags WHERE tagname LIKE MYSERVER.Simulation0000'_'

ORDER BY tagname
```

AS

Use `AS` when you want to control the name of an output column. You can use `AS` in all columns and tables except the `ihTrend` table. In the `ihTrend` table, you can only use `AS` with the `TimeStamp` column.

Example: Set the Output Column Name

```
SELECT status, collectorname AS Name, collectortype,

status AS 'The Status', collectordescription FROM ihcollectors
```

DISTINCT

`DISTINCT` eliminates duplicate rows when all columns are equal. Floating-point values, however, may not compare as expected, depending on the precision. For example, if the numbers to the right of the decimal point are not equal for all values, similar columns are not eliminated. The columns must be exactly equal to be eliminated.

Example 1: Retrieve the Set of Unique Data Types Used in an Archive

```
SELECT DISTINCT datatype FROM ihtags
```

Example 2: Retrieve the Set of Tags With Raw Data Samples on a Specific Date

```
SELECT DISTINCT tagname FROM ihRawData WHERE samplingmode=rawbytime

AND timestamp>='11/28/2001' AND timestamp<='11/29/2001'
```

GROUP BY

`GROUP BY` combines records with identical values in the specified field list into a single record. Then, you can compute an aggregate value for the grouped records. The aggregate column does not exist in the actual table. Another calculated column is created with the results.

Example: Group Messages by User Name and Topic

```
SELECT username, topic, COUNT(*) FROM ihmessages

WHERE timestamp >= '1-dec-2001 00:00:00'

AND timestamp <= '7-dec-2001 00:00:00'

GROUP BY username, topic ORDER BY username, topic
```

SQL Aggregate Functions

SQL aggregate functions perform a calculation on a set of values in a column and return a single value. For instance, when comparing multiple tags, you can retrieve the minimum (`MIN`) of the returned minimum values. You usually use aggregate functions with the `GROUP BY` clause, but it is not required. For more information, see Group By.

Table 44. Supported Aggregate Functions

Function	Description
`AVG`	Returns the average of the values in a group. Null values are ignored.
`COUNT`	Returns the number of items in a group. Null values are not ignored.
`MAX`	Returns the maximum value in a group. Null values are ig-nored.
`MIN`	Returns the minimum value in a group. Null values are ig-nored.

Table 44. Supported Aggregate Functions (continued)

Function	Description
`SUM`	Returns the sum of all the values in a group. `SUM` can be used with numeric columns only. Null values are ignored.
`STDEV`	Returns the statistical standard deviation of all values in a group. Null values are ignored.
`STDEVP`	Returns the statistical standard deviation for the population for all values in a group. Null values are ignored.
`VAR`	Returns the statistical variance of all values in a group. Null values are ignored.
`VARP`	Returns the statistical variance for the population for all values in a group. Null values are ignored.

STDEV, STDEVP, VAR, and VARP

If a variance is defined as the deviation from an average data set value, and *N* is the number of values in the data set, then the following equations apply:

```
VAR = (Sum of Variances)^2 / (N - 1)

VARP = (Sum of Variances)^2 / (N)

STDEV = SquareRoot (VAR)

STDEVP = SquareRoot (VARP)
```

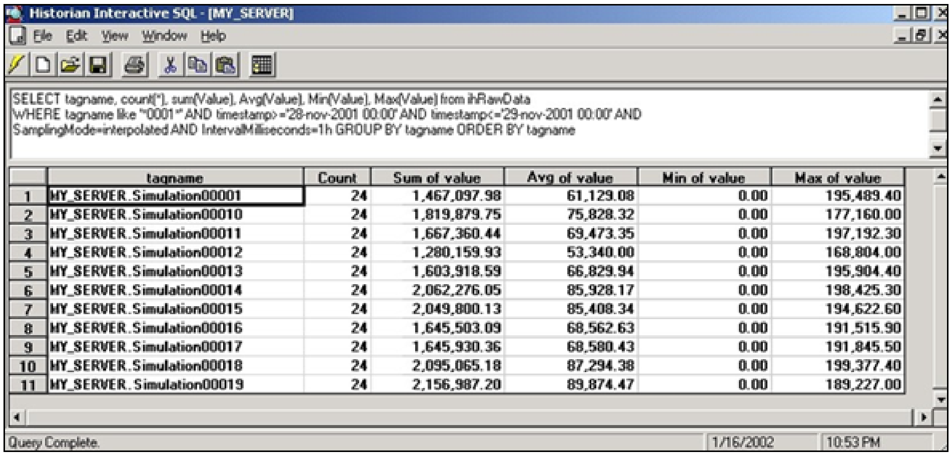
Example 1: Retrieve the Total Number of Tags

```
SELECT COUNT(*) FROM ihTags
```

Example 2: Calculate Values for Multiple Tags

```
FROM ihrawdata WHERE tagname LIKE '*0001*'

AND timestamp>='28-dec-2001 00:00'

AND timestamp<='29-dec-2001 00:00'

AND samplingmode=interpolated

AND intervalmilliseconds=1h GROUP BY tagname ORDER BY tagname
```

The following figure displays the results of this query. Note the column names (**Sum of value**, **Avg of value**, **Min of value**, and **Max of value**) returned for the calculated columns.



Conversion Functions

The Historian OLE DB provider generally returns data with the `VARIANT` data type. Some OLE DB clients may not understand `VARIANT` data, however, and will require the data to be returned as an integer, float, or string data type. To accommodate this, the OLE DB provider includes the functions described in the following table.

Table 45. Conversion Functions

Function	Description
`to_double (`*column*`)`	Converts the specified *column* to a double float data type.
`to_integer (`*column*`)`	Converts the specified *column* to a single integer data type.
`to_string (`*column*`)`	Converts the specified *column* to a string data type.

Note:

- You must edit the SQL statement manually to add conversion functions.
- You can also use the fully qualified column name (for example, `ihRawData.value`).
- Conversion functions are not available in `WHERE` or `JOIN (ON)` clauses.
- Conversion functions cannot be used within aggregate functions.

Example: Convert Values to Double Float

```
select timestamp, to_double(value), quality from ihRawData
```

JOIN

A table join is an operation that combines rows from two or more tables. You can join as many tables as you want within one `JOIN` statement. When you use a table `JOIN` in a `SELECT` statement, you must specify the column name and table when selecting the columns that you want to compare. The syntax for table joins follows standard SQL language format.

Table 46. Supported Join Operations

Supported Join Feature	Description
Inner Join	Combines records from two tables whenever there are matching values.
Left Join or Left Outer Join	Returns all of the rows from the left (first) of two tables, even if there are no matching values for records in the right (second) table.
Right Join or Right Outer Join	Returns all of the rows from the right (second) of two tables even if there are no matching values for records in the left (first) table.
Full Join or Outer Join	Returns all rows in both the left and right tables. Any time a row has no match in the other table, `SELECT` list columns from the other table contain null values. When there is a match between the tables, the entire result set row contains data values from the base tables.
Cross Join	Returns all rows from the left table. Each row from the left table is combined with all rows from the right table.
Old Join syntax	Simply selects columns from multiple tables using the `WHERE` clause without using the `JOIN` keyword.

Table joins are a powerful tool when organizing and analyzing data. A few examples are included in this section. However, refer to the documentation for your third-party reporting software for more complete information on building more complex queries.

JOIN Operations Rules

The following rules apply when working with `JOIN` operations for the Historian OLE DB provider:

- You cannot join a table with itself.
- You cannot join any table with the `ihTrend` or `ihQuerySettings` tables.

The following examples display different types of joins with the `ihComments` table. Comments themselves are not usually that useful unless they are combined with data, as you do with the `JOIN` statements in the following examples.

Example 1: Perform an Inner Join to Retrieve Only Data With Associated Comments

```
SELECT d.timestamp, d.tagname, d.value, c.username, c.comment

FROM ihrawdata d INNER JOIN ihcomments c

ON c.tagname=d.tagname AND c.timestamp=d.timestamp

WHERE d.tagname LIKE '*0001*'

ORDER BY d.timestamp, d.tagname, c.username, c.comment
```

Example 2: Perform a Left Outer Join to Retrieve All Data With and Without Comments

```
SELECT d.timestamp, d.tagname, d.value, c.comment FROM ihrawdata d

LEFT OUTER JOIN ihcomments c

ON c.tagname=d.tagname AND c.timestamp=d.timestamp

WHERE d.tagname LIKE '*0001*' ORDER BY d.timestamp, d.tagname
```

Example 3: Perform a Right Outer Join to Retrieve All Comments and Their Accompanying Data

```
SELECT d.tagname, d.timestamp, d.value, c.comment FROM ihrawdata d

RIGHT OUTER JOIN ihcomments c

ON c.tagname=d.tagname AND c.timestamp=d.timestamp

WHERE d.tagname LIKE '*0001*' ORDER BY d.tagname, d.timestamp
```

Example 4: Perform a Cross Join

```
SELECT * FROM ihCollectors CROSS JOIN ihArchives
```

Example 5: Perform a Cross Join (Older Syntax)

```
SELECT ihTags.Tagname, iharchives.Filename FROM ihTags, ihArchives
```

Example 6: Join the ihMessages and ihArchives Tables

This example uses `SET StartTime` before the `SELECT` statement. The `SET` statement is necessary because the timestamp criteria in `SELECT` do not narrow down the time range for the `ihMessages` table until after the results have been collected and the join takes place.

```
SET starttime='1-jan-2000'

SELECT a.starttime, a.endtime, m.*

FROM ihmessages m JOIN iharchives a
```

```
ON m.timestamp>=a.starttime

AND m.timestamp<=a.endtime WHERE a.iscurrent=true
```

Example 7: Interleave Data and Messages by Timestamp

```
SELECT d.timestamp, m.timestamp, d.tagname, m.messagestring,

d.value FROM ihRawData d FULL OUTER JOIN ihMessages m

ON d.timestamp=m.timestamp WHERE d.tagname=simulation00001

AND d.timestamp>='30-nov-2001 00:00:00'

AND d.timestamp<='06-dec-2001 00:00:00'
```

Example 8: Retrieve the Greatest Values Across All Simulation Tags

In the following example, we join the `ihRawData` and `ihTags` tables, because the `ihRawData` table does not contain the `CollectorType` column.

```
SELECT TOP 300 ihRawData.tagname, ihRawData.timestamp,

ihRawData.value, ihRawData.Quality FROM ihRawData

INNER JOIN ihTags ON ihRawdata.Tagname = ihTags.Tagname

WHERE ihRawData.tagname LIKE simulation*

AND ihRawData.timestamp>=11/28/2001

AND ihRawData.timestamp<=11/29/2001

AND ihRawData.samplingmode=interpolated AND ihRawData.intervalmilliseconds=1H

AND ihTags.datatype!=FixedString

AND ihTags.datatype!=variablestring

AND ihRawData.quality>0

ORDER BY value DESC, timestamp DESC
```

Example 9: Join the ihComments and ihRawData Tables

```
SET starttime='28-nov-2001 08:00', endtime='29-nov-2001 09:00',

samplingmode=interpolated, intervalmilliseconds=6m

SELECT d.tagname, d.timestamp, d.value, c.storedontimestamp, c.username,

c.datatypehint, c.comment FROM ihcomments c

FULL OUTER JOIN ihrawdata d ON c.tagname=d.tagname

AND c.timestamp=d.timestamp

WHERE d.tagname LIKE '*0001*'

ORDER BY d.tagname, d.timestamp,c.storedontimestamp, c.datatypehint,

c.username, c.comment
```

Example 10: Report by Tag Description

In the following example, we join the `ihRawData` and `ihTags` tables to get the `Description` column from the `ihTags` table.

```
SELECT d.timestamp, t.description, d.value, d.quality
FROM ihrawdata d INNER JOIN ihtags t ON d.tagname=t.tagname
WHERE d.tagname LIKE '*0001' ORDER BY d.timestamp, t.description
```

Example 11: Join Three Tables

```
SELECT ihTags.Tagname, ihTags.Description, ihRawData.TimeStamp,
ihRawData.Value, ihRawData.SamplingMode, ihComments.Comment
FROM ihTags ihTags, ihRawData ihRawData, ihComments ihComments
WHERE ihTags.Tagname = ihRawData.Tagname
AND ihRawData.Tagname = ihComments.Tagname
AND ihRawData.Timestamp = ihComments.Timestamp
AND ihRawData.TimeStamp >= {ts '2002-03-01 09:39:00.000'}
AND ihRawData.TimeStamp <= {ts '2002-03-01 09:41:00.000'}
AND ihRawData.SamplingMode = 'RawByTime'
AND ihTags.Tagname LIKE '%TestTag1%'
```

Example 12: Perform a Right Join (Older Syntax)

```
SELECT ihTags.Tagname, ihTags.CollectionInterval, ihCollectors.CollectorName,
ihCollectors.DefaultCollectionInterval
FROM ihTzzz|
```

Example 13: Perform a Left Join (Older Syntax)

```
SELECT ihTags.Tagname, ihTags.CollectionInterval, ihCollectors.CollectorName,
ihCollectors.DefaultCollectionInterval
FROM ihTags ihTags, ihCollectors ihCollectors
WHERE
ihTags.CollectionInterval *=ihCollectors.DefaultCollectionInterval
AND ihTags.Tagname LIKE '%TestTag%'
```

Quotation Marks

You must use quotation marks when you specify a string that contains a space, a comma, or a reserved word. Reserved words are defined by the SQL-92 conventions. Single and double quotes are equivalent in queries.

Example: Use Quotes When a Text String Contains a Space

```
SELECT * FROM ihtags WHERE comment LIKE 'alert message'
```

Timestamp Formats

Timestamps appear not just in the `TimeStamp` columns, but also in columns such as the `StartTime`, `EndTime`, and `LastModified` columns. You can use the date and/or time in a SQL statement that contains a timestamp. Valid date and time formats are as follows:

- System short date and time.
- SQL date and time.
- ODBC date and time.

The time format for system short timestamps is the same as the time format defined in the Windows Control Panel.

When entering a query you should use a period as the decimal separator to separate seconds from milliseconds or microseconds.

When using the SQL date and time, you should always use the English abbreviations for the desired month.

If you enter only a start time, the end time is assumed to be now. For example, if you enter `starttime > yesterday` in a `WHERE` clause, the end time for the query is now, even if you previously set an end time.

If you enter only an end time, the start time is December 31, 1969, 19:00:00.001. If you use this as the start time, you can overload the Historian server and the provider. For example, if you use `timestamp < now`, you might cause an overload.

Example 1: Use the System Short Date and Time

```
SET starttime='02/01/2002 11:00:00'
```

Example 2: Use the SQL Date and Time

```
SET starttime='14-sep-2001 11:00:00'
```

Example 3: Use the ODBC Date and Time

```
SET starttime={ts '2002-06-20 15:34:08'}
```

Example 4: Set the Start Time to 4 AM Today

```
SET starttime='04:00:00'
```

Example 5: Set the Start Time in Milliseconds

```
SET starttime='7/12/2011 12:03:16.183'
```

Example 6: Set the Start Time in Microseconds

```
SET starttime='7/12/2011 12:03:16.178439'
```

Date and Time Shortcuts

Time Segment	Meaning
`now`	Now (the time and date that you execute the query)
`today`	Today at midnight
`yesterday`	Yesterday at midnight
`mon`	The previous Monday at midnight
`tues`	The previous Tuesday at midnight
`wed`	The previous Wednesday at midnight
`thurs`	The previous Thursday at midnight
`fri`	The previous Friday at midnight
`sat`	The previous Saturday at midnight
`sun`	The previous Sunday at midnight
`boy`	First day of year at midnight
`eoy`	Last day of year at midnight
`bom`	First day of month at midnight
`eom`	Last day of month at midnight

Example 1: Set the Start Time to the First Day of the Month

```
SET starttime=bom
```

Example 2: Retrieve Messages Dated Today

```
SELECT * FROM ihmessages WHERE timestamp>=today
```

Relative Date and Time Shortcuts

Optionally, you can add or subtract relative time shortcuts to the absolute times.

Table 47. Relative Date and Time Shortcuts

Time Segment	Meaning
`s`	Second
`m`	Minute
`h`	Hour
`d`	Day
`w`	Week
`micro`	Microsecond

You can use relative time shortcuts when defining time intervals. For instance, use these shortcuts when you specify a value for the `IntervalMilliseconds` column.

> **Note:**
> You cannot use relative time shortcuts to add or subtract microseconds to or from absolute times.

Example 1: Set the Start Time to 10 Days Before Yesterday and End Time to Today

```
SET starttime=yesterday-10d, endtime=today

SELECT * FROM ihQuerySettings
```

Example 2: Retrieve the Previous 24 Hours of Messages

```
SELECT * FROM ihMessages WHERE timestamp>=Now-24h
```

Example 3: Select Data Starting at 1AM Yesterday and Ending Now

```
SELECT * FROM ihrawdata WHERE timestamp>=yesterday+1h AND timestamp<=now
```

Example 4: Retrieve Raw Data With a 1 Hour (3600000 Milliseconds) Interval Between Returned Samples

```
SELECT * FROM ihrawdata WHERE intervalmilliseconds=1h
```

Example 5: Retrieve Raw Data With a 100 Microseconds Interval Between Returned Samples

```
SELECT * FROM ihrawdata WHERE intervalmilliseconds=100micro

and starttime>= '7/12/2011 12:03:16.100000' and endtime<='
```

Example 6: Retrieve This Week's Output to Date

```
SET starttime=Sun, endtime=Now, intervalmilliseconds=1d, samplingmode=rawbytime

SELECT tagname, SUM(value) FROM ihRawData WHERE tagname LIKE *00* GROUP BY tagname
```

Comparison Operators

Table 48. Expression Comparisons

Comparison Symbol	Meaning
`<`	Less Than
`>`	Greater Than
`<=`	Less Than or Equal
`>=`	Greater Than or Equal
`=`	Equal
`!=`	Not Equal
`!>`	Not Greater Than
`!<`	Not Less Than
`BETWEEN x AND y`	Between the values *x* and *y* inclusive, where *x* and *y* are numeric values

A literal on the left side of the comparison operator is not supported. For example, this statement would fail:

```
SELECT DISTINCT tagname FROM ihRawData WHERE 50>Value
```

But the following statement succeeds since the `Value` column is to the left of the `>` operator:

```
SELECT DISTINCT tagname FROM ihRawData WHERE Value>50
```

Example 1: Retrieve Tags with a High EGU Greater Than 300

```
SELECT DISTINCT tagname, loengineeringunits, hiengineeringunits

FROM ihTags WHERE hiengineeringunits > 300
```

Example 2: Retrieve Tags with a Specific Description

```
SELECT tagname, description FROM ihTags WHERE description = "aa"
```

Example 3: Retrieve All Samples Where the Value Exceeds Query Supplied Values

```
SELECT timestamp, tagname, value FROM ihRawData

WHERE samplingmode=rawbytime AND value>75
```

Example 4: Retrieve All Samples Where the Value is Between Query Supplied Values

```
SELECT timestamp, tagname, value FROM ihRawData

WHERE samplingmode=lab AND value BETWEEN 25 AND 75
```

Example 5: Retrieve All Tag Names Starting with an A or B

```
SELECT * FROM ihtags WHERE tagname < 'C'
```

Logical Operators

The following logic operators are supported:

- `AND`
- `OR`
- `NOT`

Example 1: Use the AND Logical Operator

```
SELECT * FROM ihTags WHERE Tagname LIKE 'Simulation*'

AND CollectionInterval<3000
```

Example 2: Use the OR Logical Operator

```
SELECT * FROM ihTags WHERE Tagname LIKE 'ComputerName.Simulation*'

OR tagname LIKE '*String*'
```

Example 3: Use the NOT Logical Operator

```
SELECT * FROM ihTags WHERE NOT Datatype=SingleFloat
```

Example 4: Use the NOT Logical Operator With a LIKE Expression

```
SELECT * FROM ihTags WHERE Tagname NOT LIKE '*String*'
```

Parenthetical Expressions

Parentheses control the order of evaluation of the logical operators in an expression. The OLE DB provider supports parentheses in a `WHERE` clause. You can use multiple sets of parentheses, and nest parenthetical expressions.

Example 1: Use Parentheses

```
SELECT * FROM ihTags

WHERE (tagname LIKE *001 AND description="aa") OR tagname LIKE *002
```

Example 2: Use Parentheses with Logical Operators and Timestamps

```
SELECT * FROM ihRawData WHERE tagname=Simulation00001 AND

(Timestamp=>Tu AND Timestamp<=Wed OR Timestamp>=Fri AND time
```

Example 3: Use Multiple Sets of Parentheses

```
SELECT * FROM ihtags

WHERE (tagname LIKE '*001*' AND description LIKE '*sim*') OR

(tagname LIKE '*02*' AND (description LIKE '*sec*' OR description LIKE '*sim*'))
```

Supported SET Statement Syntax

The use of `SET` statements is not mandatory because you can also specify query parameters in a `WHERE` clause. However, `SET` statements can make your queries more readable. By using `SET` statements, you can save time by simplifying `SELECT` queries, because you do not have to retype query parameters each time you issue a new `SELECT` statement. The `SET` parameters persist for the entire session.

With a `SET` statement, you can define various defaults for your queries to use, such as:

- The start date and time of the selected data
- The end date and time
- The calculation mode
- The number of rows returned
- The data sampling mode

For more information, refer to ihQuerySettings Table *(on page 365)*.

When entering numbers, do not use a thousands separator. For example, if you want to set a collection interval to 7,000 milliseconds, use the following code:

```
SET IntervalMilliseconds = 7000
```

Correct SET Without Comma to Separate Thousands Place

Multiple `SET` statements in the same command are not supported. Combine multiple variables in the same `SET` statement.

Correct:

```
SET starttime=yesterday-10d, endtime=today, samplingmode=interpolated
```

Incorrect:

```
SET starttime=yesterday-10d
SET endtime=today
SET samplingmode=interpolated
```

SET Variables

The following table outlines the supported SQL variables and settings that you can use in a `SET` statement. If you do not change any variables using the `SET` statement or a `WHERE` clause in your `SELECT` statement, the default session variables are considered. You can apply any of the variables described in the following table to the current session. In turn, these settings are used when retrieving information from the Historian database tables. `SET` variables persist from statement to statement.

Some session variables that you define with the `SET` statement accept abbreviations. You must type at least the abbreviation for the statement to work. For instance, for the `CalculationMode` setting, you can enter the abbreviation `Interp` for the `Interpolated` setting. The accepted abbreviations are highlighted in bold in the following table.

Table 49. SET Statement Variables

Variable	Description	Default Setting
`StartTime`	A valid date and time string, such as: • `StartTime = '14-sep-200111:00:00'` • `StartTime = Now -1h` • `StartTime = '02/01/199811:00:00'` • `StartTime = {ts '2002-06-20 15:34:08'}` • `StartTime = '7/12/201112:03:16.100000'`	Two hours prior to execution of the query.
`EndTime`	A valid date and time string, such as: `EndTime = '14-sep-200112:00:00'`	The current time that you execute the query.

Table 49. SET Statement Variables (continued)

Variable	Description	Default Setting
`SamplingMode`	String that represents the mode of sampling data from the archive: • `CurrentValue` • `Interpolated` • `InterpolatedtoRaw` • `RawByTime` • `RawByNumber` • `Calculated` • `Lab` • `LabtoRaw` • `Trend` • `TrendtoRaw` • `Trend2` • `TrendtoRaw2` • `RawByFilterToggle`	`Calculated`
`Direction`	String that represents the direction of data sampling from the archive, beginning at the start time. `Direction` applies to the `RawByTime` and `RawByNumber` sampling modes: • `Forward` • `Backward`	`Forward`
`NumberOfSamples`	Any positive integer that represents the number of samples from the archive to retrieve. Do not enter a thousands separator. For example, enter `1000` and not `1,000`. Samples are evenly spaced within the time range defined by start and end times for most sampling modes. For the `RawByNumber` sampling mode, the `NumberOfSamples` attribute determines the maximum number of values to retrieve. For the `RawByTime` sampling mode, the `NumberOfSamples` is ignored.	`0` (use `IntervalMilliseconds`)
`IntervalMilliseconds`	Any positive integer that represents the interval (in milliseconds) between returned samples. For example: • If you run a query with `'IntervalMilliseconds = 100'`, it returns samples in 100-millisecond intervals. • If you run a query with `'IntervalMilliseconds = 100micro'`, it returns samples in 100-microsecond intervals.	`60000` (one minute)

Table 49. SET Statement Variables (continued)

Variable	Description	Default Setting
`CalculationMode`	The `CalculationMode` column applies only if the `SamplingMode` is set to `Calculated`. It represents the type of calculation to perform on archive data: • `Average` • `StandardDeviation` • `Total` • `Minimum` • `MaximumCount` • `RawAverage` • `RawStandardDeviation` • `RawTotal` • `MinimumTime` • `MaximumTime` • `Count` • `TimeGood` • `FirstRawValue` • `FirstRawTime` • `LastRawValue` • `LastRawTime` • `TagStats`	`Average`
`FilterTag`	A valid tagname used to define the filter. For example: `FilterTag = 'SimulationString00001'` You can specify only a single tag ID can be specified in the `FilterTag`. Wildcards are not supported. `FilterTag` is used in conjunction with `FilterValue`, `FilterComparisonMode`, and `FilterMode`.	An empty space (meaning `FilterTag` is not used)
`FilterMode`	String that represents the type of time filter: • `ExactTime` • `BeforeTime`	`BeforeTime`

Table 49. SET Statement Variables (continued)

Variable	Description	Default Setting
	• `AfterTime` • `BeforeAndAfterTime` For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and leading up to the timestamp of the archive value that triggered the `False` condition. `FilterMode` is used in conjunction with `FilterValue`, `FilterComparisonMode`, and `FilterTag`.	
`FilterComparisonMode`	String that represents the type of comparison to be made on the filter comparison value: • `Equal` • `EqualFirst` • `EqualLast` • `NotEqual` • `LessThan` • `GreaterThan` • `LessThanEqual` • `GreaterThanEqual` • `AllBitsSet` • `AnyBitSet` • `AnyBitNotSet` • `AllBitsNotSet` If you enter `FilterTag` and `FilterComparisonValue` in the `SET` statement, time periods are filtered from the results where the filter condition is `False`. `FilterComparisonMode` is used in conjunction with `FilterValue`, `FilterMode`, and `FilterTag`.	`Equal`
`FilterExpression`	An expression that includes multiple filter conditions. You can use `FilterExpression` instead of `FilterTag`, `FilterComparisonMode`, and `FilterValue`. `FilterExpression = 'BatchID=B1'` While using `FilterExpression`, the expression is passed within single quotes, and for complex expressions we write the conditions within parentheses. There is no maximum length for `FilterExpression`.	

Table 49. SET Statement Variables (continued)

Variable	Description	Default Setting
`FilterValue`	String that represents the value with which to compare the filter tag to determine the appropriate filter times. Wildcards are not supported. Do not use a comma for the thousands separator. For example: `FilterValue = 'ABCD-1086031382099'` The `FilterValue` is used in conjunction with `FilterComparisonMode`, `FilterMode`, and `FilterTag`.	An empty space (meaning filtering is not used)
`TimeZone`	String that represents the type of time zone that should be applied to timestamps: • `Client` • `Server` • Explicit bias number (number of minutes from GMT) For example, an explicit bias number of `300` represents 300 minutes from GMT. **Note:** Time zones are not supported on Windows 9x computers.	`Client`
`DaylightSavingTime`	Indicates whether Daylight Saving Time logic should be applied to timestamps. Valid values: • `True` • `False`	Date and time settings in your Windows Control Panel
`RowCount`	A number that indicates the maximum number of rows that can be returned. `0` indicates there is no limit to the number of rows returned.	`5000`

SET Statements and Variables Examples

If you do not change any variables using the `SET` statement or a `WHERE` clause in your `SELECT` statement, the default session variables are considered. For instance, if you do not specify a start and end time for your collected data, the data output from a `SELECT` statement will be the last two hours prior to execution of the query.

For example, if you want to `SELECT` all of the messages from the `ihMessages` table for the last day, you must explicitly state that you want the messages from the last day in the query. Otherwise, only the messages from the last two hours are displayed when you run the query, since that is the default setting.

`SET` statement variables persist during a session until changed. You can combine the `SET` statement on the same line as the `SELECT` statement.

Perform a Simple SET

```
SET samplingmode=currentvalue
```

Perform Multiple SETs

```
SET starttime='14-sep-2001 11:00:00', endtime='14-sep-2001 12:00:00',

samplingmode=interpolated, intervalmilliseconds=
```

Prepare for a RawByTime Query

```
SET starttime='14-sep-2001 11:00:00', endtime='14-sep-2001 12:00:00',

samplingmode=rawbytime
```

Prepare for a RawByNumber Query

```
SET starttime='14-sep-2001 11:00:00', samplingmode=rawbynumber,

numberofsamples=10, direction=backward
```

Prepare for One Hour Minimums

```
SET starttime='15-sep-2001 00:00:00', endtime='16-sep-2001 00:00:00',

samplingmode=calculated, intervalmilliseconds=36
```

Prepare for a Filtered Data Query

```
SET starttime='14-sep-2001 11:00:00', endtime='14-sep-2001 12:00:00',

samplingmode=current, filtertag='MY_SERVER.simul
```

Throttle Results with a SET Statement

```
SET ROWCOUNT = 4

SELECT Tagname FROM ihTags
```

Combined SET and SELECT Statements

The OLE DB provider allows you to execute one `SELECT` statement and one `SET` statement per query. Enter a space or a line break to indicate the end of a statement in a query. You do not need to use a semicolon (;) at the end of the line or statement.

Use SET and SELECT Statements on the Same Line

```
SET samplingmode=interpolated SELECT * FROM ihquerysettings
```

Use SET and SELECT Statements on Different Lines

```
SET samplingmode=calculated, starttime=yesterday, endtime=today

SELECT * FROM ihquerysettings
```

Parameterized SQL Queries

Parameterized SQL queries allow you to place parameters in an SQL query instead of a constant value. A parameter takes a value only when the query is executed, which allows the query to be reused with different values and for different purposes. Parameterized SQL statements are available in some analysis clients and Historian SDK.

For example, the following query contains a parameter for the collector name:

```
SELECT* FROM ihtags WHERE collectorname=? ORDER BY tagname
```

If your analysis client passes the parameter `iFIX_Albany` along with the query, it looks like follows when executed in Historian:

```
SELECT* FROM ihtags WHERE collectorname='iFIX_Albany' ORDER BY tagname
```

The advantage of using parameterized SQL queries is that you can prepare them ahead of time and reuse them for similar applications without having to create distinct SQL queries for each case. For instance, you can use the previous example in any context where you want to get tags from a collector. You can also use parameterized queries with dynamic data, where you do not know what the values will be until the statement is executed.

If your analysis client supports parameterized queries, it will automatically pass the parameter data along with a named query for Historian to process. In the case of multiple parameters, the analysis client will read the named query, and order the parameters to match.

> **Note:**
> You cannot use parameters to substitute table names or columns in a query.

Multiple Parameters

To create a query with multiple parameters, place a question mark (`?`) for every parameter whose value you want to substitute in the query. For example, if you want an SQL query to match two `WHERE` conditions, `collectorname` and `tagname`, use the following parameterized query:

```
SELECT* FROM ihtags WHERE collectorname=? AND tagname like ? ORDER BY tagname
```

When executed, the parameterized SQL query will add the parameters as they are received from the analysis application. In the previous example, the `collectorname` parameter would be received first, followed by the `tagname` parameter. Your analysis client will order the parameters based on the query it is running.

> **Note:**
> If you want to enter wildcard data in your parameterized queries, include the wildcard characters as part of the parameter. For instance, in the previous example, if you want to find any tagnames with the string iFIX in them, pass it the `*iFIX*` parameter.

Optimize the Query Performance

To optimize query performance, follow these guidelines:

- Perform `GROUP BY` on the server whenever available. For instance, Crystal Reports gives you the option to group on the server as opposed to the client.
- Use `DISTINCT` to eliminate duplicate rows.
- Be specific when specifying tag names. For instance, when using wildcards, be as specific as possible.
- Limit the duration between start and end times.
- Get as precise a data type as possible to improve storage efficiency and allow reporting tools such as Power BI or Crystal Reports to properly format the data in reports.
- Do not rely on `TOP` or `ROWCOUNT` to optimize performance because they do not change the load on the archive or network but instead they just limit what is returned to the caller.

Troubleshooting and Frequently Asked Questions

Troubleshooting

The following sections outline what to do if the following problems occur:

- Cannot Connect With the Historian Interactive SQL Application *(on page 297)*
- Cannot Log Into the Historian Interactive SQL Application *(on page 298)*
- Cannot Get Historian OLE DB provider Data *(on page 298)*
- Samples Do Not Run *(on page 298)*
- Time Zones Do Not Work *(on page 299)*
- Cannot Get String Data From the ihRawData Table *(on page 299)*
- Timestamps Include Only the Previous Two Hours *(on page 299)*
- Row Count Less Than Expected *(on page 300)*
- Linked Server Not Working *(on page 300)*
- SET Not Applied to SELECT When Using a Linked Server *(on page 300)*
- Client Crashes When Using Historian OLE DB provider *(on page 300)*

The sections that follow the answers to this list describe frequently asked questions. These answers may help you when you are first configuring and using the Historian OLE DB provider.

Cannot Connect With the Historian Interactive SQL Application

When the OLE DB provider connects to the archiver, a connection message is generated and logged to the archiver messages list. If you are having problems connecting with the Historian Interactive SQL application (`ihSQL.exe`), you will either not see a connection message or see a connection error instead.

If you suspect that you are having problems connecting to the archiver, follow these steps:

1. Access Historian Administrator *(on page 384)*.
2. Select **Messages**.
   The message fields appear in the main window.
3. In the **Priority** group box, select the **All** option.
4. In the **Topic** drop-down list, select **All Topics**.
5. Select **Search**.
   A list of messages appears on the right side of the window.
6. Scroll through the list of connection messages and look for any missing connections or connection errors.

Connections denied due to security display the user name passed to the archiver. For example, the message would be similar to this:

```
Unknown(\kmckenna) failed login at 03/01/2002 04:30:58.415 PM.
```

Cannot Log Into the Historian Interactive SQL Application

When you use `ihSQL.exe` for the first time, you may need to select **Run As Administrator**. If you do not do this the first time you use `ihSQL.exe`, you may not be able to log in. After this, you do not need to select **Run as Administrator**.

Cannot Get Historian OLE DB provider Data

If you cannot get data and you suspect there is a security problem with Historian, follow these steps to confirm that the Historian OLE DB provider is working:

1. Open the Historian Interactive SQL application and connect to the OLE DB provider.
2. Enter the following command:

   ```
   SELECT * FROM ihQuerySettings
   ```

3. Select the **Execute Query** button.
4. Confirm that data appears in the bottom half of the window:
   - If one row of data returns, then the provider is installed and working correctly, but you may have security problems between the provider and the server. You must use a valid Historian username and password.
   - If no rows return, then there is a connection problem between the client and the OLE DB provider.

The `ihQuerySettings` data is internal to the OLE DB provider and does not use any Historian security. Browsing the tables and columns also is unaffected by Historian security and is another way to confirm the connection between the client and provider.

Samples Do Not Run

If you follow the recommended installation procedures, you should not have any difficulty in running the sample reports. If you do encounter any problems, they are likely to relate to the locations of files.

For example, if you are using Crystal Reports, check that you changed the server name. If the server name is incorrect, the data links will not update correctly. See Changing the Server Name *(on page 246)* for directions on how to change it.

Time Zones Do Not Work

If you are using Windows 9x, times zones are not supported on this operating system. Returned data displays the client time zone.

If you are expecting a server or explicit bias time zone and a client time zone displays, check the defaults in the `ihQuerySettings` table. By default, the `TimeZone` column is set to `Client`. See Supported SET Statement Syntax *(on page 288)* for more information on setting defaults using a `SET` statement, or see WHERE Clauses for information on specifying a time zone in the `SELECT` statement.

Cannot Get String Data From the ihRawData Table

The Historian OLE DB provider, by default, does not return string data types in the `ihRawData` table. This is because the default `SamplingMode` value is `Calculated`. You have to change the `SamplingMode` value to `Interpolated` using the `SET` statement or a `WHERE` clause.

For example, this query does not return interpolated data:

```
SELECT * FROM ihRawData

WHERE tagname = simulationstring00001
```

However, this query does:

```
SELECT * FROM ihRawData

WHERE tagname = simulationstring00001 AND

samplingmode = interpolated
```

And so does this query:

```
SET samplingmode=interpolated

SELECT * FROM ihRawData

WHERE tagname = simulationstring00001
```

Timestamps Include Only the Previous Two Hours

By default, the data returned only includes data from two hours prior to the execution of the query. If you want to change the time frame of the data query, you need to specify a start and end time in a `SET` statement, or use a `WHERE` clause to specify a date and time period.

Row Count Less Than Expected

By default, all queries return up to a maximum of 5,000 rows. If you want to change the maximum number of rows returned, you can specify another `RowCount` value in a `SET` statement, or use the `TOP` predicate in your `SELECT` statement.

If you specify `RowCount=0` in the `SET` statement, the `RowCount` limit is disabled. However, the `RowCount` is not actually unlimited. It can be constrained by other factors such as the time interval, or by using the `TOP` predicate in your `SELECT` statement.

Linked Server Not Working

Check that you selected the **Select the Level Zero Only** and **Allow in Process** options in the **Provider Options** window. You may have forgotten to set them when you were creating your linked server. These are the only two options that should be selected.

SET Not Applied to SELECT When Using a Linked Server

Make sure that the `SET` and `SELECT` statements are combined in the same query. If you open the connection and only perform the `SET`, as shown below, the `SET` parameters only get applied for the duration of the connection.

```
SELECT * FROM OPENQUERY(linkedserver, 'SET SamplingMode=interpolated')
```

The `SamplingMode` option in the previous example does not get applied to the next `OPENQUERY` that you perform with a `SELECT` statement. The `SET` statement only gets applied to the query if it is included with the `SELECT` statement. See Use OPENQUERY to Access a Linked Server for examples of how to include the `SET` statement with a `SELECT` statement.

Client Crashes When Using Historian OLE DB provider

Ensure that your client is initializing `COM` in `Apartment` threaded mode.

Frequently Asked Questions

The following sections outline some of the most frequently asked questions when using the Historian OLE DB provider. These questions include:

How Are Historian Calculation Modes and SQL Aggregate Functions Different?

You can extract calculated data from Historian by setting the `SamplingMode` column to `Calculated` and the `CalculationMode` column to the desired calculation mode type. You can use SQL aggregate functions to perform a calculation on a set of values, possibly calculated data, for the same tag or different tags and return a single value.

For instance, when comparing multiple tags you could retrieve the minimum (`MIN`) value of each tag. By setting calculation modes, Historian Administrator only calculates the minimum for each tag over a given time period. By using aggregate functions, the Historian OLE DB provider calculates the minimum value across all tags (all rows in a table), in other words, the minimum of all minimum tag values.

How Are the ihTrend and ihRawData Tables Different?

Typically, you use the `ihTrend` table when you want to compare multiple tags at the same time. The OLE DB provider needs to synchronize all the returned data by time, so it takes more time to query the `ihTrend` table than to query the `ihRawData` table. You can retrieve multiple tags from the `ihRawData` table, but the tags are not synchronized.

Can I Run Multiple Applications Using the OLE DB provider?

Yes. For instance, you can use the OLE DB provider to access data using Crystal Reports and VisiconX at the same time.

Can I Retrieve Data From Multiple Servers?

Yes. The OLE DB provider can have connections to multiple servers at the same time. Each is regarded as a separate session.

You cannot mix multiple servers in the same `SELECT` statement, except indirectly in a linked server in Microsoft SQL Server. Crystal Reports allows you to create subreports inside of a report. Each report gets its own data source (which would be a Historian server) and its own `SELECT` query. However, the reports cannot share data. You can have multiple VisiconX data controls in one picture, each going to a different server.

For instance, say you run iFIX and Crystal Reports at the same time. From the VisiconX page, establish a connection to the Historian OLE DB provider and perform a query on Server1. Next, run a report from Crystal Reports connecting to the same provider, but with a connection to a different server, Server2. After you run the report and go back to the VisiconX page, you will notice that VisiconX is still connected to Server1. If you refresh the control, it uses the same settings and server as it did before. The provider maintains these two sessions separately, each with its own `SET` parameters.

So, in general, you can access multiple servers, but the data from each server remains independent. You must work with linked servers in Microsoft SQL Server to combine data from multiple servers.

What is a Session?

A session is defined as an OLE DB connection. You can run multiple server connections to the OLE DB provider. Each is regarded as a separate session.

You can have multiple sessions with multiple clients, such as Crystal Reports and iFIX. Multiple sessions between a client computer and a server computer count as one licensed session.

How Do the > and >= Operators Work With Timestamps?

The > and >= comparison operators, when used with `timestamp`, return the same values. For example, this SQL statement...

```
SELECT * FROM ihRawData WHERE tagname=simulation00001 AND

timestamp>='4/1/2001 01:50:00' AND

timestamp<='4/1/2001 04:00:00' AND

samplingmode=lab
```

...returns exactly the same first result as this statement:

```
SELECT * FROM ihRawData WHERE tagname=simulation00001 AND

timestamp > '4/1/2001 01:50:00' AND

timestamp <= '4/1/2001 04:00:00' AND samplingmode=lab
```

The first result is timestamped at 1:51:00.

How Do I Throttle Query Results?

The default maximum row count is 5,000. If you want to throttle the number of rows that you return in a single query, you can do one of the following:

- Use the `SET` statement to specify the `RowCount` to a specific number of rows.
- Use the `TOP` predicate to specify the top number or top percentage of rows that you want to return.
- Use the `MaxRecords` property on the `recordset` object in ADO.

When Should I Use Excel Instead of the Historian Excel Add-In?

Use the Excel Add-In when you want to get data into Microsoft Office 2003, 2007 or 2010 (32-bit or 64-bit). Use Excel with the Historian OLE DB provider when you want to perform advanced filtering, sorting, and joining of data. For other features that you might to perform with Excel and the Historian OLE DB provider, see Microsoft Excel *(on page 250)*.

Why Is the Raw Sample at the Start Time Not Returned?

Historian OLE DB provider does not return raw samples with timestamps that match the start time. If you want to include the start time, you need to set the start time to a time earlier than the first raw sample desired.

> **Note:**
> This only applies to `RawByTime` sampling mode and not `RawByNumber`.

For example, if you want to return raw samples starting at 11/28/2001 18:25:00 you can use `1/28/2001 18:24:59` as the start time. For example, you would enter the following SQL command:

```
SELECT TimeStamp, Tagname, Value FROM ihRawData

WHERE (SamplingMode = 'RawByTime') AND

(TimeStamp >= {ts '11/28/2001 18:24:59'})

ORDER BY TimeStamp ASC
```

If your timestamps are using millisecond resolution, you can retrieve timestamps starting at 11/28/2001 18:24:59.999 to prevent any sample prior to 18:50:00 from being returned.

What Username and Password Is Used if Not Specified in the Connect String?

If you leave a username and password empty in the connect string, then the user that owns the process, usually the currently logged-in user, is passed to the archiver for validation. For example, this statement leaves the username and password empty:

```
ConnectionString="Provider=ihOLEDB.iHistorian.1;User Id=;Password="
```

This statement also leaves the username and password empty:

```
ConnectionString="Provider=ihOLEDB.iHistorian.1"
```

If you saved username and password information in Historian Administrator or the iFIX WorkSpace for connecting to that server, that information is not used by the OLE DB provider.

What Is an Array Tag?

Historian allows you to store a set of values with a single timestamp and single quality and then read the elements back individually or as an array.

On retrieval, if you specify only the tag name, then all elements are returned. If you want to retrieve only an element, you can specify *<TagName>[n]* where *n* is the element number you want to retrieve.

In an array tag:

- The size of the array tag does not need to be configured. The Data Archiver will store the number of elements that were written.
- The maximum number of elements that an array tag can store is 10000. If this limit is exceeded, Historian does not accept any further elements.
- All calculation modes except `TagStats` are supported by array tags. The calculation mode is applied on array elements and not on the array. For example, if you do a minimum on a three-element array, this works like three individual tags. The minimum of element [0] over time is computed and returned as the minimum of element [0]. The Data Archiver does not compute the minimum of element [0], [1], [2] at a single point in time and return that as the minimum of the array.
- When a normal tag is converted to an array tag, on data retrieval, the data of the normal tag cannot be retrieved.

You can query both an array tag and an element of the tag. Each element of the array tag will be displayed in a separate row and they all will have the same timestamp.

What Is a User-Defined Type?

Historian gives you the ability to create a new user-defined data type which includes multiple fields of any data type and then create Historian tags of that type. All the regular tag operations can be performed on this tag. You can perform raw and calculated queries on the collected data.

What Is Not Supported?

A frequently asked question that may also relate to troubleshooting is what functions are not supported by Historian OLE DB provider. Some of these unsupported items include:

- Concatenation in SQL statements. For example, this syntax does not work:

  ```
  SELECT * FROM ihtags WHERE tagname= "MY_SERVER." + ihtags.Tagname
  ```

- Calculation in SQL statements. For example, this syntax does not work:

  ```
  SELECT * FROM ihtags WHERE ihrawdata.value * 2 > ihtags.LoEngineeringUnits
  ```

- SQL inserts, updates, deletes, or commits.

- Ordering by columns not specified in the `SELECT` statement.
- The semicolon (`;`) as a separator between `SET` and `SELECT` statements (which is commonly used in DTS and Oracle). Only a space or line break is necessary.
- Nested `SELECT` statements.
- The `UCASE` macro or other similar SQL syntax.
- `ASYNC` executes in ADO and Visual Basic.
- Bookmarks in ADO and Visual Basic.
- Table creation in SQL.
- The `UNION` statement in SQL.
- The `HAVING` clause in a `SELECT` statement.
- Using comments in a query.
- The `DISTINCT` clause in aggregate functions. For example, this syntax does not work:

```
SELECT Topic, count(DISTINCT *), sum(DISTINCT messagenumber), avg(DISTINCT messagenumber) FROM
 ihmessages GROUP BY topic ORDER BY Topic
```

- A literal on the left side of a comparison operator. SQL-92 standards support this feature, but GE Intelligent Platforms does not currently support it. For example, this syntax does not work:

```
SELECT DISTINCT tagname FROM ihRawData WHERE 50>Value
```

- Analysis of the `ihTrend` table in Crystal Reports or the Microsoft SQL Server DTS application.
- Command or connect timeouts (`Connection.ConnectTimeout`, `Connection.CommandTimeout`, or `Command.CommandTimeout`) in Visual Basic. For example, this syntax does not work:

```
SET adoConn = New ADODB.Connection
adoConn.ConnectionString = "Provider=ihOLEDB.iHistorian.1;User Id=;Password="
adoConn.ConnectionTimeout = 5 ' does nothing
adoConn.CommandTimeout = 5 ' does nothing
SET cmdTestTimer = New ADODB.Command
SET cmdTestTimer.ActiveConnection = adoConn
cmdTestTimer.CommandText = "SELECT * FROM ihtags"
cmdTestTimer.CommandType = adCmdText
cmdTestTimer.CommandTimeout = 15 ' does nothing
```

Historian Database Tables

The Historian Database Tables

The Historian database tables contain read-only data from the Historian archive.

Table Name	Description
ihTags Table *(on page 312)*	Contains Historian tag configuration information.
ihArchives Table *(on page 319)*	Contains Historian archive configuration information, plus performance statistics for each archive.
ihCollectors Table *(on page 322)*	Contains configuration and status information for each collector connected to the Historian server.
ihMessages Table *(on	Contains Historian messages such as alerts, informational topics, and connection information contained in the audit log.

Table Name	Description
page 327)	
ihRawData Table (on page 330)	Contains collected data for each tag in the Historian server. It contains not just raw data, but also calculated and interpolated data.
ihComments Table (on page 342)	Contains the comments associated with the Historian data.
ihTrend Table (on page 350)	Another way to look at collected data. Contains a row of data for each unique timestamp. You can use this table to look at your data at a summarized level. You would typically use this table to compare multiple tags with the same timestamp.
ihQuerySettings Table (on	Contains a set of parameters that apply to all queries you make in that session, unless overridden by a `WHERE` clause.

Table Name	Description
page 365)	
ihCalculationDependencies *(on page 371)*	Contains the calculation dependencies for tags.
ihAlarms Table *(on page 372)*	Contains collected alarms and events data.
ihEnumeratedSets Table *(on page 376)*	Contains information about enumerated sets.
ihEnu-	Contains information about enumerated states.

Table Name	Description
meratedStates Table (on page 377)	
ihUserDefinedTypes Table (on page 379)	Contains information about user-defined data types.
ihFields Table (on page 381)	Contains information about fields used in user-defined types.

The following conditions apply when using these tables:

- You cannot write/update data in these tables.
- Null values are not supported in any column. A blank space is returned when there is no value provided by the Historian server (instead of a `Null` field).
- Almost all columns in these tables support comparison operators except for the following:

    - `SamplingMode`
    - `Direction`
    - `NumberOfSamples`
    - `IntervalMilliseconds`
    - `CalculationMode`
    - `FilterTag`
    - `FilterMode`
    - `FilterComparisonMode`
    - `FilterValue`
    - `FilterExpression`
    - `TimeZone`
    - `DaylightSavingTime`
    - `RowCount`

  These columns only support the `=` comparison operator.

Historian Security Groups and the Database Tables

A user with membership in the iH Readers security group can access any table in the Historian OLE DB provider, even the `ihArchives` and `ihCollectors` tables. Members of the iH Readers group have read-only access to these tables.

Since the Historian OLE DB provider only supports read-only access to data and does not allow `INSERT` or `UPDATE` operations, no users can make changes to the data in these tables. This includes members of the iH Readers security group and even security administrators in the iH Security Admins security group.

For more information on Historian group rights, refer to Chapter 5 in the *Getting Started with Historian* manual.

Input Data and Historian Archive Data in Table Columns

There are two types of column data in the Historian OLE DB provider tables: input data and Historian archive data. Input data contains settings stored in the Historian OLE DB provider and has nothing to do with the data stored in the Historian archives. Historian archive data is the data retrieved from the Historian server.

While most columns contain Historian archive data, there are a few columns that contain input data. The following columns, no matter what table they appear in, contain input data and do not originate from the Historian archives:

- `SamplingMode`
- `Direction`
- `NumberOfSamples`
- `IntervalMilliseconds`
- `CalculationMode`
- `FilterTag`
- `FilterMode`
- `FilterComparisonMode`
- `FilterValue`
- `FilterExpression`
- `TimeZone`
- `DaylightSavingsTime`
- `RowCount`

The columns in the previous list are used in a `WHERE` clause to specify query parameters for retrieved data.

About the Table Descriptions

The following sections describe each table, list each column in the table, and list the data type and description for each column. The following table outlines the data types that are used throughout this chapter.

Table 50. Column Data Types

Data Type	Format of Data
`VT_BOOL`	Boolean
`VT_BSTR`	String
`VT_DBTimeStamp`	Date and Time
`VT_I4`	Integer
`VT_R4`	Float
`VT_R8`	Double Float
`VT_UI1`	Short Integer
`VT_VARIANT`	Numeric or String

Also included after each table description are examples of SQL statements used with the specified database table. These examples are only provided to get you started with creating SQL statements with the Historian OLE DB provider. For more detailed information on creating SQL queries, refer to your reporting software documentation.

ihTags Table

The `ihTags` table contains the set of tag names and the properties of each tag. Each row in the table represents one tag.

Column Name	Data Type	Description
`Tagname`	`VT_-BSTR`	`Tagname` property of the tag. **Note:** There is no length limit for Historian tag names in the Data Archiver. However, differen applications may have their own limits.
`Description`	`VT_-BSTR`	User description of the tag.
`EngUnits`	`VT_-BSTR`	Engineering units description of the tag.
`Comment`	`VT_-BSTR`	User comment associated with the selected tag.
`DataType`	`VT_-BSTR`	The data type of the tag: • `Scaled` • `SingleFloat` • `DoubleFloat` • `SingleInteger` • `DoubleInteger` • `Quad Integer` • `Unsigned Single Integer` • `Unsigned Double Integer` • `Unsigned Quad Integer` • `Byte` • `Boolean`

Column Name	Data Type	Description
		• FixedString • VariableString • BLOB The data type returned in this column is the data type that you defined in Historian Administra cation.
FixedStringLength	VT_UI1	Zero unless the data type is FixedString. If the data type is FixedString, this number represen maximum length of the string value.
CollectorName	VT_-BSTR	Name of the collector responsible for collecting data for the specified tag.
SourceAddress	VT_-BSTR	Address used to identify the tag at the data source. For iFIX systems, this is the NTF (Node.Tag
CollectionType	VT_-BSTR	Type of collection used to acquire data for the tag: • **Unsolicited:** The collector accepts data from the source whenever the source presents • **Polled:** The collector acquires data from a source on a periodic schedule determined b lector. **Note:** Not all collectors support unsolicited collection.
CollectionInter-val	VT_I4	The time interval, in milliseconds, between readings of data from this tag. For polled collection, this field represents the time between samples. For unsolicited collectio represents the minimum time allowed between samples.
CollectionOffset	VT_I4	The time shift from midnight, in milliseconds, for collection of data from this tag.
LoadBalancing	VT_-BOOL	Indicates whether the data collector should automatically shift the phase of sampling to distri activity of the processor evenly over the polling cycle. This is sometimes called phase shifting
TimeStampType	VT_-BSTR	The timestamp type applied to data samples at collection time: • **Source:** The source delivers the timestamp along with the data sample. • **Collector:** The collector delivers the timestamp along with the collected data.

Column Name	Data Type	Description
HiEngineeringUnits	VT_R8	The high end of the engineering units range. Used only for scaled data types and input scaled
LoEngineeringUnits	VT_R8	The low end of the engineering units range. Used only for scaled data types and input scaled t
InputScaling	VT_BOOL	Indicates whether the measurement should be converted to an engineering units value. When False, the measurement is interpreted as a raw measurement. When set to True, the system converts the value to engineering units by scaling the value betw HiScale and LoScale columns. If not enabled, the system assumes the measurement is alread ed into engineering units.
HiScale	VT_R8	The high-end value of the input scaling range used for the tag.
LoScale	VT_R8	The low-end value of the input scaling range used for the tag.
CollectorCompression	VT_BOOL	Indicates whether collector compression is enabled for the tag. Collector compression applies a smoothing filter to incoming data by ignoring incremental ch values that fall within a deadband centered around the last collected value. The collector pass archiver) any new value that falls outside the deadband and then centers the deadband aroun value.
CollectorDeadbandPercentRange	VT_R4	The current value of the compression deadband.
ArchiveCompression	VT_BOOL	Indicates whether archive collector compression is enabled for the tag.
ArchiveDeadbandPercentRange	VT_R4	The current value of the archive compression deadband.
CollectorGeneral1	VT_BSTR	The general (or spare) configuration fields for the tag.
CollectorGeneral2	VT_BSTR	The general (or spare) configuration fields for the tag.
CollectorGeneral3	VT_BSTR	The general (or spare) configuration fields for the tag.

Column Name	Data Type	Description
`CollectorGeneral4`	`VT_-BSTR`	The general (or spare) configuration fields for the tag.
`CollectorGeneral5`	`VT_-BSTR`	The general (or spare) configuration fields for the tag.
`ReadSecurityGroup`	`VT_-BSTR`	The name of the Windows security group that controls the reading of data for the tag. Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for def the various security levels and groups.
`WriteSecurity-Group`	`VT_-BSTR`	The name of the Windows security group that controls the writing of data for the tag. Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for def the various security levels and groups.
`AdministratorSe-curityGroup`	`VT_-BSTR`	The name of the Windows security group responsible for controlling configuration changes fo
`Calculation`	`VT_-BSTR`	The equation for the calculation performed for the tag.
`LastModified`	`VT_DB-TimeS-tamp`	The date and time that the tag configuration was last modified. The time structure includes m onds.
`LastModifiedUser`	`VT_-BSTR`	The username of the Windows user who last modified the tag configuration.
`CollectorType`	`VT_-BSTR`	The type of collector responsible for collecting data for the tag: • `Undefined` • `iFIX` • `Simulation` • `OPC` • `File` • `iFIXLabData` • `ManualEntry` • `Simulation` • `Other`

Column Name	Data Type	Description
`StoreMilliseconds`	`VT_-BOOL`	Indicates whether milliseconds are recorded in timestamps. If not enabled, the time resolution is in seconds instead of milliseconds. Maximum data comp achieved when this option is set to `False`. This is the optimum setting for most applications. **Note:** `StoreMilliseconds` returns `False` in Historian v4.5 and later.
`TimeResolution`	`String`	Indicates the timestamp resolution in seconds, milliseconds, or microseconds.
`UTCBias`	`VT_I4`	The time zone bias for the tag. Time zone bias is used to indicate the natural time zone of the pressed as an offset from UTC (Universal Time Coordinated) in minutes. UTC is the international time standard, the current term for what was commonly referred to as wich Mean Time (GMT).
`AverageCollec-tionTime`	`VT_I4`	The average time it takes to execute the calculation tag since you started the Calculation colle
`CollectionDis-abled`	`VT_I4`	Indicates whether collection is enabled (`0`) or disabled (`1`) for the tag. The default setting is en
`CollectorCompres-sionTimeout`	`VT_I4`	Indicates the maximum amount of time the collector will wait between sending samples to th This time is kept per tag, as different tags report to the archiver at different times. This value should be in increments of your collection interval, and not less. Ideally, this field is used for polled data values. It can be used with unsolicited data, but when you are dependent on the data source for the value to change. With unsolicited data, since His ly records the value when it changes, the actual time before the timeout might exceed the con timeout.
`ArchiveCompres-sionTimeout`	`VT_I4`	Indicates the maximum amount of time from the last stored point before another point is stor value does not exceed the archive compression deadband. The data archiver treats the incoming sample after the timeout occurs as if it exceeded comp then stores the pending sample.
`TimeZone`	`VT_-BSTR`	The type of time zone used:

Column Name	Data Type	Description
		• Client • Server • Explicit bias number (number of minutes from GMT)
`DaylightSaving-Time`	`VT_-BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates there is no the number of rows returned.
`InterfaceAbsolut-eDeadbanding`	`VT_-BOOL`	Indicates whether absolute collector deadband is enabled for this tag.
`InterfaceAbsolut-eDeadband`	`VT_R8`	Indicates the value for absolute collector deadband.
`ArchiveAbsolute-Deadbanding`	`VT_-BOOL`	Indicates whether absolute archive deadband is enabled for this tag.
`ArchiveAbsolute-Deadband`	`VT_R8`	Indicates the value for absolute archive deadband.
`SpikeLogic`	`VT_-BOOL`	Indicates whether Spike Logic is enabled for the tag.
`SpikeLogicOver-ride`	`VT_-BOOL`	Indicates whether the Spike Logic setting for this tag overrides the collector.
`StepValue`	`VT_-BOOL`	Indicates whether the `StepValue` property is enabled for the tag.
`EnumeratedSetName`	`VT_-BSTR`	Indicates the enumerated set name associated with a tag. You can get more information abou via the `ihEnumeratedSet` table.
`DataStoreName`	`VT_-BSTR`	Indicates the name of the data store the tag belongs to.
`NumberOfElements`	`VT_I4`	Indicates whether the tag is an array tag. If set to `-1`, the tag is an array tag. If set to `0`, the tag is not an array tag. Since the size of the a namic, there is no single number of elements that can be returned.
`CalcType`	`Enum`	Indicates whether the tag is an analytical tag or a normal tag.

Column Name	Data Type	Description
`IsAlias`	`VT_-BOOL`	Indicates whether the tag has an alias or not.

ihTags Examples

Tasks that you might want to perform on the `ihTags` table are outlined in the following examples.

Example 1: Find All Tags That Belong to a Specific Collector

```
SELECT * FROM ihtags WHERE collectorname=MYCOMPUTER_Simulation ORDER BY tagname
```

Example 2: Find All Tags With a Specific Poll Rate, a Range of Poll Rates, or Polling Disabled

```
SELECT * FROM ihtags WHERE CollectionInterval=500

OR (CollectionInterval>=1000 AND CollectionInterval<=1200)

OR CollectionInterval=0
```

Example 3: Retrieve All Tags Collected by Each Collector

```
SELECT collectorname, tagname FROM ihTags ORDER BY collectorname
```

Example 4: Retrieve All Tags With a Specific Poll Rate

```
SELECT tagname FROM ihtags WHERE collectioninterval=1000
```

Example 5: Retrieve All Tags With Subsecond Collection

```
SELECT tagname FROM ihtags

WHERE collectioninterval BETWEEN 1 AND 999
```

Example 6: Retrieve All Tags with Polling Disabled

```
SELECT tagname, collectioninterval FROM ihtags

WHERE collectioninterval=0
```

Example 7: Count the Number of Tags and Group by Collector Name

```
SELECT collectorname, COUNT(*) FROM ihTags GROUP BY collectorname
```

Example 8: Count the Number of Tags and Group by Collector Type

```
SELECT ihCollectors.collectortype, COUNT(*)

FROM ihTags INNER JOIN ihCollectors

WHERE ihTags.collectorname=ihCollectors.collectorname

GROUP BY ihcollectors.collectortype
```

Example 9: Retrieve Tags Associated With a Specific Enumerated Set

```
SELECT * FROM ihtags

WHERE EnumeratedSetName='ExampleSet'
```

ihArchives Table

Historian archives are stored as data files, each of which contains data gathered during a specific period of time.

The `ihArchives` table contains Historian archive configuration information and performance statistics for each archive. Each row in this table represents one archive. The following table describes the columns of the `ihArchives` table.

Table 51. ihArchives Table

Column Name	Data Type	Description
`ArchiveName`	`VT_BSTR`	Name of the archive for the current server if the authenticated user is a member of Historian Administrators group.
`ArchiveStatus`	`VT_BSTR`	The status of the specified archive: • `Undefined` • `Empty` • `NotEmpty`
`FileName`	`VT_BSTR`	The file name for the specified archive. The file name must be specified in the context of the Historian server drives and directories.
`IsCurrent`	`VT_BOOL`	Indicates whether the specified archive is the newest archive that new data currently flows into.
`IsReadOnly`	`VT_BOOL`	Indicates whether the read-only status is set for the specified archive.

Table 51. ihArchives Table (continued)

Column Name	Data Type	Description
`FileSizeCurrentDisk`	`VT_I4`	The actual space on the hard disk (in MB) for the specified archive.
`FileSizeCurrent`	`VT_I4`	The size of the archive file that is currently being used (in MB) for the specified archive.
`FileSizeTarget`	`VT_I4`	The target size of the specified archive file (in MB).
`StartTime`	`VT_DBTimeStamp`	The start time of the specified archive. This represents the earliest timestamp (including date and time) for any tag contained in the archive.
`EndTime`	`VT_DBTimeStamp`	The end time of the specified archive. This represents the latest timestamp (including date and time) for any tag contained in the archive.
`LastBackup`	`VT_DBTimeStamp`	The date and time the most recent online backup was performed on this archive.
`LastBackupUser`	`VT_BSTR`	The name of the user who performed the most recent online backup.

Table 51. ihArchives Table (continued)

Column Name	Data Type	Description
`LastModified`	`VT_DBTimeStamp`	The date and time that the archive was last modified. The time structure includes milliseconds.
`LastModifiedUser`	`VT_BSTR`	The username of the Windows user who last modified the archive.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT).
`DaylightSavingTime`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates there is no limit to the number of rows returned.
`DataStoreName`	`VT_BSTR`	Indicates the name of the data store the tag belongs to.

ihArchives Examples

A task that you might want to perform on the `ihArchives` table is retrieving and recording the state of the archives and archive sizes when an event happens. Recording conditions when an event happens is useful in troubleshooting.

Sample SQL statements for the `ihArchives` table are outlined in the following examples.

Example 1: Retrieve the Archive List Sorted by StartTime

```
SELECT archivename, starttime, endtime

FROM iharchives ORDER BY starttime
```

Example 2: Retrieve All Properties of the Current Archive

```
SELECT * FROM iharchives WHERE iscurrent=true
```

ihCollectors Table

The `ihCollectors` table contains the configuration and status information for each collector connected to the Historian server. Each row in this table represents a collector that is connected to the archiver. The following table describes the columns of the `ihCollectors` table.

Table 52. ihCollectors Table

Column Name	Data Type	Description
`CollectorName`	`VT_BSTR`	The name of the collector. The collector name is unique in a specific Historian server.
`CollectorDescription`	`VT_BSTR`	The user description for the collector.
`Comment`	`VT_BSTR`	The user comment associated with the collector.
`ComputerName`	`VT_BSTR`	The name of the Windows computer on which the collector is running.
`Status`	`VT_BSTR`	The status of the specified collector: • `Unknown` • `Starting` • `Running` • `Stopping` • `Stopped`
`CollectorType`	`VT_BSTR`	The type of collector responsible for collecting data for the tag: • `Undefined` • `iFIX` • `Simulation` • `OPC`

Table 52. ihCollectors Table (continued)

Column Name	Data Type	Description
		• `OPC AE` • `File` • `iFIXLabData` • `ManualEntry` • `Simulation` • `Calculation` • `ServerToServer` • `Other`
`MaximumDiskFreeBuffer-Size`	`VT_I4`	The maximum size (in MB) of the disk buffer for outgoing data.
`MaximumMemoryBuffer-Size`	`VT_I4`	The maximum size of the memory buffer (in MB) for outgoing data. The memory buffer stores data during short-term or momentary interruptions of the server connection. The disk buffer handles long-duration outages.
`ShouldAdjustTime`	`VT_BOOL`	If the data source supplies the timestamps, this value is `False`. If the collector supplies the timestamps, this value is `True`. **Note:** This column does not change collector times to match the server time. It indicates whether an increment of time is added or subtracted to compensate for the relative difference between the server and collector clocks, independent of time zone differences.
`ShouldQueueWrites`	`VT_BOOL`	Indicates whether queue writes are allowed.
`CanBrowseSource`	`VT_BOOL`	If `True`, this column indicates that the collector can browse its source for tags.

Table 52. ihCollectors Table (continued)

Column Name	Data Type	Description
`CanSourceTimestamp`	`VT_BOOL`	Indicates whether the data source can provide timestamps along with the data.
`StatusOutputAddress`	`VT_BSTR`	An address or tagname in the data source to output current collector status.
`RateOutputAddress`	`VT_BSTR`	An address or tagname in the data source into which the collector writes the current value of the events per minute output.
`HeartbeatOutputAddress`	`VT_BSTR`	The address in the source database into which the collector writes the heartbeat signal output.
`CollectorGeneral1`	`VT_BSTR`	The general (or spare) configuration fields for the collector. The `CollectorGeneral1` column is not user-defined, and is different for each collector.
`CollectorGeneral2`	`VT_BSTR`	The general (or spare) configuration fields for the collector. The `CollectorGeneral2` column is not user-defined, and is different for each collector.
`CollectorGeneral3`	`VT_BSTR`	The general (or spare) configuration fields for the collector. The `CollectorGeneral3` column is not user-defined, and is different for each collector.
`CollectorGeneral4`	`VT_BSTR`	The general (or spare) configuration fields for the collector. The `CollectorGeneral4` column is not user-defined, and is different for each collector.
`CollectorGeneral5`	`VT_BSTR`	The general (or spare) configuration fields for the collector. The `CollectorGeneral5` column is not user-defined, and is different for each collector.
`LastModified`	`VT_DBTimeStamp`	The date and time that the collector configuration was last modified. The time structure includes milliseconds.
`LastModifiedUser`	`VT_BSTR`	The username of the Windows user who last modified the collector configuration.

Table 52. ihCollectors Table (continued)

Column Name	Data Type	Description
`SourceTimeInLocalTime`	`VT_BOOL`	For data source timestamps only. Indicates whether the timestamps use local time. If the value is `False`, UTC time is used.
`CollectionDelay`	`VT_I4`	The length of time, in seconds, that the collector should delay collection at startup (to allow the data source time to initialize).
`DefaultTagPrefix`	`VT_BSTR`	The prefix that is automatically applied to all tag-names added by the specified collector.
`DefaultCollectionIn-terval`	`VT_I4`	The collection interval, in milliseconds, for tags added by the collector.
`DefaultCollectionType`	`VT_BSTR`	Type of collection used to acquire data for tags added by the collector: • **Unsolicited:** The collector accepts da-ta from the source whenever the source presents the data. • **Polled:** The collector acquires data from a source on a periodic schedule determined by the collector. **Note:** Not all collectors support unsolicited type collection.
`DefaultTimeStampType`	`VT_BSTR`	Type of timestamp applied to data samples at col-lection time for tags added by the collector: • **Source:** The source delivers the timestamp along with the data sample. • **Collector:** The collector delivers the time-stamp along with the collected data.
`DefaultCollectorCom-pression`	`VT_BOOL`	Indicates whether default collector compression is enabled for tags added by the collector.

Table 52. ihCollectors Table (continued)

Column Name	Data Type	Description
`DefaultCollectorCompressionDeadband`	`VT_R4`	The default collector compression deadband for tags added by the collector.
`DefaultCollectorCompressionTimeout`	`VT_I4`	The default collector compression timeout value.
`DisableOnTheFlyChanges`	`VT_BOOL`	Indicates whether a user can make on-the-fly changes to this tag. When enabled (`True`) you can make changes to this tag without having to restart the collector. When disabled (`False`), any changes you make to this tag do not affect collection until you restart the collector.
`DefaultSpikeLogic`	`VT_BOOL`	Indicates whether Spike Logic is enabled.
`DefaultSpikeMultiplier`	`VT_R4`	The default Spike Logic multiplier.
`DefaultSpikeInterval`	`VT_I4`	The default Spike Logic interval.
`RedundancyEnabled`	`VT_BOOL`	Indicates whether collector redundancy is enabled.
`PrincipalCollector`	`VT_BSTR`	Indicates the primary collector.
`IsActiveRedundantCollector`	`VT_BOOL`	Indicates whether the current collector is active.
`FailoverOnCollectorStatus`	`VT_BOOL`	Indicates whether the collector is set to fail over on an unknown collector status.
`FailoverOnBadQuality`	`VT_BOOL`	Indicates whether the collector is set to fail over on bad data quality received from the watchdog tag.
`FailoverOnValue`	`VT_BOOL`	Indicates whether the collector is set to fail over on a change in value.
`FailoverValueChangeType`	`VT_I4`	The value for the `FailoverOnValue` option.
`WatchdogValueMaxUnchangedPeriod`	`VT_I4`	The maximum period for an unchanged value.

Table 52. ihCollectors Table (continued)

Column Name	Data Type	Description
`WatchdogTagName`	`VT_BSTR`	The watchdog tag name.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT)
`DaylightSavingTime`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`RowCount`	`VT_I4`	The maximum number of rows that can be returned. A value of `0` indicates there is no limit to the number of rows returned.

ihCollectors Examples

One task that you might want to perform on the `ihCollectors` table could be retrieving and recording the state of the collectors when an event happens. Recording conditions when an event happens is useful in troubleshooting.

Sample SQL statements for the `ihCollectors` table are outlined in the following examples.

Example 1: Retrieve All Collectors With Status Information

```
SELECT collectorname, collectordescription AS desc, status

FROM ihcollectors
```

Example 2: Retrieve All Collectors Not Running

```
SELECT collectorname, collectordescription AS desc, status

FROM ihcollectors WHERE status!=running
```

ihMessages Table

The `ihMessages` table contains Historian messages such as alerts, informational topics, and connection information contained in the audit log. Each row in this table represents a message. The following table describes the columns of the `ihMessages` table.

Table 53. ihMessages Table

Column Name	Data Type	Description
`TimeStamp`	`VT_DBTimeStamp`	The date and time that the message was created.
`TimeStampSeconds`	`VT_DBTimeStamp`	The date and time that the message was created.
`Microseconds`	`VT_I4`	The microsecond portion of the date and time for the message.
`Topic`	`VT_BSTR`	The topic name of the message: • `AlertTopics` • `AllTopics` • `ConfigurationAudit` • `Connections` • `General` • `MessageTopicMax` • `MessageTopics` • `Performance` • `ServiceControl` • `Security` • `Undefined`
`Username`	`VT_BSTR`	Name of the Windows user who generated the message, or who the message is associated with.

Table 53. ihMessages Table (continued)

Column Name	Data Type	Description
`MessageNumber`	`VT_I4`	Message number for the message. A message number is a unique identifier associated with the message template.
`MessageString`	`VT_BSTR`	Translated text of the message, including any substitutions. Messages generally include translated fixed text and variable substitutions such as timestamps, usernames, and tagnames.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT)
`DaylightSavingTime`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates there is no limit to the number of rows returned.

ihMessages Examples

One task that you might want to perform on the `ihMessages` table is retrieving a history of alerts and messages, with timestamps and user information. For instance, you might want to query the alerts for a day, or all messages associated with a particular username.

Sample SQL statements for the `ihMessages` table are outlined in the following examples.

Example 1: Retrieve All Messages and Alerts for Today

```
SELECT * FROM ihmessages WHERE timestamp>=today
```

Example 2: Retrieve All Alert Messages for a Specific User and Time

```
SELECT * FROM ihmessages

WHERE timestamp>'12-sep-2001 02:00:00'

AND topic=AlertTopics

AND username='DataArchiver' ORDER BY timestamp
```

Example 3: Retrieve All Messages in Your Archive

```
SELECT * FROM ihMessages WHERE timestamp <= Now
```

Example 4: Retrieve All Messages for a Specific User

```
SELECT * FROM ihMessages WHERE username=operator1

AND timestamp<=Now
```

Example 5: Count All Messages by a Specific User

```
SELECT username, COUNT(*) FROM ihMessages

WHERE timestamp <=Now GROUP BY username
```

ihRawData Table

The `ihRawData` table contains any collected data for each tag contained in the Historian server. It contains not just raw data, but also calculated data and interpolated data. This table is the one typically used for reporting.

There is one row in the `ihRawData` table for each combination of tagname and timestamp. For instance, you can have two rows for the same tag, each with different timestamps. You can retrieve data for more than one tag name in a simple query.

The following table describes the columns of the `ihRawData` table.

Table 54. ihRawData Table

Column Name	Data Type	Description
`Tagname`	`VT_BSTR`	Tagname property of the tag.

Table 54. ihRawData Table (continued)

Column Name	Data Type	Description
		Note: There is no length limit for Historian tag names in the Data Archiver. However, different client applications may have their own limits.
`TimeStamp`	`VT_DBTimeStamp`	The date and time for the data sample.
`TimeStampSeconds`	`VT_DBTimeStamp`	The date and time for the data sample.
`Microseconds`	`VT_DBTimeStamp`	The microsecond interval for the data sample.
`Value`	`VT_VARIANT`	The value of the data.
`Quality`	`VT_VARIANT`	For non-raw sampled data, this column displays the percentage of good quality samples in the interval. For instance, a value of `100` means all samples in the interval are good. For raw sampled data, data values are: • `Good` • `Bad` • `Uncertain` • `Not Available` • `Really Unknown` This column also includes the subquality of the data value, if it exists: • `NonSpecific` • `ConfigError` • `NotConnected`

Table 54. ihRawData Table (continued)

Column Name	Data Type	Description
		• `DeviceFail` • `SensorFail` • `LastKnownValue` • `CommFailure` • `OutOfService` • `ScaledOutOfRange` • `OffLine` • `NoValue` • `Really Unknown`
`OPCQualityValid`	`VT_BSTR`	Indicates whether the `OPCQuality` column contains valid real OPC quality. A value of `0` indicates that you should ignore the `OPCQuality` field, and a value of `1` indicates that the `OPCQuality` column contains valid real OPC quality.
`OPCQuality`	`VT_I4`	Indicates the OPC quality as delivered by the OPC server to the Historian OPC collector. The exact meaning of the bits depends on the OPC specification and the OPC server documentation. Typically, a value of `0` represents bad quality, and a value of `192` represents good quality.
`SamplingMode`	`VT_BSTR`	The mode used to sample data from the archive: • `CurrentValue`: Retrieves the current value. Time frame criteria are ignored. • `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and the time frame criteria. • `RawByTime`: Retrieves raw archive values based on time frame criteria. • `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria. **Note:** `EndTime` criteria are ignored for this sampling mode. • `RawByFilterToggle`: Returns filtered time ranges. The values returned are `0` and `1`. If the value is `1`, then the condition is true and `0` means false. This sampling mode is used with the time range and filter tag conditions. The result starts with a starting timestamp and ends with an ending timestamp.

Table 54. ihRawData Table (continued)

Column Name	Data Type	Description
		• `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, the time frame criteria, and the `CalculationMode` criteria. • `Lab`: Returns actual collected values without interpolation. • `Trend`: Returns raw minimum and raw maximum values for each specified interval. Use this sampling mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian ignores any leftover values at the end, rather than putting them into a smaller interval. • `Trend2`: Returns raw minimum and raw maximum values for each specified interval. Use this sampling mode to maximize performance when retrieving data points for plotting. Also, if the sampling period does not evenly divide by the interval length, Historian creates as many intervals of the interval length as will fit into the sampling period, and then creates a remainder interval from whatever time is left. This sampling mode is more suitable than the `Trend` mode for analysis of minimums and maximums and for plotting programs that can handle unevenly spaced data. • `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples is less than the available samples. • `TrendtoRaw`: This sampling mode almost always produces the same results as the `Trend` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. `TrendtoRaw` is therefore used instead of `Trend` when the number of actual data samples is less than the requested number of samples. • `TrendtoRaw2`: This sampling mode almost always produces the same results as the `Trend2` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. `TrendtoRaw2` is therefore used instead of `Trend2` when the number of actual data samples is less than the requested number of samples. • `LabtoRaw`: Provides raw data for the selected calculated data when the number of samples is less than the available samples.
`Direction`	`VT_BSTR`	The direction (forward or backward from the start time) of data sampling from the archive.
`Number-`	`VT_I4`	Number of samples from the archive to retrieve.

Table 54. ihRawData Table (continued)

Column Name	Data Type	Description
`OfSamples`		Samples will be evenly spaced within the time range defined by the start and end times for most sampling modes. For the `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawByTime` mode, this value is ignored. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used.
`IntervalMilliseconds`	`VT_I4`	For non-raw sampled data, this column represents a positive integer for the time interval (in milliseconds) between returned samples. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used.
`CalculationMode`	`VT_BSTR`	This column applies only if the `SamplingMode` is set to `Calculated`. It represents the type of calculation to perform on archive data: • `Average` • `Count` • `Maximum` • `MaximumTime` • `Minimum` • `MinimumTime` • `StandardDeviation` • `Total` • `RawAverage` • `RawStandardDeviation` • `RawTotal` • `TimeGood` • `FirstRawValue` • `FirstRawTime` • `LastRawValue`

Table 54. ihRawData Table (continued)

Column Name	Data Type	Description
		• `LastRawTime` • `TagStats`
`FilterTag`	`VT_BSTR`	Tagname used to define the filter, if specified. Only a single tag can be specified, and wildcards are not supported.
`FilterMode`	`VT_BSTR`	The type of time filter: • `ExactTime`: Retrieves data for the exact times that the filter condition is `True`. • `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. • `AfterTime`: Retrieves data from the time of the last `True` filter condition to the next `False` condition. • `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the next `False` condition. This mode defines how time periods before and after transitions in the filter condition should be handled. For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and leading up to the timestamp of the archive value that triggered the `False` condition.
`FilterComparisonMode`	`VT_BSTR`	The type of comparison to be made on the filter comparison value: • `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value. • `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value. • `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value. • `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value. • `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value. • `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value. • `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value.

Table 54. ihRawData Table (continued)

Column Name	Data Type	Description
		• `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value. • `AllBitsSet`: Filter condition is `True` when the binary `FilterTag` value is equal to all the bits in the condition. It is represented as `^` to be used in `FilterExpression`. • `AnyBitSet`: Filter condition is `True` when the binary `FilterTag` value is equal to any of the bits in the condition. It is represented as `~` to be used in `FilterExpression`. • `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is represented as `!~` to be used in `FilterExpression`. • `AllBitsNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to all the bits in the condition. It is represented as `!^` to be used in `FilterExpression`. This column defines how archive values for the `FilterTag` value should be compared to the `FilterValue` value to establish the state of the filter condition. If `FilterTag` and `FilterComparisonValue` values are specified, time periods are filtered from the results where the filter condition is `False`.
`FilterValue`	`VT_BSTR`	The value with which to compare the `FilterTag` value to determine appropriate filter times.
`FilterExpression`	`VT_BSTR`	An expression which includes one or more filter conditions. The type of conditions used are: • `AND` • `OR` • Combination of `AND` and `OR` `FilterExpression` can be used instead of the `FilterTag`, `FilterComparisonMode` and `FilterValue` parameters. While using `FilterExpression`, the expression is passed within single quotes. For complex expressions, write the conditions within parentheses. There is no maximum length for the `FilterExpression` value, but if called using OLE DB or Excel, those tools may have their own limitations.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT)

Table 54. ihRawData Table (continued)

Column Name	Data Type	Description
`Day-light-Sav-ing-Time`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`Row-Count`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.

The `ihRawData` table can generate a large number of rows if not used with caution. You can easily generate queries which take a very long time to complete and put stress on the archiver and generate network traffic.

ihRawData Examples

Tasks that you might want to perform on the `ihRawData` table are outlined in the following examples.

Example 1: Retrieve All Samples With a Value Outside the Query Supplied Values

```
SELECT * FROM ihRawData WHERE value<140000 OR value>150000
```

Example 2: Retrieve All Bad Samples (Raw Data)

```
SELECT * FROM ihRawData WHERE quality NOT LIKE good*

AND samplingmode=RawbyTime
```

Example 3: Count Bad Samples (Raw Data)

```
SELECT COUNT(*) FROM ihRawData WHERE quality NOT LIKE good*

AND samplingmode=RawbyTime
```

Example 4: Retrieve All Bad Samples Over the Last Day (Interpolated Data)

```
SELECT timestamp, tagname, value, quality FROM ihRawData

WHERE samplingmode=rawbytime

AND Quality NOT LIKE good*

AND timestamp>=Now-24H
```

Example 5: Use an Explicit Time Zone

```
SELECT * FROM ihRawData WHERE timezone=300
```

Example 6: Perform a Simple Sequence of Events

```
SELECT timestamp, tagname, value, quality FROM ihrawdata

WHERE samplingmode=rawbytime ORDER BY timestamp
```

Example 7: Report the Busiest Tags

```
SELECT tagname, value FROM ihRawData

WHERE samplingmode=calculated

AND calculationmode=count

AND numberofsamples=1

AND timestamp>='07/30/2002 10:00:00'

AND timestamp<='07/30/2002 11:00:00' order by value descending
```

Example 8: Retrieve All Bad Samples Over the Last Day

```
SELECT timestamp, tagname, value, quality FROM ihRawData

WHERE samplingmode=rawbytime

AND Quality NOT LIKE good*

AND timestamp>=Now-24H
```

Example 9: Retrieve All Bad Samples, Ignore End of Collection Markers

```
SELECT timestamp, tagname, value, quality FROM ihRawData

WHERE samplingmode=rawbytime

AND Quality NOT LIKE good*

AND quality NOT LIKE 'bad offline' AND timestamp>=Now-24H
```

Example 10: Count Bad Samples, Ignore End of Collection Markers

```
SELECT COUNT(*) FROM ihRawData WHERE samplingmode=rawbytime

AND Quality NOT LIKE good* and Quality NOT LIKE 'bad offline'

AND timestamp>=Now-24H
```

Example 11: Obtain All Raw Samples With Comments From Yesterday

```
SELECT ihRawData.Tagname, ihRawData.TimeStamp, ihRawData.Value

FROM ihRawData

INNER JOIN ihComments ON ihComments.Tagname = ihRawData.Tagname
```

```
AND ihComments.Timestamp = ihRawData.Timestamp

AND ihComments.Comment = "The comment" WHERE samplingmode=rawbytime

AND ihComments.Timestamp > Yesterday

AND ihComments.Timestamp < Today
```

Example 12: Determine the Number of Milliseconds Per Interval With Good Data

```
SELECT timestamp, tagname, value as TimeGood, quality, intervalmilliseconds FROM ihRawData

WHERE tagname=Denali.Simulation00001

AND samplingmode=calculated

AND calculationmode=timegood

AND intervalmilliseconds=10s

AND timestamp>='1/20/2003 13:18:00'

AND timestamp<='1/20/2003 13:20:00'
```

Example 13: Retrieve Raw Minimum and Maximum Values Per Interval

In this example, you use the data retrieved from the query (with the `Trend` sampling mode) to plot points.

```
SELECT timestamp, tagname, value, quality

FROM ihRawData

WHERE tagname=dFloatTag5

AND samplingmode=trend

AND intervalmilliseconds=24h

AND timestamp>='1/01/2003 07:00:00'

AND timestamp<='1/10/2003 12:00:00'
```

Example 14: Retrieve Data with Native Values and Tags Associated With Enumerated Sets

If the `enumnativevalue` query modifier is not set, the data is retrieved with string values by default. If it is set, the raw values are retrieved. These values are then retrieved by default for the current session and will only change when you open a new session.

```
SELECT * from ihrawdata

WHERE samplingmode='rawbytime' and tagname=mytag AND criteriastring='#enumnativevalue'


SELECT timestamp,value,quality from ihrawdata WHERE tagname = MyTag AND samplingmode=Interpolated and numberofsamples=6

 and criteriastring='#enumnativevalue'


SET criteriastring='#enumnativevalue'
```

```
SELECT * from ihrawdata

WHERE samplingmode='rawbytime' and tagname=mytag
```

Example 15: Retrieve Average Values for Enumerated Sets

```
SET criteriastring='#enumnativevalue'


SELECT * from ihrawdata

WHERE tagname LIKE Call AND samplingmode=calculated AND calculationmode=average
```

ihHabAlarms Table

The `ihHabAlarms` table contains alarm data collected from Habitat by the HAB collector. This data is stored in the Historian archive files.

Column Name	Data Type	Description
`tagname`	`VT_BSTR`	The tagname property of the tag.
`timestamp`	`VT_DBTimeStamp`	The date and time for the data sample (based on the timestamp of collector)
`timestampseconds`	`VT_I4`	The date and time for the data sample.
`microseconds`	`VT_I4`	The microsecond interval for the data sample.
`samplingmode`	`VT_BSTR`	The mode used to retrieve data from the archive. Only the RawByTime sampling mode is used for alarms. It retrieves raw archive values for a time period.
`quality`	`VT_BSTR`	The quality of the tag data. For raw sampled data, the valid data values is Good.

Column Name	Data Type	Description
`text`	`VT_BSTR`	The alarms message
`location`	`VT_BSTR`	The substation or location as defined in the Habitat database
`priority`	`VT_BSTR`	The alarm priority as defined in the Habitat database
`category`	`VT_BSTR`	The alarm category as defined in the Habitat database
`exception`	`VT_BSTR`	The alarm exception as defined in the Habitat database
`area`	`VT_BSTR`	The alarm area as defined in the Habitat database
`fieldtime`	`VT_DBTimeStamp`	The field time of the alarm message (if available)
`fmillisec`	`VT_I4`	The milliseconds part of the field time
`fnanosec`	`VT_I4`	The nanoseconds part of the field time
`time`	`VT_DBTimeStamp`	The time of the alarm generated in Habitat (mapped to TIME_CIRCLG)
`timefmt`	`VT_I4`	

Column Name	Data Type	Description
`tnanosec`	`VT_I4`	The nanoseconds part of the alarms time
`rowcount`	`VT_I4`	The maximum number of rows that can be returned. A value of 0 indicates that there is no limit to the number of rows returned.

ihComments Table

The `ihComments` table contains the annotations associated with the collected data. There is a separate row of data in the `ihComments` table for each comment associated with a tag. For instance, you can have five rows that contain the same tag and timestamp, but each contain a different comment value.

It is possible to have different data types of annotations. Comments are most often strings, but can be binary numbers or BLOBs. Only string comments are returned in the `ihComments` table.

The following table describes the columns of the `ihComments` table.

Table 55. ihComments Table

Column Name	Data Type	Description
`Tagname`	`VT_BSTR`	Tagname property of the tag. **Note:** There is no length limit for Historian tag names in the Data Archiver. However, different client applications may have their own limits.
`TimeStamp`	`VT_DBTimeStamp`	The date and time that the data was generated.
`TimeStampSeconds`	`VT_DBTimeStamp`	The date and time that the data was generated.
`Microseconds`	`VT_I4`	The microsecond portion of the date and time.
`StoredOnTimeStamp`	`VT_DBTimeStamp`	The date and time that the comment was generated.
`StoredOnTimeStamp`	`VT_DBTimeStamp`	The time that the comment was added to the archive.

Table 55. ihComments Table (continued)

<table>
<tr><th>Column Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>SuppliedUsername</td><td>VT_BSTR</td><td>The username of the currently logged-in Windows user at the time that the comment was entered.</td></tr>
<tr><td>Username</td><td>VT_BSTR</td><td>Username provided along with the comment.</td></tr>
<tr><td>Comment</td><td>VT_BSTR</td><td>The actual comment.</td></tr>
<tr><td>DataTypeHint</td><td>VT_BSTR</td><td>Name of the data type for the comment:
• String
• Read-only
• Optional</td></tr>
<tr><td>SamplingMode</td><td>VT_BSTR</td><td>The mode used to sample data from the archive:
• CurrentValue: Retrieves the current value. Time frame criteria are ignored.
• Interpolated: Retrieves evenly spaced interpolated values based on interval or NumberOfSamples and time frame criteria.
• RawByTime: Retrieves raw archive values based on time frame criteria.
• RawByNumber: Retrieves raw archive values based on the StartTime, NumberOfSamples, and Direction criteria.

Note: EndTime criteria are ignored for this sampling mode.

• RawByFilterToggle: Returns filtered time ranges. The values returned are 0 and 1. If the value is 1, then the condition is true and 0 means false. This sampling mode is used with the time range and FilterTag conditions. Results have starting and ending timestamps.</td></tr>
</table>

Table 55. ihComments Table (continued)

Column Name	Data Type	Description
		• `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria. • `Lab`: Returns actual collected values without interpolation. • `Trend`: Returns raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian ignores any leftover values at the end instead of putting them into a smaller interval. • `Trend2`: Returns raw minimum and maximum values for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian creates as many intervals of the interval length as will fit into the sampling period, and then creates a remainder interval from whatever time is left. This sampling mode is more suitable than the `Trend` mode for analysis of minimums and maximums and for plotting programs that can handle unevenly spaced data. • `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples requested is less than the number of available samples. • `TrendtoRaw`: This mode almost always produces the same results as the `Trend` mode. The exception is that when a greater num-

Table 55. ihComments Table (continued)

Column Name	Data Type	Description
		ber of samples are requested than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend` when the number of actual data samples is less than the requested number of samples. • `TrendtoRaw2`: This sampling mode almost always produces the same results as the `Trend2` mode. The exception is that when a greater number of samples are requested than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend2` when the number of actual data samples is less than the requested number of samples. • `LabtoRaw`: Provides raw data for the selected calculated data when the number of samples is less than the number of available samples.
`Direction`	`VT_BSTR`	The direction (forward or backward from the start time) of data sampling from the archive.
`NumberOfSamples`	`VT_I4`	Number of samples from the archive to retrieve. Samples will be evenly spaced within the time range defined by start and end times for most sampling modes. For the `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawByTime` mode, this value is ignored.

Table 55. ihComments Table (continued)

Column Name	Data Type	Description
		Note: The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used.
`IntervalMilliseconds`	`VT_I4`	For non-raw sampled data, this column represents a positive integer for the time interval (in milliseconds) between returned samples. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used.
`CalculationMode`	`VT_BSTR`	The calculation mode, if used.
`FilterTag`	`VT_BSTR`	Tagname used to define the filter, if specified. Only a single tag can be specified, and wildcards are not supported.
`FilterMode`	`VT_BSTR`	The type of time filter: • `ExactTime`: Retrieves data for the exact times that the filter condition is `True`. • `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. • `AfterTime`: Retrieves data from the time of the last `True` filter condition to the time of the next `False` condition. • `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `False` condition.

Table 55. ihComments Table (continued)

Column Name	Data Type	Description
		The `FilterMode` defines how time periods before and after transitions in the filter condition should be handled. For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and ending at the timestamp of the archive value that triggered the `False` condition.
`FilterComparisonMode`	`VT_BSTR`	The type of comparison to be made on the filter comparison value: • `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value. • `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value. • `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value. • `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value. • `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value. • `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value. • `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value.

Table 55. ihComments Table (continued)

Column Name	Data Type	Description
		• `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value. • `AllBitsSet`: Filter condition is `True` when the binary `FilterTag` value is equal to all the bits in the condition. It is represented as `^` to be used in `FilterExpression`. • `AnyBitSet`: Filter condition is `True` when the binary `FilterTag` value is equal to any of the bits in the condition. It is represented as `~` to be used in `FilterExpression`. • `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is represented as `!~` to be used in `FilterExpression`. • `AllBitsNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to all the bits in the condition. It is represented as `!^` to be used in `FilterExpression`. This column defines how archive `FilterTag` values should be compared to `FilterValue` values to establish the state of the filter condition. If `FilterTag` and `FilterComparisonValue` values are specified, time periods are filtered from the results where the filter condition is `False`.
`FilterValue`	`VT_BSTR`	The value with which to compare the `FilterTag` value to determine appropriate filter times.
`FilterExpression`	`VT_BSTR`	An expression which includes one or more filter conditions. The type of conditions used are:

Table 55. ihComments Table (continued)

Column Name	Data Type	Description
		• `AND` • `OR` • Combination of `AND` and `OR` This column can be used instead of the `FilterTag`, `FilterComparisonMode`, and `FilterValue` columns. While using `FilterExpression`, the expression is passed within single quotes, and for complex expressions you write the conditions within parentheses. There is no maximum length for `FilterExpression`, but if called using OLE DB or Excel, these tools may have their own limitations.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT)
`DaylightSavingTime`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.

ihComments Examples

Example SQL statements for the `ihComments` table are outlined in the following examples.

Example 1: Retrieve All Comments for a Specific Tag for This Month

```
SELECT * FROM ihcomments WHERE tagname LIKE '*001'

AND timestamp>bom
```

Example 2: Retrieve Comments That Contain a Substring

```
SELECT * FROM ihcomments WHERE comment LIKE '*abc*'
```

Example 3: Retrieve All Comments in an Archive

```
SELECT * FROM ihComments WHERE timestamp<=Now

AND samplingmode=rawbytime
```

ihTrend Table

The `ihTrend` table allows you to compare multiple tags for the same timestamp. It contains a row of data for each unique timestamp, but with columns from one or more tags. The column names are dynamic and determined by the returned tag names. The `ihTrend` table is similar to a pivot table or, for instance, a cross-tab report that you can create in Crystal Reports.

The `ihTrend` table can store up to 100 columns in a returned set. This allows you to compare `Value` columns with up to 99 tags for a single timestamp, or `Value` and `Quality` columns with up to 49 tags.

> **Note:**
> Currently, you cannot analyze the `ihTrend` table in Crystal Reports or the Microsoft SQL Server DTS application.

The following table describes the columns of the `ihTrend` table, including all possible tag columns. Different queries on this table can produce different column results.

> **Note:**
> In all column names in the following table, *TagID* is used as a placeholder for the actual tag name.

Table 56. IhTrend Table

Column Name	Data Type	Description
`TimeStamp`	`VT_DBTimeStamp`	The date and time that the trend was generated.
`TimeStampSeconds`	`VT_DBTimeStamp`	The date and time for the data sample.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
`Microseconds`	`VT_I4`	The microsecond interval for the data sample.
`SamplingMode`	`VT_BSTR`	The mode of sampling data from the archive: • `CurrentValue`: Retrieves the current value. Time frame criteria are ignored. • `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and time frame criteria. • `RawByTime`: Retrieves raw archive values based on time frame criteria. • `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria. **Note:** `EndTime` criteria are ignored for this mode. • `RawByFilterToggle`: Returns filtered time ranges. The values returned are `0` and `1`. If the value is `1` then the condition is true and `0` means false. This mode is used with the time range and `FilterTag` conditions. Results start and end with timestamps. • `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria. • `Lab`: Returns actual collected values without interpolation. • `Trend`: Returns the raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian ignores any leftover values at the end instead of putting them into a smaller interval. • `Trend2`: Returns the raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian puts leftover values into a remainder interval. This mode is more suitable than the `Trend` mode for analysis of minimums and maximums and for plotting programs that can handle unevenly spaced data. • `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples is less than the number of available samples.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
		• `TrendtoRaw`: This mode almost always produces the same results as the `Trend` mode. The excep-tion is that when the number of samples requested is greater than the number of raw data point this mode returns all available raw data points with no further processing. This mode is therefor used instead of `Trend` when the number of actual data samples is less than the requested num-ber of samples. • `TrendtoRaw2`: This mode almost always produces the same results as the `Trend2` mode. The ex-ception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend2` when the number of actual data samples is less than the re-quested number of samples. • `LabtoRaw`: Provides raw data for the selected calculated data when the number of requested san ples is less than the number of available samples.
`Direc-tion`	`VT_BSTR`	The direction (forward or backward from the start time) of data sampling from the archive.
`Number-OfSam-ples`	`VT_I4`	Number of samples to retrieve from the archive. Samples will be evenly spaced within the start and end times defined for most sampling modes. For th `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawBy-Time` mode, this column is ignored. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSam-ples` is used, `IntervalMilliseconds` is not used.
`Inter-valMil-lisec-onds`	`VT_I4`	For non-raw sampled data, this column represents a positive integer for the time interval (in millisec-onds) between returned samples. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSam-ples` is used, `IntervalMilliseconds` is not used.
`Calcula-tionMode`	`VT_BSTR`	This column applies only if the `SamplingMode` is set to `Calculated`. It represents the type of calculation t perform on archive data:

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
		• `Average` • `Count` • `Maximum` • `MaximumTime` • `Minimum` • `MinimumTime` • `StandardDeviation` • `Total` • `RawAverage` • `RawStandardDeviation` • `RawTotal` • `TimeGood` • `FirstRawValue` • `FirstRawTime` • `LastRawValue` • `LastRawTime` • `TagStats`
`Filter-Tag`	`VT_BSTR`	Tagname used to define the filter, if specified. Only a single tag can be specified. Wildcards are not supported.
`Filter-Mode`	`VT_BSTR`	The type of time filter: • `ExactTime`: Retrieves data for the exact times that the filter condition is `True`. • `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. • `AfterTime`: Retrieves data from the time of the last `True` filter condition to the time of the next `False` condition. • `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `False` condition. This value defines how time periods before and after transitions in the filter condition should be handled.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
		For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and ending at the timestamp of the archive value tha triggered the `False` condition.
`Filter-Compar-isonMode`	`VT_BSTR`	The type of comparison to be made on the filter comparison value: • `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value. • `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison val ue. • `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison valu • `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value. • `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value. • `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison va ue. • `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the com parison value. • `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to th comparison value. • `AllBitsSet`: Filter condition is `True` when the binary `FilterTag` value is equal to all the bits in the condition. It is represented as `^` to be used in `FilterExpression`. • `AnyBitSet`: Filter condition is `True` when the binary `FilterTag` value is equal to any of the bits in the condition. It is represented as `~` to be used in `FilterExpression`. • `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is represented as `!~` to be used in `FilterExpression`. • `AllBitsNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to all the bits in the condition. It is represented as `!^` to be used in `FilterExpression`. This column defines how archive values for the `FilterTag` value should be compared to the `FilterVal-ue` value to establish the state of the filter condition. If `FilterTag` and `FilterComparisonValue` values are specified, time periods are filtered from the results where the filter condition is `False`.
`Filter-Value`	`VT_BSTR`	The value with which to compare the `FilterTag` value to determine appropriate filter times.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
`FilterExpression`	`VT_BSTR`	An expression which includes one or more filter conditions. The type of conditions used are: • `AND` • `OR` • Combination of `AND` and `OR` This column can be used instead of the `FilterTag`, `FilterComparisonMode`, and `FilterValue` columns. While using `FilterExpression`, the expression is passed within single quotes. For complex expressions you write the conditions within parentheses. There is no maximum length for `FilterExpression`, but if called using OLE DB or Excel, these tools may have their own limitations.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT)
`DaylightSavingTime`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.
`TagID``.Value`	`VT_VARIANT`	The value of the data for the specified tag ID.
`TagID``.Quality`	`VT_VARIANT`	For non-raw sampled data, this column displays the percentage of good quality samples in the interval. For instance, a value of `100` means all samples in the interval are good. For raw sampled data, data values are: • `Good` • `Bad` • `Uncertain` • `Not Available` • `Really Unknown`

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
		This column also includes the subquality of the data value, if it exists: • `NonSpecific` • `ConfigError` • `NotConnected` • `DeviceFail` • `SensorFail` • `LastKnownValue` • `CommFailure` • `OutOfService` • `ScaledOutOfRange` • `OffLine` • `NoValue` • `Really Unknown`
Tag-ID.Tagname	`VT_BSTR`	Tagname property of the specified tag ID.
Tag-ID.Description	`VT_BSTR`	User description for the specified tag ID.
Tag-ID.EngUnits	`VT_BSTR`	Engineering unit description for the specified tag ID.
Tag-ID.Comment	`VT_BSTR`	User comment associated with the specified tag ID.
Tag-ID.DataType	`VT_BSTR`	The data type for the specified tag ID: • `Scaled` • `SingleFloat` • `DoubleFloat`

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
		• `SingleInteger` • `DoubleInteger` • `QuadInteger` • `UnsignedSingleInteger` • `UnsignedDoubleInteger` • `UnsignedQuadInteger` • `FixedString` • `VariableString` • `Byte` • `Boolean` • `BLOB` • `Time` • `Undefined` The data type returned in this column is the data type that you defined in Historian Administrator application.
`Tag-ID``.Fixed-String-Length`	VT_UI1	This value is `0` unless the data type is `FixedString`. If the data type is `FixedString`, this number represents the maximum length of the string value.
`Tag-ID``.Col-lector-Name`	VT_BSTR	The name of the collector responsible for collecting data for the specified tag ID.
`Tag-ID``.Source-Address`	VT_BSTR	The address used to identify the specified tag ID at the data source. For iFIX systems, this is the NTF (`Node.Tag.Field`).
`Tag-ID``.Col-lection-Type`	VT_BSTR	Type of collection used to acquire data for the tag: • **Unsolicited:** The collector accepts data from the source whenever the source presents the data • **Polled:** The collector acquires data from a source on a periodic schedule determined by the collector.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
		Note: Not all collectors support unsolicited collection.
`Tag-ID.Col-lection-Interval`	`VT_I4`	The time interval, in milliseconds, between readings of data from this tag. For polled collection, this field represents the time between samples. For unsolicited collection, this fie represents the minimum time allowed between samples.
`Tag-ID.Col-lection-Offset`	`VT_I4`	The time shift from midnight, in milliseconds, for collection of data from this tag.
`Tag-ID.Load-Balanc-ing`	`VT_BOOL`	Indicates whether the data collector should automatically shift the phase of sampling to distribute the activity of the processor evenly over the polling cycle for the specified tag ID. This is sometimes called phase shifting.
`Tag-ID.Time-Stamp-Type`	`VT_BSTR`	The timestamp type applied to data samples at collection time: • `Source`: The source delivers the timestamp along with the data sample. • `Collector`: The collector delivers the timestamp along with the collected data.
`Tag-ID.Hi-Engi-neering-Units`	`VT_R8`	The high end of the engineering units range. Used only for scaled data types and input scaled tags.
`Tag-ID.Lo-Engi-neering-Units`	`VT_R8`	The low end of the engineering units range. Used only for scaled data types and input scaled tags.
`Tag-ID.In-`	`VT_BOOL`	Indicates whether the measurement should be converted to an engineering units value. When set to `False`, the measurement is interpreted as a raw measurement.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
`putScaling`		When set to `True`, the system converts the value to engineering units by scaling the value between the `HiScale` and `LoScale` values. If not enabled, the system assumes the measurement is already converted into engineering units.
`TagID``.HiScale`	`VT_R8`	The high-end value of the input scaling range used for the tag.
`TagID``.LoScale`	`VT_R8`	The low-end value of the input scaling range used for the tag.
`TagID``.CollectorCompression`	`VT_BOOL`	Indicates whether collector compression is enabled for the specified tag ID. Collector compression applies a smoothing filter to incoming data by ignoring incremental changes in values that fall within a deadband centered around the last collected value. The collector passes (to the archiver) any new value that falls outside the deadband and then centers the deadband around the new value.
`TagID``.CollectorDeadbandPercentRange`	`VT_R4`	The current value of the compression deadband.
`TagID``.ArchiveCompression`	`VT_BOOL`	Indicates whether archive collector compression is enabled for the tag.
`TagID``.ArchiveDeadband-`	`VT_R4`	The current value of the archive compression deadband.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
`Percent-Range`		
`Tag-ID``.Col-lector-General1`	`VT_BSTR`	The general (or spare) configuration fields for the specified tag ID.
`Tag-ID``.Col-lector-General2`	`VT_BSTR`	The general (or spare) configuration fields for the specified tag ID.
`Tag-ID``.Col-lector-General3`	`VT_BSTR`	The general (or spare) configuration fields for the specified tag ID.
`Tag-ID``.Col-lector-General4`	`VT_BSTR`	The general (or spare) configuration fields for the specified tag ID.
`Tag-ID``.Col-lector-General5`	`VT_BSTR`	The general (or spare) configuration fields for the specified tag ID.
`Tag-ID``.Read-Securi-tyGroup`	`VT_BSTR`	The name of the Windows security group that controls the reading of data for the specified tag ID. Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for definitions o the various security levels and groups.
`Tag-ID``.Write-Securi-tyGroup`	`VT_BSTR`	The name of the Windows security group that controls the writing of data for the specified tag ID. Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for definitions o the various security levels and groups.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
TagID.AdministratorSecurityGroup	VT_BSTR	The name of the Windows security group responsible for controlling configuration changes for the specified tag ID.
TagID.Calculation	VT_BSTR	The equation for the calculation performed for the specified tag ID.
TagID.LastModified	VT_DBTimeStamp	The date and time that the tag configuration was last modified. The time structure includes milliseconds.
TagID.LastModifiedUser	VT_BSTR	The username of the Windows user who last modified the tag configuration.
TagID.CollectorType	VT_BSTR	The type of collector responsible for collecting data for the specified tag ID: • Undefined • iFIX • Simulation • OPC • File • iFIXLabData • ManualEntry • Simulation • Other
TagID.StoreMilliseconds	VT_BOOL	Indicates whether time resolution in milliseconds is enabled for the specified tag ID. If not enabled, time resolution is in seconds instead of milliseconds. Maximum data compression is achieved when this value is set to False. This is the optimum setting for most applications.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
Tag-ID.UTCBias	VT_I4	The time zone bias for the specified tag ID. Time zone bias is used to indicate the natural time zone of the tag expressed as an offset from UTC (Universal Time Coordinated) in minutes. UTC is the international time standard, the current term for what was commonly referred to as Greenwich Mean Time (GMT).
Tag-ID.AverageCollectionTime	VT_I4	The average time it takes to execute the calculation tag since you started the Calculation collector for the specified tag ID.
Tag-ID.CollectionDisabled	VT_I4	Indicates whether collection is enabled (0) or disabled (1) for the specified tag ID. The default setting is enabled (0).
Tag-ID.CollectorCompressionTimeout	VT_I4	Indicates the maximum amount of time the collector will wait between sending samples to the archive This time is kept per tag, as different tags report to the archiver at different times. This value should be in increments of your collection interval, and not less. Ideally, this field is used for polled data values. It can be used with unsolicited data, but when you do s you are dependent on the data source for the value to change. With unsolicited data, since Historian o ly records the value when it changes, the actual time before the timeout might exceed the compressio timeout.
Tag-ID.ArchiveCompressionTimeout	VT_I4	Indicates the maximum amount of time from the last stored point before another point is stored, if the value does not exceed the archive compression deadband for the specified tag ID.
Tag-ID.InterfaceAbsolut-	VT_BOOL	Indicates whether absolute collector deadband is enabled for the specified tag ID.

Table 56. IhTrend Table (continued)

Column Name	Data Type	Description
eDead-banding		
Tag-ID.In-terface-Absolut-eDead-band	VT_R8	Indicates the value for absolute collector deadband.
Tag-ID.Archive-Absolut-eDead-banding	VT_BOOL	Indicates whether absolute archive deadband is enabled for the specified tag ID.
Tag-ID.Archive-Absolut-eDead-band	VT_R8	Indicates the value for absolute archive deadband.
Tag-ID.Spike-Logic	VT_BOOL	Indicates whether Spike Logic is enabled on the collector.
Tag-ID.Spike-LogicOv-erride	VT_BOOL	Indicates whether the Spike Logic setting for the specified tag ID overrides the collector setting (`True`) the collector setting is used (`False`).

Use care when building queries against the `ihTrend` table. Because a query to this table compares multiple tags at the same time, it takes longer to query the `ihTrend` table than it does the `ihRawData` table. The `ihTrend` table can be quite large, so be sure to either use the default start and end times, or define a specific time interval. See Query Performance Optimization *(on page 270)* for more ideas on how to optimize your query of the `ihTrend` table.

ihTrend Examples

Example SQL statements for the `ihTrend` table are outlined in the following examples.

Example 1: Retrieve Value and Quality of the First 50 Tags

```
SELECT timestamp, *.value, *.quality FROM ihtrend
```

Example 2: Retrieve Value of the First 100 Tags

```
SELECT timestamp, *.value FROM ihTrend
```

Example 3: Retrieve Values of All Tags That Match a Specific Pattern

```
SELECT timestamp,*0001.value FROM ihtrend ORDER BY MY_SERVER.Simulation00001.Value
```

Example 4: Retrieve Hourly Interpolated Values of TagNames That Match *0001

```
SET samplingmode=interp, intervalmilliseconds=1h

SELECT timestamp, *0001.value FROM ihtrend

ORDER BY Simulation00001.value DESC, timestamp DESC
```

Example 5: Retrieve Maximum Values of All TagNames That Match *0001

The following example shows how to use a `TagName` (`simulation.00001.Value`) in a `WHERE` clause.

```
SELECT timestamp, *0001.value FROM ihtrend

WHERE timestamp>='28-nov-2001 00:00'

AND timestamp<='29-nov-2001 00:00:00'

AND samplingmode=calc

AND intervalmilliseconds=1h

AND calculationmode=max

AND simulation00001.Value > 1000 ORDER BY timestamp
```

Example 6: Select Interpolated Values for All Single Float Tags

The following example shows how to select interpolated values for all single float tags, without doing a `JOIN` with the `ihTags` table to retrieve the `DataType` property.

```
SELECT timestamp, *.value,*.description FROM ihtrend

WHERE timestamp>>='28-nov-2001 00:00'

AND timestamp<='29-nov-2001 00:00:00'

AND samplingmode=calculated
```

```
AND intervalmilliseconds=2h

AND *.datatype = singlefloat ORDER BY timestamp
```

Example 7: Select Interpolated Data for TagNames That Match sim*

The following example shows how to sort the returned rows by a `TagName` , `simulation.00001.Value`.

```
SET starttime='28-nov-2001 00:00', endtime='29-nov-2001 00:00:00', samplingmode=interp, intervalmilliseconds=1h

SELECT timestamp, sim*.*, sim*.description, sim*.lastmodifieduser FROM ihtrend

WHERE sim*.description LIKE '*sim*'

AND sim*.description like '*first*'

AND *.datatype = singlefloat

ORDER BY simulation00001.value DESC, timestamp
```

ihQuerySettings Table

The `ihQuerySettings` table contains the current session settings. These settings are applied to all queries you make in a session, unless overridden with a `WHERE` clause. This table displays settings stored in the provider, and has nothing to do with the data stored in the archives.

The `ihQuerySettings` table provides a convenient way to display all your session settings. You cannot, however, write or update settings in this table. This table contains only one row with the settings for the current session. The only way to change these parameters is by using the `SET` statement.

The following table describes the columns of the `ihQuerySettings` table.

Table 57. ihQuerySettings Table

Column Name	Data Type	Description
`StartTime`	`VT_DBTimeStamp`	The start time of the query. This represents the earliest timestamp for any tag contained in the query. If no `StartTime` value is specified, the start time is two hours prior to execution of the query.
`EndTime`	`VT_DBTime-`	The end time of the query. This represents the latest timestamp for any tag contained in the query. If no `EndTime` value is specified, the end time is the time that you execute the query.

Table 57. ihQuerySettings Table (continued)

Column Name	Data Type	Description
	`S-tamp`	
`Sampling-Mode`	`VT_-BSTR`	The mode of sampling data from the archive: • `CurrentValue`: Retrieves the current value. Time frame criteria are ignored. • `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and time frame criteria. • `RawByTime`: Retrieves raw archive values based on time frame criteria. • `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria. **Note:** `EndTime` criteria are ignored for this mode. • `RawByFilterToggle`: Returns filtered time ranges. The values returned are `0` and `1`. If the value is `1`, then the condition is true and `0` means false. This mode is used with the time range and `FilterTag` conditions. Results start and end with timestamps. • `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria. • `Lab`: Returns actual collected values without interpolation. • `Trend`: Returns the raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian ignores any leftover values at the end instead of putting them into a smaller interval. • `Trend2`: Returns the raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian puts leftover values into a remainder interval. This mode is more suitable than the `Trend` mode for analysis of minimums and maximums and for plotting programs that can handle unevenly spaced data. • `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples is less than the number of available samples.

Table 57. ihQuerySettings Table (continued)

Column Name	Data Type	Description
		• `TrendtoRaw`: This mode almost always produces the same results as the `Trend` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend` when the number of actual data samples is less than the requested number of samples. • `TrendtoRaw2`: This mode almost always produces the same results as the `Trend2` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend2` when the number of actual data samples is less than the requested number of samples. • `LabtoRaw`: Provides raw data for the selected calculated data when the number of requested samples is less than the number of available samples. `Calculated` is the default setting.
`Direction`	`VT_BSTR`	The direction (`Forward` or `Backward` from the start time) of data sampling from the archive. The default value is `Forward`.
`NumberOfSamples`	`VT_I4`	Number of samples to retrieve from the archive. Samples will be evenly spaced within the specified start and end times defined for most sampling modes. For the `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawByTime` mode, this column is ignored. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used.
`IntervalMil-`	`VT_I4`	For non-raw sampled data, this column represents a positive integer for the time interval (in milliseconds) between returned samples.

Table 57. ihQuerySettings Table (continued)

Column Name	Data Type	Description
lisec-onds		**Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `IntervalMilliseconds` is used, `NumberofSamples` is not used.
`CalculationMode`	`VT_BSTR`	This column applies only if the `SamplingMode` is set to `Calculated`. It represents the type of calculation to perform on archive data: • `Average` • `Count` • `Maximum` • `MaximumTime` • `Minimum` • `MinimumTime` • `OPCQOr` and `OPCQAnd` • `StandardDeviation` • `StateCount` • `StateTime` • `Total` • `RawAverage` • `RawStandardDeviation` • `RawTotal` • `TimeGood` • `FirstRawValue` • `FirstRawTime` • `LastRawValue` • `LastRawTime` • `TagStats` The default value is `Average`.
`FilterTag`	`VT_BSTR`	Tagname used to define the filter, if specified. Only a single tag can be specified. Wildcards are not supported.

Table 57. ihQuerySettings Table (continued)

Column Name	Data Type	Description
`FilterMode`	`VT_BSTR`	The type of time filter: • `ExactTime`: Retrieves data for the exact times that the filter condition is `True`. • `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. • `AfterTime`: Retrieves data from the time of the last `True` filter condition to the time of the next `False` condition. • `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `False` condition. This value defines how time periods before and after transitions in the filter condition should be handled. For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and ending at the timestamp of the archive value that triggered the `False` condition.
`FilterComparisonMode`	`VT_BSTR`	The type of comparison to be made on the filter comparison value: • `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value. • `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value. • `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value. • `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value. • `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value. • `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value. • `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value. • `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value. • `AllBitsSet`: Filter condition is `True` when the binary `FilterTag` value is equal to all the bits in the condition. It is represented as `^` to be used in `FilterExpression`.

Table 57. ihQuerySettings Table (continued)

Column Name	Data Type	Description
		• `AnyBitSet`: Filter condition is `True` when the binary `FilterTag` value is equal to any of the bits in the condition. It is represented as `~` to be used in `FilterExpression`. • `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is represented as `!~` to be used in `FilterExpression`. • `AllBitsNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to all the bits in the condition. It is represented as `!^` to be used in `FilterExpression`. This option defines how archive values for the `FilterTag` value should be compared to the `FilterValue` value to establish the state of the filter condition. If `FilterTag` and `FilterComparisonValue` values are specified, time periods are filtered from the results where the filter condition is `False`.
`FilterValue`	`VT_BSTR`	The value with which to compare the `FilterTag` value to determine appropriate filter times.
`FilterExpression`	`VT_BSTR`	An expression which includes one or more filter conditions. The type of conditions used are: • `AND` • `OR` • Combination of `AND` and `OR` This column can be used instead of the `FilterTag`, `FilterComparisonMode`, and `FilterValue` columns. While using `FilterExpression`, the expression is passed within single quotes. For complex expressions, you write the conditions within parentheses. There is no maximum length for this value, but if called using OLE DB or Excel, these tools may have their own limitations.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT)
`DaylightSav-`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.

Table 57. ihQuerySettings Table (continued)

Column Name	Data Type	Description
`ing-Time`		
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. If the query result contains more rows than the `RowCount` value, the Historian OLE DB provider truncates the extra rows. The truncation is performed last. For instance, if you use `ORDER BY` in your `SELECT` statement, the truncation occurs after the rows are ordered.
`AlarmType`	`VT_BSTR`	Indicates the alarm type: • `Alarms` • `Alarm_History` • `Events`

ihQuerySettings Examples

Example SQL statements for the `ihQuerySettings` table are outlined in the following examples.

Example 1: Show All Settings for the Current Session

```
SELECT * FROM ihquerysettings
```

Example 2: Show the Selected Session Settings

```
SELECT starttime, endtime FROM ihquerysettings
```

ihCalculationDependencies Table

The `ihCalculationDependencies` table contains the calculation and server-to-server tags and their triggers. The following table describes the columns of the `ihCalculationDependencies` table.

Table 58. ihCalculationDependencies Table

Column Name	Data Type	Description
`Tagname`	`VT_BSTR`	A calculation or server-to-server tag with unsolicited collection and at least one dependent tag.
`DependentTagname`	`VT_BSTR`	A dependent tagname. If a tag has multiple dependent tags, there are multiple rows in the table for that tagname.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.

ihCalculationDependencies Examples

Example SQL statements for the `ihCalculationDependencies` table are outlined in the following examples.

Example 1: Show the Dependencies for a Specific Tag

```
SELECT * FROM ihcalculationdependencies WHERE tagname = c1
```

Example 2: Show the Dependencies for a Specific Dependent Tag

```
SELECT * FROM ihcalculationdependencies

WHERE dependenttagname=brahms.ai1.f_cv
```

ihAlarms Table

The `ihAlarms` table contains collected alarms and events data. The following table describes the columns of the `ihAlarms` table.

CAUTION:
When you perform joins of the `ihRawData` and `ihAlarms` tables, you can easily construct queries that temporarily consume all your system resources. Although this scenario typically does not affect data collection, it can interfere with data analysis. To avoid this issue, always define a start and end time for the query to limit the number of rows returned.

Table 59. ihAlarms Table

Column Name	Data Type	Description
`AlarmID`	`VT_I4`	The unique ID of the alarm or event in the Historian alarm database.
`ItemID`	`VT_BSTR`	The OPC `ItemID` of the alarm. This contains the source address of the data access tag with which th alarm is associated. This can contain a `NULL` value if an alarm is not associated with a tag.
`Source`	`VT_BSTR`	The unique identifier used by the OPC AE Collector for the alarm or event.
`DataSource`	`VT_BSTR`	The collector interface name associated with the alarm or event.
`Tagname`	`VT_BSTR`	The Historian tag name associated with the alarm. This value is `NULL` unless the tag is also collected Historian.
`AlarmType`	`VT_BSTR`	The alarm type: • `Alarms`: In Historian, the full life cycle of an alarm is stored as a single record in the alarm arch • `Alarm_History`: The separate transitions for all alarms. One row per transition is returned. • `Events`: The simple and tracking events.
`EventCategory`	`VT_BSTR`	The OPC event category of the alarm or event.
`Condition`	`VT_BSTR`	The OPC condition of the alarm. Does not apply to event data. This value combined with the `Source` v ue comprises an alarm.
`SubCondition`	`VT_BSTR`	The OPC subcondition of the alarm. Does not apply to event data. This value represents the state of alarm.
`StartTime`	`VT_DBTimeStamp`	The start time or timestamp of the alarm or event.
`EndTime`	`VT_DBTimeStamp`	The end time of the alarm. Does not apply to event data.
`AckTime`	`VT_DBTimeStamp`	The time the alarm was acknowledged. Does not apply to event data.
`Microseconds`	`VT_I4`	The microsecond portion of the date and time.

Table 59. ihAlarms Table (continued)

Column Name	Data Type	Description
`Message`	`VT_BSTR`	The message attached to the alarm or event.
`Acked`	`VT_BOOL`	Stores the acknowledgement status of the alarm. If the alarm is acknowledged, this is set to `TRUE`.
`Severity`	`VT_I4`	The severity of the alarm or event. Stored as an integer value with a range of 1–1000.
`Actor`	`VT_BSTR`	The operator who acknowledged the alarm, or caused the tracking event.
`Quality`	`VT_-VARIANT`	The quality of the alarm or event. Stored as a string, with values of `GOOD` or `BAD`.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT)
`Daylight-SavingTime`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`RowCount`	`VT_I4`	The maximum number of rows returned by the current query.
`User-Defined Variable #X`	`VT_-VARIANT`	User-defined variables. This is a dynamic list of columns that varies based on the collectors running the Historian system.

Note:
Additional fields may be added by third-party products such as iFIX. Please consult the relevant product documentation for further information.

ihAlarms Examples

Example 1: Show All Alarms for the Last Two Hours, Including Vendor Attributes

```
SELECT * FROM ihAlarms

SELECT * FROM ihAlarms WHERE alarmtype = alarms //same as above
```

Example 2: Show Alarm History

```
SELECT * FROM ihAlarms WHERE alarmtype = alarm_history
```

Example 3: Show Tracking and System Events

```
SELECT * FROM ihAlarms WHERE alarmtype = events
```

Example 4: Return All Closed Events and Associated Tag Data

```
SELECT

alarmid, ihalarms.tagname, ihalarms.starttime, ihalarms.endTime, ihrawdata.timestamp, ihrawdata.value

FROM ihalarms, ihrawdata

WHERE ihalarms.tagname=ihrawdata.tagname

AND ihalarms.starttime <= ihrawdata.timestamp

AND ihalarms.endtime >= ihRawdata.timestamp

AND ihalarms.subcondition == "OK"

OR ihalarms.quality = "Bad"

ORDER BY ihalarms.starttime
```

> **Note:**
> When you join data from the `ihRawData` and `ihAlarms` tables, be sure to specify a timestamp range.

Example 5: Return All Open Alarms and Associated Tag Data

```
SELECT

alarmid, ihalarms.tagname, ihalarms.starttime, ihalarms.endTime, ihrawdata.timestamp, ihrawdata.value

FROM ihalarms, ihrawdata

WHERE ihalarms.tagname=ihrawdata.tagname

AND ihalarms.starttime <= ihrawdata.timestamp

AND ihalarms.endtime >= ihRawdata.timestamp

AND ihalarms.subcondition <> "OK"

AND ihalarms.quality = "Good"

ORDER BY ihalarms.starttime
```

> **Note:**
> When you join data from the `ihRawData` and `ihAlarms` tables, be sure to specify a timestamp range.

ihEnumeratedSets Table

The `ihEnumeratedSets` table contains information about enumerated sets that are defined in the system. The following table describes the columns of the `ihEnumeratedSets` table.

Table 60. ihEnumeratedSets Table

Column Name	Data Type	Description
`SetName`	`VT_BSTR`	The name of the set.
`Description`	`VT_BSTR`	The description of the set.
`NumberofStates`	`VT_I4`	The number of states a set contains.
`NumberofTagReferences`	`VT_I4`	The number of tags with which a set is associated.
`SetDataType`	`VT_BSTR`	The data type of the set.
`AdministratorSecurityGroup`	`VT_BSTR`	The security group to which the set belongs.

Table 60. ihEnumeratedSets Table (continued)

Column Name	Data Type	Description
`LastModifiedUser`	`VT_BSTR`	Indicates which user last modified the set.
`LastModifiedTime`	`VT_DBTimeStamp`	Indicates the last time the set was modified.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.

ihEnumeratedSets Examples

Sample SQL statements for the `ihEnumeratedSets` table are outlined in the following examples.

Example 1: Retrieve All Sets By Using Integer States

```
SELECT * FROM ihEnumeratedSets

WHERE SetDataType=’integer’
```

Example 2: Retrieve a Set By Name From Sets

```
SELECT * FROM ihEnumeratedSets

WHERE setname like PLC1
```

ihEnumeratedStates Table

The `ihEnumeratedStates` table contains information about enumerated sets that are defined in the system. The following table describes the columns of the `ihEnumeratedStates` table.

Table 61. ihEnumeratedStates Table

Column Name	Data Type	Description
`SetName`	`VT_BSTR`	The name of the set.
`Description`	`VT_BSTR`	The description of the set.
`NumberofStates`	`VT_I4`	The number of states a set contains.
`NumberofTagReferences`	`VT_I4`	The number of tags with which a set is associated.
`SetDataType`	`VT_BSTR`	The data type of the set.
`AdministratorSecurityGroup`	`VT_BSTR`	The security group to which the set belongs.
`LastModifiedUser`	`VT_BSTR`	Indicates which user last modified the set.

Table 61. ihEnumeratedStates Table (continued)

Column Name	Data Type	Description
`LastModifiedTime`	`VT_DBTimeStamp`	Indicates the last time the set was modified.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.

ihEnumeratedStates Examples

Sample SQL statements for the `ihEnumeratedStates` table are outlined in the following examples.

Example 1: Retrieve All States That Belong to a Specific Set

```
SELECT * FROM ihEnumeratedStates

WHERE setname=plcset1 order by statelowvalue ascending
```

Example 2: Retrieve All States From a Specific Set

```
SELECT * FROM ihEnumeratedStates

WHERE setname = 'setname'
```

ihUserDefinedTypes Table

The `ihUserDefinedTypes` table contains information about user-defined data types in the system.

Use this table to see the set of types and get information about each field in the data type.

The following table describes the columns of the `ihUserDefinedTypes` table.

Table 62. ihUserDefinedTypes Table

Column Name	Data Type	Description
`TypeName`	`VT_BSTR`	The name of the user-defined type.
`DataType`	`VT_BSTR`	The data type of the user-defined type.
`Description`	`VT_BSTR`	The description of the user-defined type.
`StoreFieldQuality`	`VT_BOOL`	Indicates whether the field-level quality is stored.
`NumberofFields`	`VT_I4`	The number of fields a user-defined type contains.
`NumberofTagReferences`	`VT_I4`	The number of tags with which a user-defined type is associated.
`AdministratorSecurityGroup`	`VT_BSTR`	The security group to which the user-defined type belongs.
`LastModifiedUser`	`VT_BSTR`	Indicates which user last modified the user-defined type.
`LastModifiedTime`	`VT_DBTimeStamp`	Indicates the last time the user-defined type was modified.

Table 62. ihUserDefinedTypes Table (continued)

Column Name	Data Type	Description
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.

ihUserDefinedTypes Examples

Sample SQL statements for the `ihUserDefinedType` table are outlined in the following examples.

Example 1: Retrieve All User-Defined Types

```
SELECT * FROM ihuserdefinedtypes
```

Example 2: Retrieve a User-Defined Type By Name

```
SELECT * FROM ihuserdefinedtypes WHERE typename LIKE New
```

ihFields Table

The `ihFields` table contains information about field elements that are specified in user-defined data types. The following table describes the columns of the `ihFields` table.

Table 63. ihFields Table

Column Name	Data Type	Description
`TypeName`	`VT_BSTR`	The name of the user-defined type.
`FieldName`	`VT_BSTR`	The name of the field.
`Description`	`VT_BSTR`	The description of the field.
`FieldValue-`	`VT_BSTR`	The data type of the field.

Table 63. ihFields Table (continued)

Column Name	Data Type	Description
`DataType`		
`MasterField`	`VT_BOOL`	Indicates whether the field is a master field.
`RowCount`	`VT_I4`	Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.

ihFields Examples

Sample SQL statements for the `ihFields` table are outlined in the following examples.

Example: Retrieve All Fields for a Specific Type

```
SELECT * FROM ihfields WHERE typename='MyUserDefinedType'
```

Chapter 9. Using Historian Administrator

Historian Administrator

Introduction to Historian Administrator

Historian Administrator is a Windows-based application, which allows you to access administrative functions. Using Historian Administrator, you can monitor, supervise, archive, retrieve, and control data gathering functions from the server, a client, or one or more remote non-web-based nodes.

> **Note:**
> You can install multiple instances of Historian Administrator. Changes that you make to parameters on one instance are not automatically updated in other instances.

Historian Administrator communicates with the Historian server using the Historian API. You can install Historian Administrator on a local or a remote machine that has a TCP/IP connection to the Historian server.

Intended Audience

This guide is intended for people who need to:

- Retrieve and analyze archived information.
- Set up and maintain configuration and other parameters for tags, collectors, and archives.
- Perform specific supervisory and security tasks for Historian.
- Maintain and troubleshoot Historian.

About Historian Administrator

Using Historian Administrator, you can:

- Examine key operating statistics for archives and collectors.
- Perform archive maintenance, including:
    - Setting archive size.
    - Selecting options and parameters.
    - Accessing security parameters.
    - Adding archives.
- Perform tag maintenance, including:

    - Adding, deleting, and copying tags.
    - Searching for tags in a data source or in the Historian database.
    - Starting and stopping data collection for a tag.
    - Configuring, displaying, and editing tag parameters and options.
    - Displaying trend data for selected tags.
- Perform collector maintenance, including:
    - Adding or deleting collectors.
    - Configuring, displaying, and editing parameters for all types of collectors.
    - Displaying performance trends for selected collectors.

> **Note:**
> You can back up and restore archives directly using Azure File System (AFS). If you try to back up archives using Historian Administrator, and error occurs.

Limitations

If the number of archives is large (that is, more than 5,000), Historian Administrator takes a long time to start.

Access Historian Administrator

- Install Historian Administrator Using the Installer *(on page 34)*.
- Create a Windows user on the Historian server *(on page 40)*.
- Use a page with a resolution of 1024 x 768 or above.

From the Start menu, select **Historian Administrator**.

> **Note:**
> By default, The system attempts to connect to the default server using the username and password of the currently logged-in user. If you want to use a different server or user account:
>
> a. Select **Main**.
>
>   A login window appears.
>
> b. Provide the server name, username, password, and domain information, and then select **OK**.

The **Proficy Historian Administrator** window appears, displaying the following pages.

- **System Statistics**: Contains system status indicators, data collector performance indicators, collector maintenance, tag maintenance, and help pages.
- **Tag Maintenance**: Contains tag names, parameters, and controls.
- **Collector Maintenance**: Contains collector names, parameters, and controls.
- **Data Store Maintenance**: Contains archive names, parameters, alarms, security, and controls.
- **Message Search**: Not applicable

Installing Historian Administrator

Install Historian Administrator Using the Installer

If you already have Historian Administrator on your machine (installed using on-premises Proficy Historian), you can just change the destination to the Azure Load Balancer IP, and begin using it:

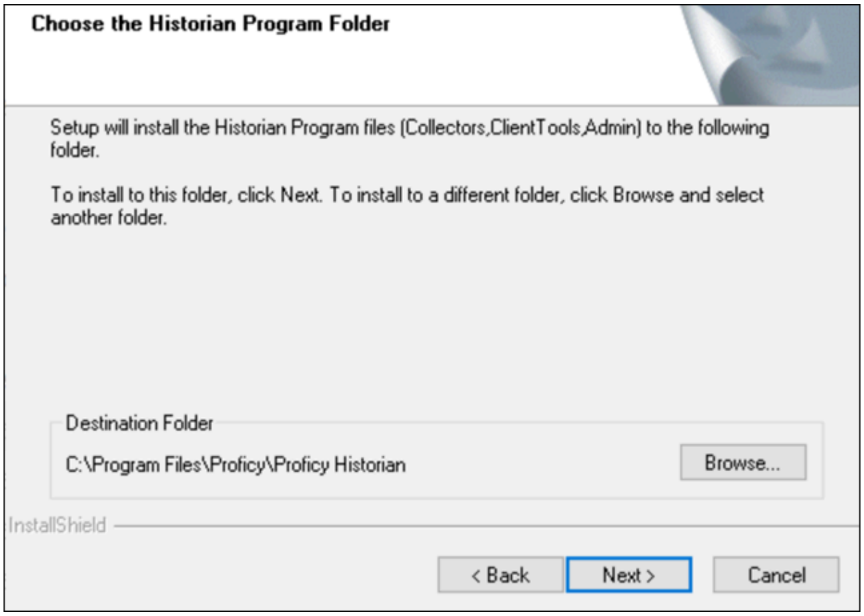
1. Select **Main**.

   A login window appears.

2. Provide the Azure Load Balancer IP, username, password, and domain information, and then select **OK**.

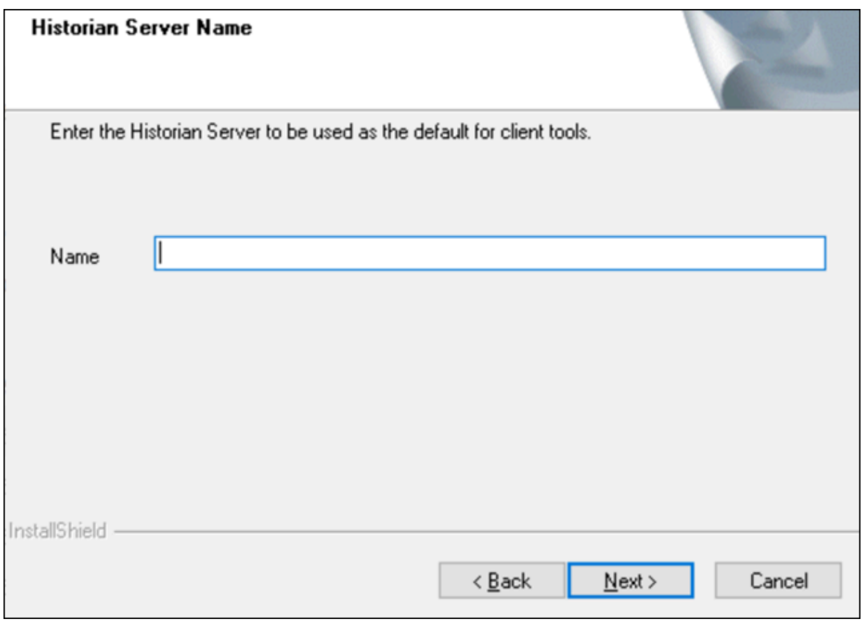
This topic describes how to install Historian Administrator using the installer. You can also install it at a command prompt *(on page 36)*.

1. Run the `InstallLauncher.exe` file. Contact the support team for this installer.
2. Select **Install Client Tools**.
   The **Select Features** page appears, displaying a list of components.
3. Select the **Historian Administrator** check box.
4. Select **Next**.
   The **Choose the Historian Program Folder** page appears.



5. As needed, change the destination folder of Historian Administrator, or leave the default folder, and then select **Next**.

   The **Historian Server Name** page appears.



6. Enter the Azure Load Balancer IP of Proficy Historian for Azure Cloud that you want to use with Historian Administrator, and then select **Next**.

> **Tip:**
> To find the Azure Load Balancer IP:

a. Go to the Azure portal.
b. Go to the **Resource Group** that was specified during deployment.
c. Select the *cluster_name-IP* to access the resource of type **Public IP Address**.
d. Select or copy the IP Address.

7. When you are asked to reboot your system, select **Yes**.

Install Historian Administrator at a Command Prompt

Install Historian Administrator using the installer *(on page 34)* on a machine. When you do so, a template file named `setup.iss` is created at `C:\Windows`. This file stores the installation options that you have provided. You can then use this template to install Historian Administrator at a command prompt on other machines.

1. Copy the `setup.iss` file to the machine on which you want to install Historian Administrator at a command prompt.
2. In the folder in which you have copied the file, run the following command: `setup.exe /s /sms`
   The installer runs through the installation steps.

   Note:
   If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

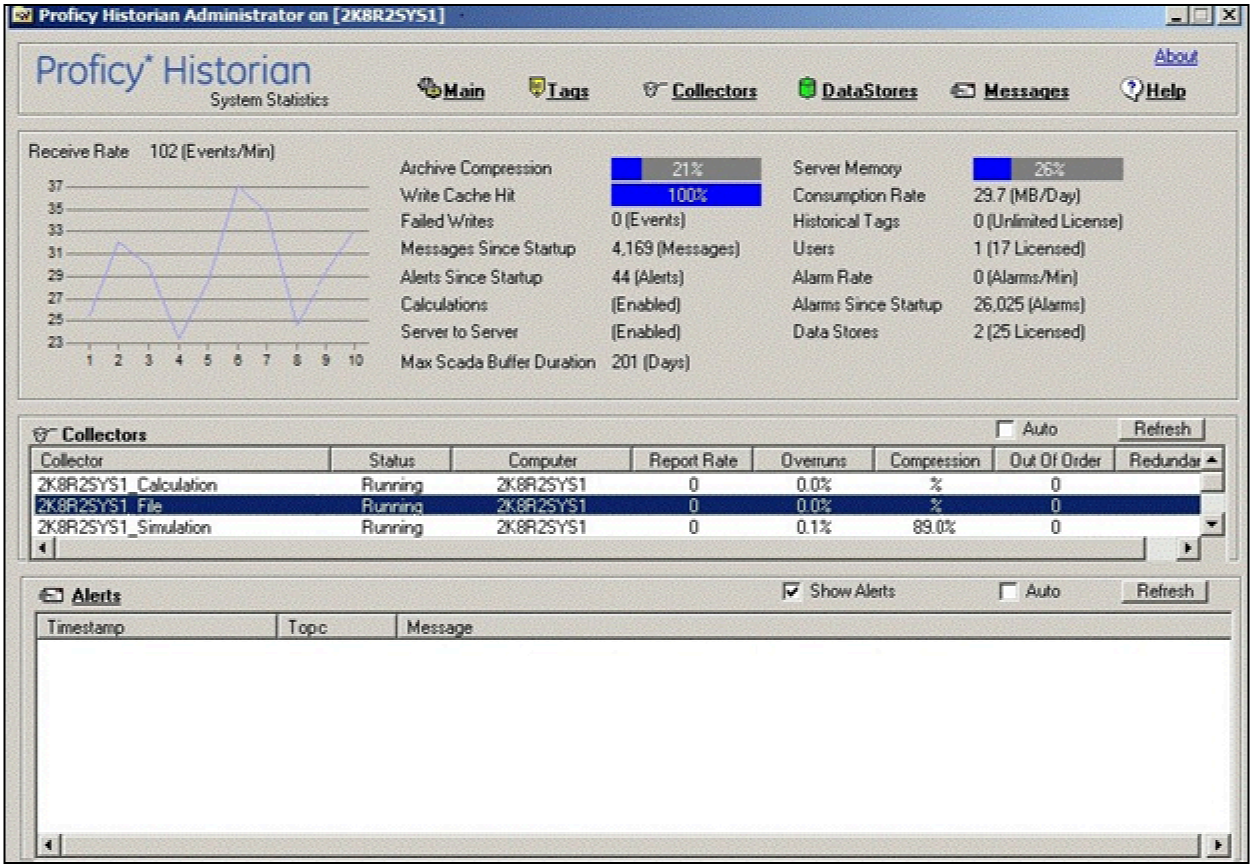
Historian Administrator is installed.

Historian Administrator - Pages

The Main Page

The **Main** page of Historian Administrator displays the system statistics, which contains the current system status and performance statistics. It provides an overall view of the system health. The page has the following sections:

- The **System Statistics** section *(on page 388)*
- The **Collectors** section *(on page 391)*



The **System Statistics** Section

The following table describes the fields in the **System Statistics** section.

> **Note:**
> The statistics displayed in this section are calculated independently on various time scales and schedules. As a result, they may be updated at different times.

Field	Description
Receive Rate (a time-based chart in events/minute)	Not applicable
Archive Compression (% compression)	Displays the current effect of archive data compression. If the value is zero, it indicates that archive compression is either ineffective or turned off. To increase the effect of data compression, increase the value of archive compression deadbands on individual tags in the Tag Maintenance section to activate compression.

Field	Description
	In calculating the effect of archive compression, Historian counts internal system tags as well as data source tags. Therefore, when working with a very small number of tags and with compression disabled on data source tags, this field may indicate a value other than zero. If you use a realistic number of tags, however, system tags will constitute a very small percentage of total tags and will therefore not cause a significant error in calculating the effect of archive compression on the total system.
Write Cache Hit	Displays the hit ratio of the write cache in percentage of total writes. It is a measure of how efficiently the system is collecting data. Typically, this value should range from 95 to 99.99%. If the data is changing rapidly over a wide range, however, the hit percentage drops significantly because current values differ from recently cached values. More regular sampling may increase the hit percentage. Out-of-order data also reduces the hit ratio.
Failed Writes	Displays the number of samples that failed to be written. Since failed writes are a measure of system malfunctions or an indication of offline archive problems, this value should be zero. If you observe a non-zero value, investigate the cause of the problem and take corrective action. Historian also generates a message if a write fails. Note that the message only appears once per tag, for a succession of failed writes associated with that tag. For example, if the number displayed in this field is 20, but they all pertain to one Historian tag, you will only receive one message until that Historian tag is functional again.
Messages Since Startup	Not applicable
Alerts Since Startup	Not applicable
Calculations	Not applicable
Server-to-Server	Displays the value **Enabled** if the Server-to-Server collector is licensed on the software key.
Alarms since Startup	Not applicable

Field	Description
Server Memory	Displays how much of the server memory the data archiver consumes.
Free Space (MB)	Displays how much disk space (in MB) is left in the current archive.
Consumption Rate (MB/day)	Displays how fast the archive disk space is consumed. If the value is too high, you can reduce it by slowing the poll rate on selected tags or data points or by increasing the filtering on the data (widening the compression deadband to increase compression).
Est. Days to Full (Days)	Not applicable
Active Tags	Displays number of tags in your configuration.
Licensed Tags	Displays the number of tags authorized for this Historian installation by the software key and license. **Note:** If this field displays 100 tags and the licensed users field displays 1 client, you are likely running in demonstration mode and may have incorrectly installed your hardware key.
Active Users	Displays the number of users currently accessing the Historian system.
Licensed Users	Displays the number of users authorized to access Historian using the software key and license. The number of users that are authorized to access Historian is strictly based on the software key and license. However, if you have utilized your available Client Access Licenses (CAL) and need an additional one to administer the system in an emergency, you have an option to reserve a CAL. This reserved CAL allows you to access the server. To do so, provide the reserved CAL to the system administrators and add them to the `ih Security Admins` group. A system administrator can then connect to Historian in an emergency. This facility is optional and does not provide a guaranteed connection. It only eliminates the emergency situations when a CAL is preventing you from accessing the system and may not work if there

Field	Description
	are other conditions. For example, if the Historian server is busy, you will not be able to connect using this feature. **Note:** If this field displays 1 client and the Licensed Tags field displays 100 tags, you are likely running in demonstration mode and you may have incorrectly installed your hardware key.
Alarm Rate	Not applicable
SCADA Tags	Displays the number of CIMPLICITY or iFIX tags.
Tags Consumed by Arrays	Not applicable

The **Collectors** Section

The **Collectors** section shows current statistics on the operation of all the connected collectors in the system. In this section, you can:

- Access the **Collector Maintenance** page of a collector by selecting the collector name. You can also access the **Collector Maintenance** page by selecting the collector link at the beginning of the **System Statistics** section.
- Automatically refresh the data every 45 seconds by selecting the **Auto** check box.
- Manually refresh the data by selecting **Refresh**.

The following table describes the fields in the **Collectors** section.

Field	Description
Collector	Displays the collector ID, which is used to identify the collector in Historian.
Status	Displays the current status of collection. This field contains one of the following values:

Field	Description
	• **Running**: Indicates that the collector is running. • **Stopped**: Indicates that the collector is not collecting data. • **Unknown**: Indicates that status information about the collector is unavailable, perhaps as a result of a lost connection between the collector and the server.
Computer	Displays the name of the computer on which the collector is running.
Report Rate	Displays the number of samples per minute that the server is receiving data from the collector. It is a measure of the collection rate and data compression. If the collector compression percent is zero, and if the value in this field is equal to the data acquisition rate, it indicates that every data point received from the collector is being reported to the server. This means that the collector is not performing any data compression. You can lower the report rate, and make the system more efficient, by increasing the data compression at the collector. To do this, widen the collection compression deadbands for selected tags.
Overruns	Not applicable
Compression %	Displays the percentage of how effective compression is at present for the specific collector since collector startup. A value of zero indicates that compression is either turned off or not effective. To increase the value, enable compression on the collector's associated tags and increase the width of the compression deadband on selected tags. The collector keeps track of how many samples it collected from the data source (for example, the OPC server) and keeps track of how many samples it reported to the Data Archiver (after collector compression is complete). A low number or zero means almost everything coming from the data source is being sent to the data archiver. The reason for the low number or zero is that too many samples are exceeding compression or you are not using collector compression. A high number or 100 means you are collecting a lot of samples, but they are not exceeding collector compression and therefore are not being sent to server.

Field	Description
Out of Order	Displays the number of samples within a series of timestamped data values normally transmitted in sequence that have been received out of sequence since collector startup. This field applies to all collectors.
Redundancy	Not applicable

The Data Store Page

Using the **Data Store** page, you can read and modify the parameters of archives, data stores, global options, security, and alarms.

The **Archive Details** Section

In the **Archive Details** section, a list of all the archives in your system appears. To access an archive, select it. In this section, you can:

- Close an archive by selecting **Close Archive**.

This topic describes the fields in each subsection in the **Archive Details** section.

> **Note:**
> You cannot back up and restore data using Historian Administrator. This is because the backup and restore functions are performed using Azure File System (AFS).

The Status Subsection

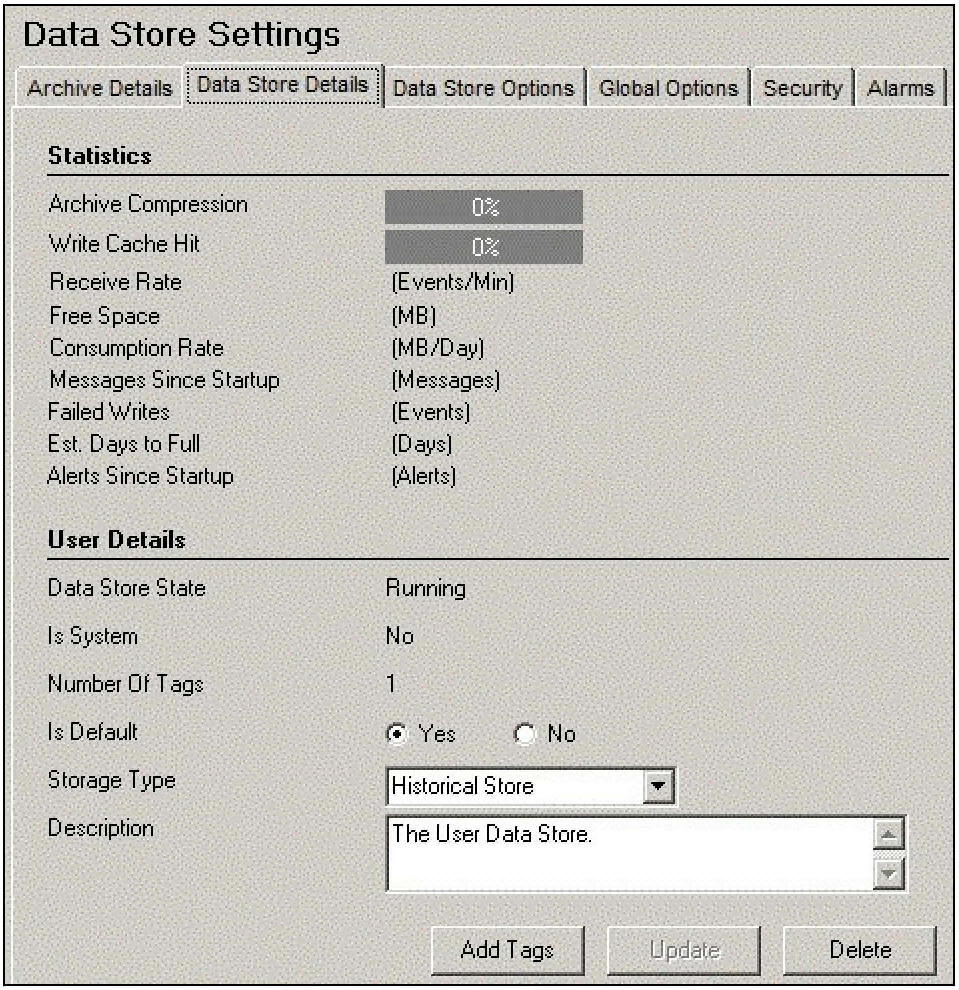
Field	Description
Status	The current operating state of the archive. This field contains one of the following values: • **Current**: Indicates that the archive is actively accepting data. • **Active**: Indicates that the archive contains data but is not currently accepting data. • **Empty**: Indicates that the archive has never accepted data.
Start Time	The time of the oldest sample in the archive.
End Time	The time the archive is closed (automatically or manually).

Table 64. Resources

Field	Description
File Location	The path and name of the archive file.
File Size	The size (in MB) of the archive file. **Note:** Historian supports a maximum archive size of 256 GB per archive.
File Attribute	The attribute to set a closed archive to read-only or read/write. **Note:** To create multiple archives at the same time, set the value of this field to Read/Write.

The **Data Store Details** Section

This topic describes the fields in each subsection in the **Data Store Details** section.



The Statistics Subsection

Field	Description
Archive Compression (% com- pression)	Displays the current effect of archive data compression. If the value is zero, it indicates that archive compression is either ineffective or turned off. To increase the effect of data compression, increase the value of archive compression deadbands on individual tags in the Tag Maintenance section to activate compression. In calculating the effect of archive compression, Historian counts internal system tags as well as data source tags. Therefore, when working with a very small number of tags and with compression dis- abled on data source tags, this field may indicate a value other than zero. If you use a realistic number of tags, however, system tags will constitute a very small percentage of total tags and will therefore

Field	Description
	not cause a significant error in calculating the effect of archive compression on the total system.
Write Cache Hit	Displays the hit ratio of the write cache in percentage of total writes. It is a measure of how efficiently the system is collecting data. Typically, this value should range from 95 to 99.99%. If the data is changing rapidly over a wide range, however, the hit percentage drops significantly because current values differ from recently cached values. More regular sampling may increase the hit percentage. Out-of-order data also reduces the hit ratio.
Receive Rate	Displays how busy the server is at a given instance and the rate at which the server is receiving data from collectors.
Free Space (MB)	Displays how much disk space (in MB) is left in the current archive.
Consumption Rate (MB/day)	Displays how fast the archive disk space is consumed. If the value is too high, you can reduce it by slowing the poll rate on selected tags or data points or by increasing the filtering on the data (widening the compression deadband to increase compression).
Messages Since Startup	Not applicable
Failed Writes	Displays the number of samples that failed to be written. Since failed writes are a measure of system malfunctions or an indication of offline archive problems, this value should be zero. If you observe a non-zero value, investigate the cause of the problem and take corrective action. Historian also generates a message if a write fails. Note that the message only appears once per tag, for a succession of failed writes associated with that tag. For example, if the number displayed in this field is 20, but they all pertain to one Historian tag, you will only receive one message until that Historian tag is functional again.
Est Days to Full (Days)	Displays how much time is left before the archive is full, based on the current consumption rate. This value is dynamically calculated by the server and becomes more accurate as an archive file gets closer to completion. This value is only an estimate and will vary based on a number of factors, including the current compression effectiveness. The System sends messages notifying you at 5, 3, and 1

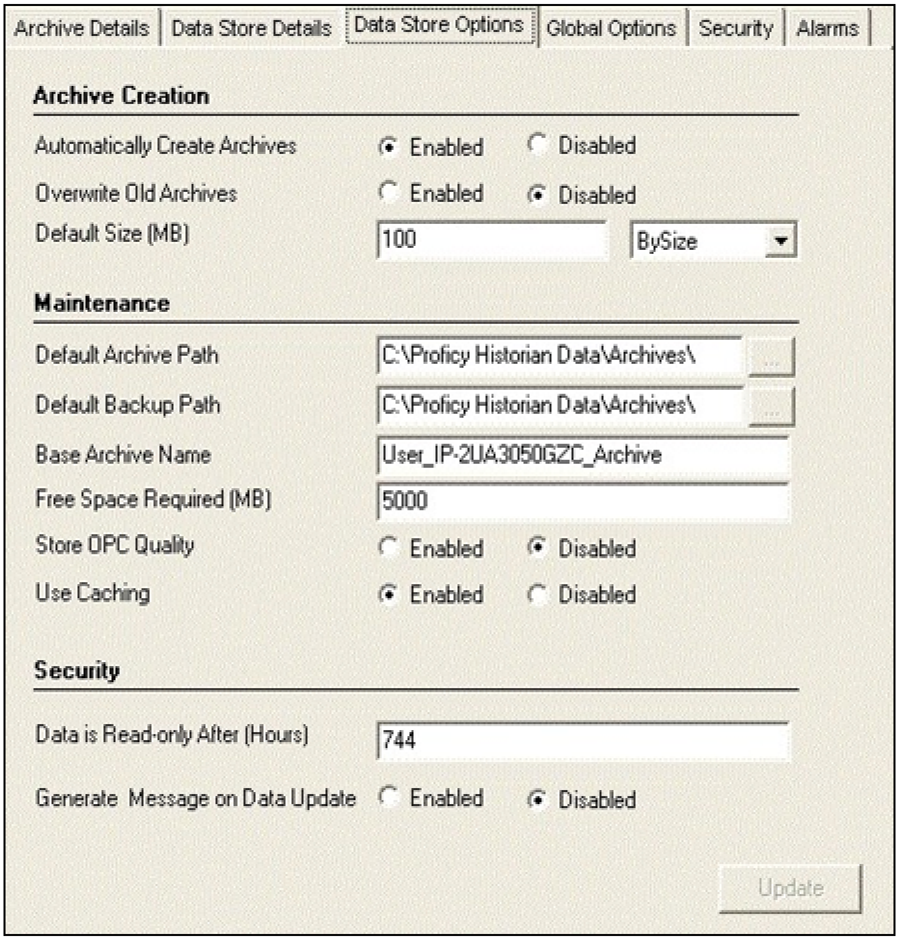
Field	Description
	days until full. After the archive is full, a new archive must be created (could be automatic). To increase this value, you must reduce the consumption rate. To ensure that collection is not interrupted, make sure that the **Automatically Create Archives** option is enabled in the **Data Store Maintenance** section (under **Global Options**). You may also want to enable the **Overwrite Old Archives** option if you have limited disk capacity. Enabling overwrite, however, means that some old data will be lost when new data overwrites the data in the oldest online archive. Use this feature only when necessary.
Alerts Since Startup	Not applicable

The User Settings Subsection

Field	Description
Data Store State	The current state of the data store. The value in this field is **Running** until you delete the data store.
Is System	Indicates whether this data store is the system data store. **Note:** By default, the **Is System** value of the system data store is set to **yes**. You cannot set the **Is System** value of any historical data store to **yes**.
Number of Tags	Displays the number of tags the data store contains.
Is Default (Yes/No)	Indicates whether the data store is the default store. Select **Yes** to set this data store as default one.
Storage Type	Indicates whether the storage type is historical or SCADA buffer.
Description	The description of the data store.

The **Data Store Options** Section

This topic describes the fields in each subsection in the **Data Store Options** section.



The Archive Creation or the SCADA BufferSubsection

The **Archive Creation** subsection appears only if the data store type is historical. The **SCADA Buffer** subsection appears only if the data store type is SCADA buffer.

Field	Description
Automatically Create Archives	Identifies whether the server must automatically create an archive file whenever the current archive file is full. The archive files are created in the default path directory. **Note:** If you plan to create multiple archives at the same time, select the **Disabled** option.
Overwrite Old Archives	When enabled, the system replaces the oldest archived data with new data when the default size has been reached. Since this action deletes historical

Field	Description
	data, exercise caution in using this feature. We recommend that you back up the archive using AFS so that you can restore it later. **Note:** To create multiple archives at the same time, select the **Disabled** option. If both the **Automatically Create Archives** and **Overwrite Old Archives** are enabled, set the ihArchiveFreeSpaceHardLimit parameter to TRUE using the Historian APIs.
Default Size (MB)	The default size of a newly created archive or the duration of a newly created archive in days or hours. Select one of the following options: • **BySize**: A new archive file is created after the current archive reaches the default size. The recommended default archive size is at least 500 MB for systems with 1000 tags or more. • **Days**: A new archive file is created after the number of days that you specify in the **Archive Duration** field that will appear. • **Hours**: A new archive file is created after the number of hours that you specify in the **Archive Duration** field that will appear.
SCADA Buffer Duration (Days)	Indicates the maximum number of days you want to store the trend data. The maximum number of days is 200.
Archive Duration (Days/ Hours)	Indicates the days or hours for which the duration of the archive is set.

The Maintenance Subsection

Field	Description
Default Archive Path	The folder path to store newly created archives. **Note:** We recommend not to use a period in the default archive path field. If you do so, you will not be able to specify a default archive name.
Default Backup Path	Not applicable
Base Archive Name	A prefix that you want to add to all the archive files.

Field	Description
Free Space Required (MB)	Indicates the remaining disk space required after a new archive is created. If the available space is less than the requirement, a new archive is not created. The default value is 5000 MB.
Store OPC Quality	Indicates whether to store the OPC data quality. **Note:** To create multiple archives at the same time, select the **Disabled** option.
Use Caching	Indicates whether caching must be enabled. When reading data from the archiver, some data is saved in the system memory and retrieved using caching. This results in faster retrieval as the data is already stored in the buffer. **Note:** This option is not available for SCADA Buffer data stores.

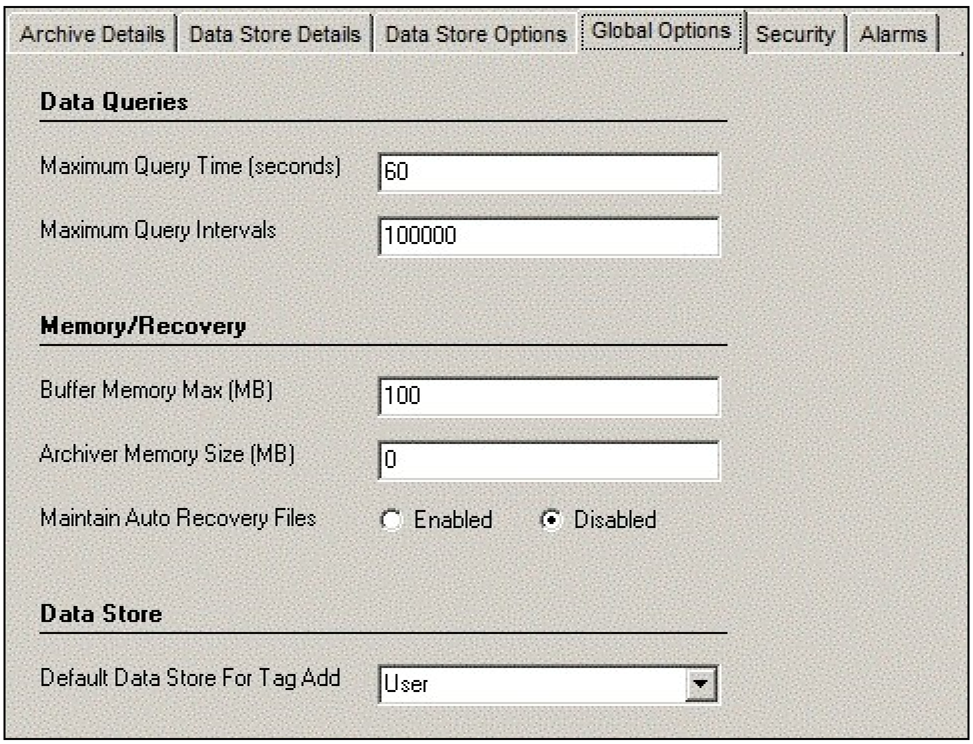
The Security Subsection

Field	Description
Data is Read-only After (Hours)	The number of hours for data to be stored in a read/write archive. After the time lapses, that portion of the archive file is automatically made read-only. Incoming data values with timestamps prior to this time are rejected. A single archive file, therefore, may have a portion made read-only, another portion that is read/write containing recently written data, and another that is unused free space. **Note:** If an archive file is read-only, you cannot move the file in Windows File Explorer. To be able to move a read-only archive file, you must first remove the archive by selecting the file and selecting **Remove** in the **Archive Maintenance** page.

Field	Description
Generate Message on Data Update	Indicates whether an audit log entry will be made any time the value of a previously archived data point is overwritten. This log entry will contain both the original and new values. **Note:** To create multiple archives at the same time, select the **Disabled** option. This option is not available for SCADA Buffer data store.

The **Global Options** Section

This topic describes the fields in each subsection in the **Global Options** section.



The Data Queries Subsection

Field	Description
Maximum Query Time (seconds)	Specifies the maximum time that a data point or query can take before it is terminated. Use this setting to limit query time and provide balanced read access to the archiver. This is applicable to all query types.

Field	Description
Maximum Query Intervals	Specifies the maximum number of samples per tag that Historian can return from a non-raw data query. Use this setting to throttle query results for non-raw data queries. This setting is not applicable to filtered data queries or raw data queries. If the number of returned samples exceeds the value in this field, the query fails and no data is returned.

The Memory/Recovery Subsection

Field	Description
Buffer Memory Max (MB)	The maximum memory buffer size that an archiver queue will use before starting to use disk buffering. The default value is 100 MB. **Note:** • You can monitor your collector data write queue using the Perftag_CollectorDataWriteQueueSize tag. If you find that the queue is exceeding 10,000 items, such as during a store and forward flush, change the value of this field to 500 or more to maintain Historian performance. • If you are upgrading from a previous version of Historian, the value in this field remains the same. You can change the value as needed.
Archiver Memory Size (MB)	The target memory usage of the archive. The default value is 0, which indicates the system will manage the memory usage. If the archiver is running on a 32-bit operating system and you want to keep more data in memory, you can enter a value up to 1800 MB. If the archiver is running on a 64-bit operating system, we recommend that you use the default value.
Maintain Auto Recovery Files	Indicates whether high availability of the latest archive (.iha) and Historian configuration (.ihc) files must be enabled. When enabled, a copy of the latest .iha and .ihc file is made once every hour.

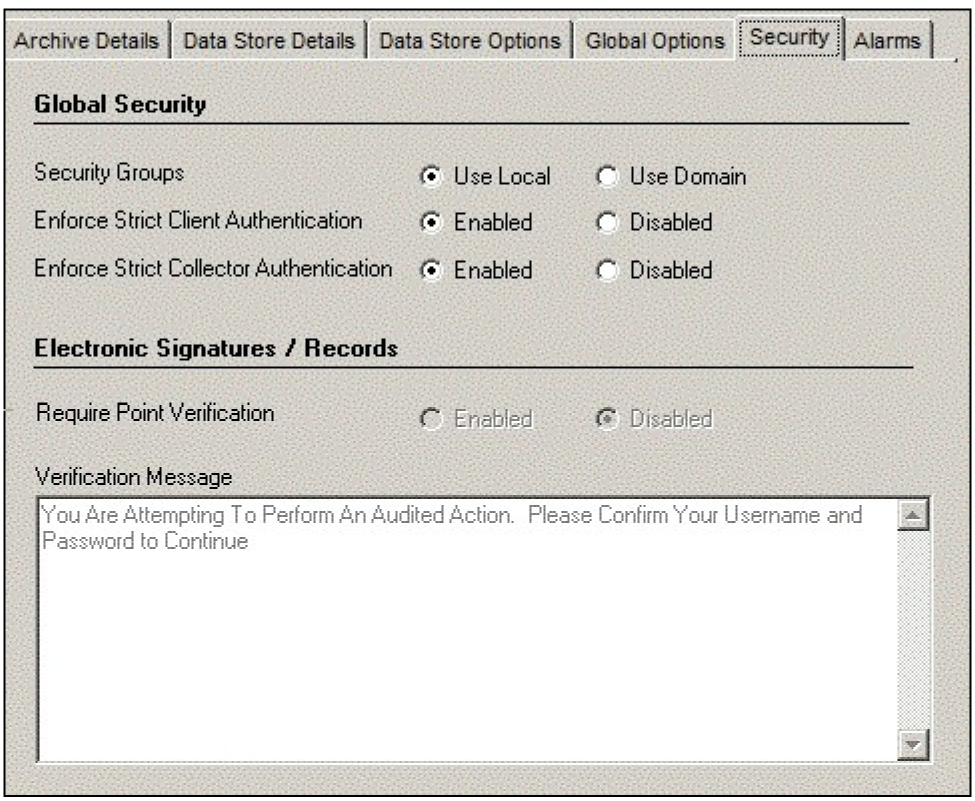
Field	Description
	Note: These files are managed internally by Historian, and should not be used as backup files. To create multiple archives at the same time, select the **Disabled** option. By default, this field is set to **Disabled** on a 64-bit operating system. On a large-scale system, we recommend that you disable this option for better performance.

The Data Store Subsection

Field	Description
Default Data Store For Tag Add	The name of the default data store to which you want to add tags.

The Security Section

This topic describes the fields in each subsection in the **Security** section



.

The Global Security Subsection

Field	Description
Security Groups	Indicates whether to use the local security groups or the domain security groups. **Note:** To ensure a secure environment when using Historian, do not create any local user accounts unless Historian is set up on a standalone computer and the guest account is disabled.
Enforce Strict Client Authentication	Indicates whether to use strict client authentication. If you enable this option, only clients using the security-token-based authentication protocol can connect. Clients using Historian versions prior to 6.0 and other Proficy software they connect to may not be able to connect unless they have the latest updates for that version. If you disable this option, clients of any version can connect if they use a valid user name and password.
Enforce Strict Collector Authentication	Indicates whether to use strict collector authentication. If you enable this option, only collectors using the security-token-based authentication protocol can connect. Collectors using Historian versions prior to 6.0 and the other Proficy software they connect to may not be able to connect unless they have the latest updates for that version. If you disable this option, collectors of any version can connect.

The Electronic Signatures / Records Subsection

The electronic signatures/records option assists users with government regulations such as the United States Food and Drug Administration's (FDA) 21 CFR Part 11 regulation or any site interested in added security by providing the ability to require a signature and password every time a change in data or configuration is requested. If you did not purchase the Electronic Signatures and Electronic Records option, the Electronic Signatures/Records field is disabled.

Field	Description
Require Point Verification	Indicates whether you must enter identifying information whenever you attempt a restricted action. Whenever you attempt to change the system con-

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>figuration (for the tag, archive, or collector), a tag value, or another record, you must electronically sign the action with a username and password. If the user is authorized to make this change, the identity of the person, the action performed, and the time it was performed, are all recorded in the audit trail.

Note:
The audit features are not dependent on this feature being enabled. Historian audits all user actions regardless of whether this option is enabled.
If you plan to create multiple archives at the same time, select the Disabled option.
Enabling electronic signatures and electronic records also requires you to reverify your identity when you use the Historian Excel add-in, modify or create a tag, or import data.

Note:
This feature is available only if you have purchased the Electronic Signatures and Electronic Records option.</td></tr>
<tr><td>Verification Message</td><td>When point verification is enabled, you are prompted to enter the username and password whenever you attempt to perform an action specified as requiring point verification.</td></tr>
</table>

Managing Data Stores

About Data Stores

A data store is a logical collection of tags. It is used to store, organize, and manage tags according to the data source and storage requirements. A data store can have multiple archive files (*.IHA), and includes both logical and physical storage definitions.

Tags can be segregated into separate archives through the use of data stores. The primary use of data stores is to segregate tags by data collection intervals. For example, you can put a name plate or static tags where the value rarely changes into one data store, and your process tags into another data store. This can improve query performance.

Historian data stores are stored as archive files that contain data gathered from all data sources during a specific period of time. You can write and read data from the archive files.

You can define two types of data stores:

- **Historical Data Store**: Tags stored under historical data store will store data as long as the disk space is available. Depending on your license, you may be able to create multiple historical data stores. The maximum number of historical data stores supported depends on the license.
- **SCADA Buffer Data Store**: Tags stored under the SCADA buffer data store will store data for a specific duration of time based on license.

When you install the Historian server, two historical data stores are installed by default.

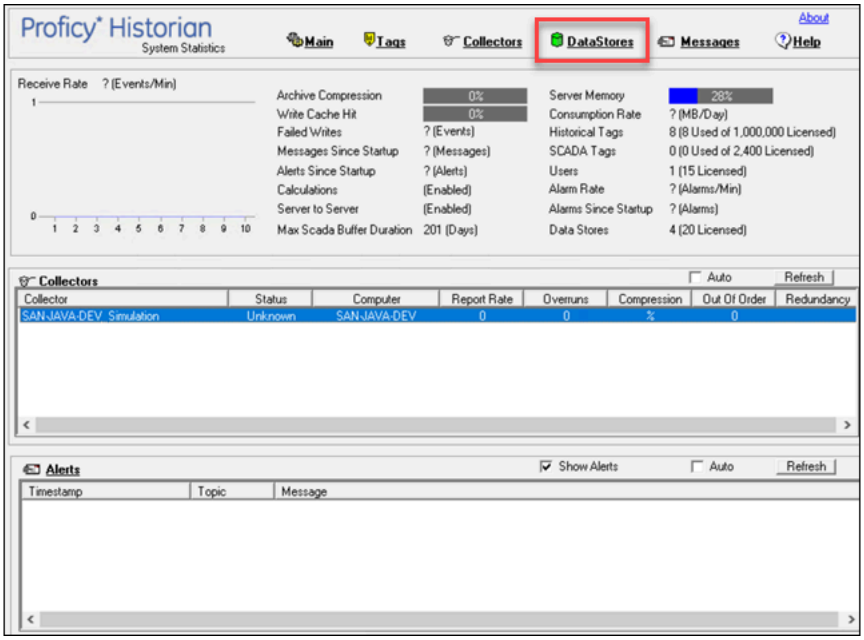
- **System**: Stores performance tags. This is only for internal usage within Historian, and you cannot add tags to this data store. You must not rename or delete the system data store.
- **User**: Stores tag data. This is a default data store. You can rename and delete a user data store as long as there is another default data store set for tag addition.

Based on your license, a SCADA Buffer data store may also be installed. It stores short-term tags and data.

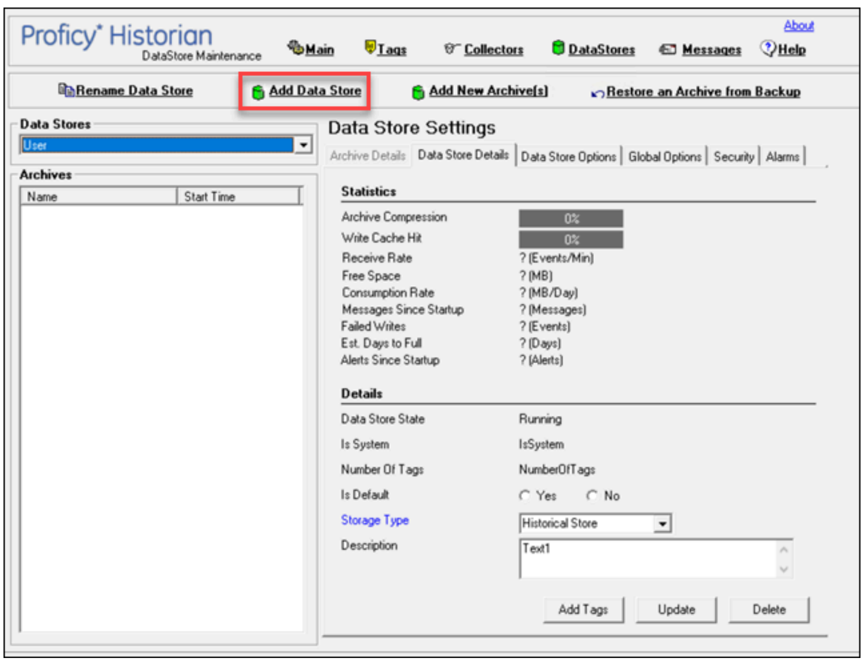
Create a Data Store

Depending on your license, you can create or add multiple data stores.

1. Access Historian Administrator *(on page 384)*.
2. Select **DataStores**.



3. Select **Add Data Store**.



The **Add New Data Store** window appears.

4. Enter values as described in the following table.

Field	Description
Data Store Name	Enter a unique name for the data store. The following characters are not allowed: `/\ \ * ? < > |`

Field	Description
Default Data Store	Select this check box to set this data store as the default one for adding tags. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default.
Description	Enter a description for the data store.

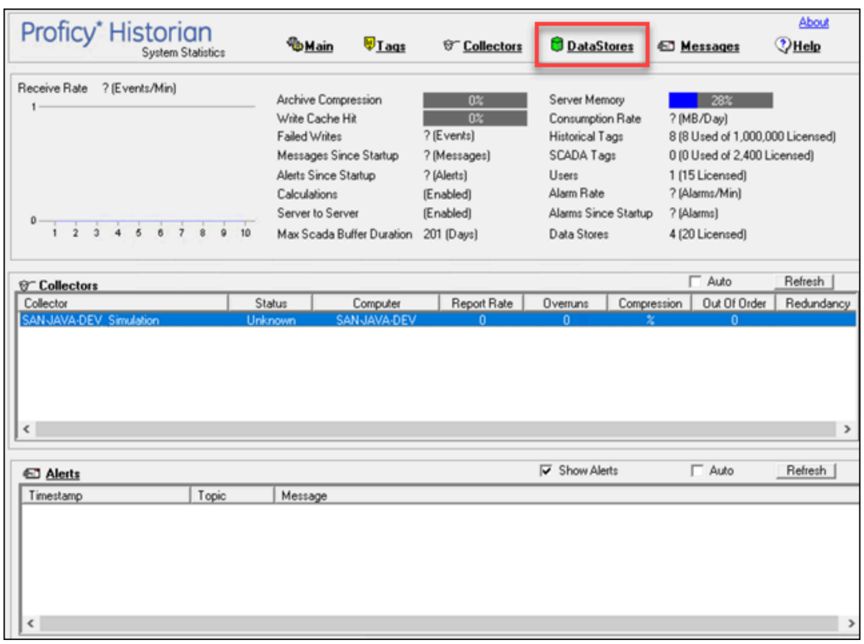
5. Select **OK**.

   The data store is created.

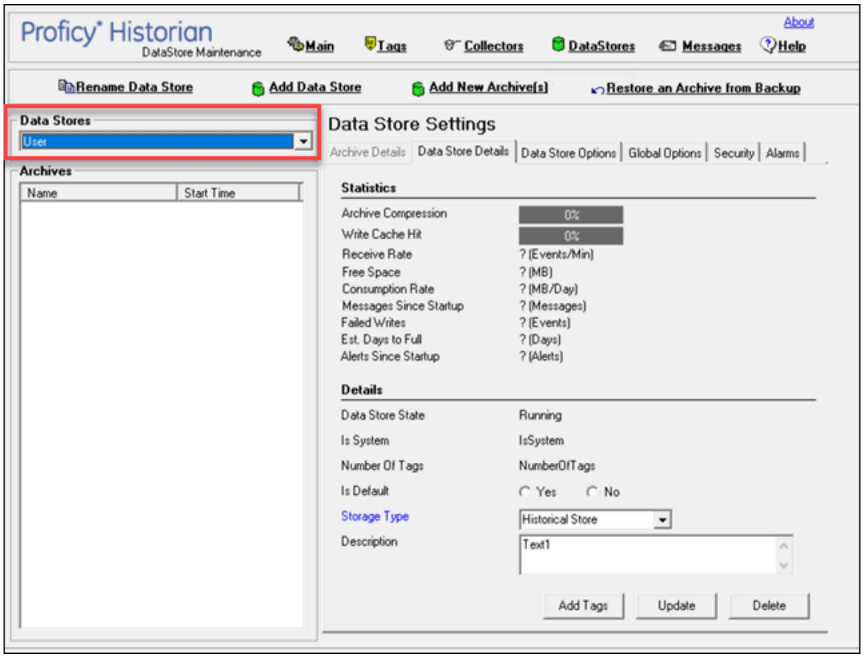
   When you add tags to the data store, it will have its own set of .IHA (iHistorian Archive) files. Ensure that you back up the new data store archives periodically using AFS.

Rename a Data Store

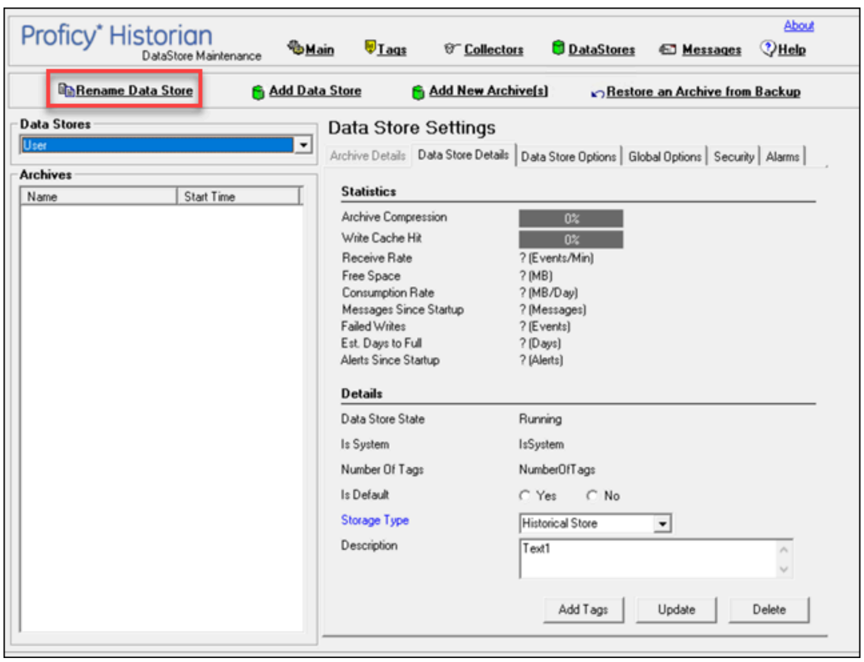
1. Access Historian Administrator *(on page 384)*.
2. Select **DataStores**.



3. In the **Data Stores** field, select the data store that you want to rename.



4. Select **Rename Data Store**.



The **Rename New Data Store** window appears.

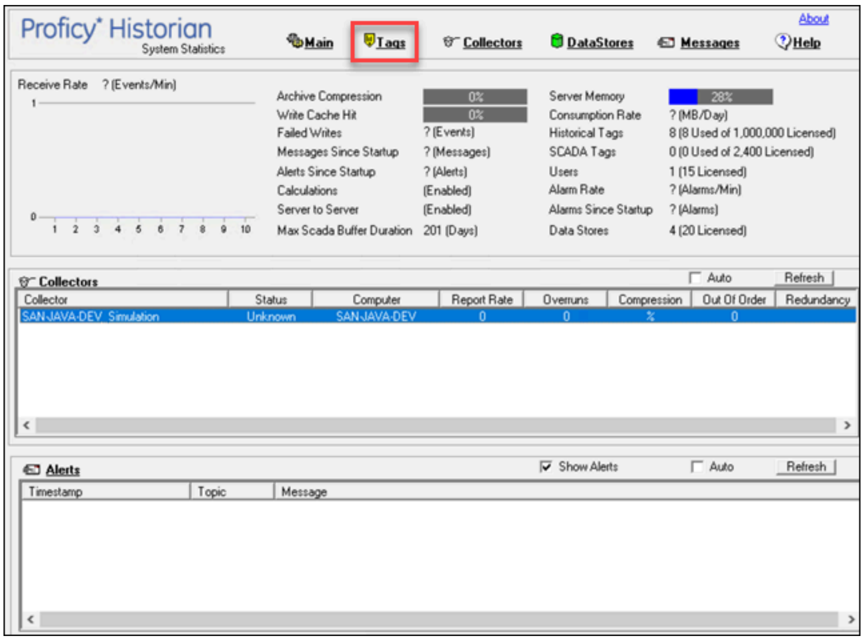
5. In the **New Data Store Name** field, enter the new name. The following special characters cannot be used in data store names: `/\ \ * ? < > |`
6. Select **Rename**.

   The data store is renamed.

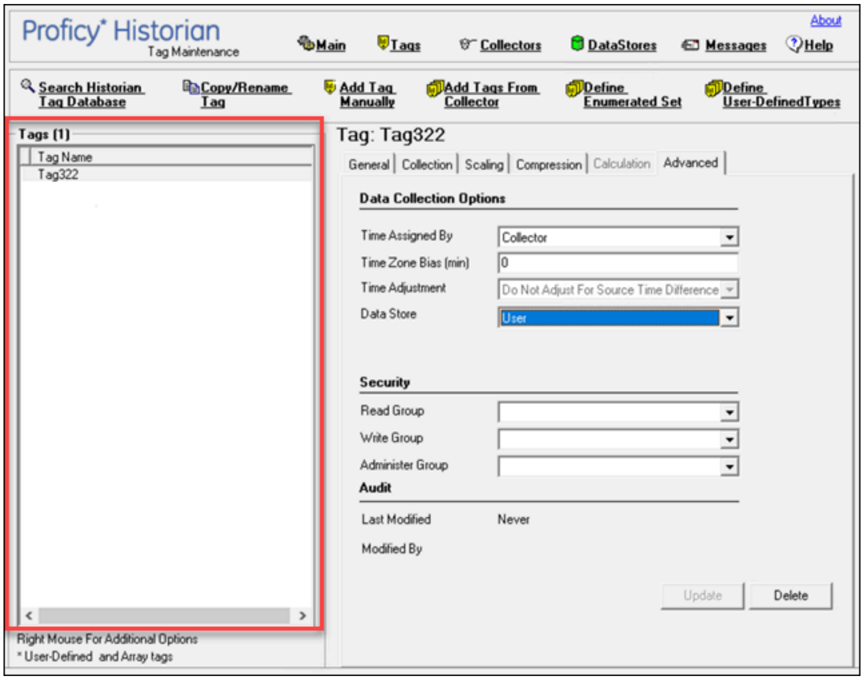
Move a Tag to Another Data Store

You can move tags from one data store to another. However, moving a tag does not automatically move the data associated with it. If you want to retrieve the data stored before the tag was moved, you have to move the data manually using the migration utility tool.

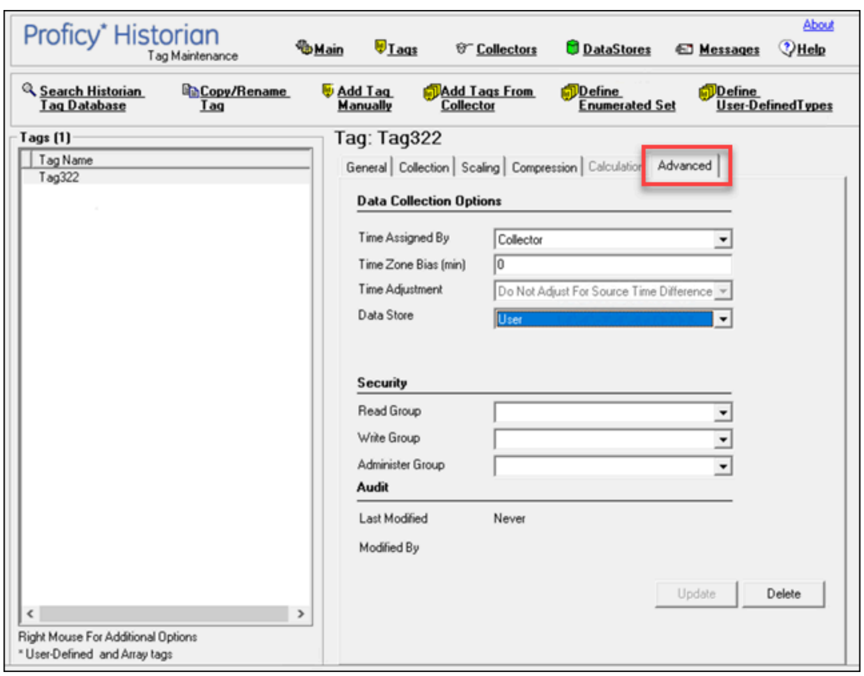
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select the tag that you want to move to a different data store.



4. Select **Advanced**.



5. In the **Data Store** field, select the data store to which you want to move the tag.

   A message appears, asking you to confirm that the you want to move the tag.

6. Select **Yes**, and then select **Update**.
   The tag has been moved. The new data for the tag will be stored in the new data store. However, if you want to store the old data as well in the new data store, you must manually migrate the tag data.

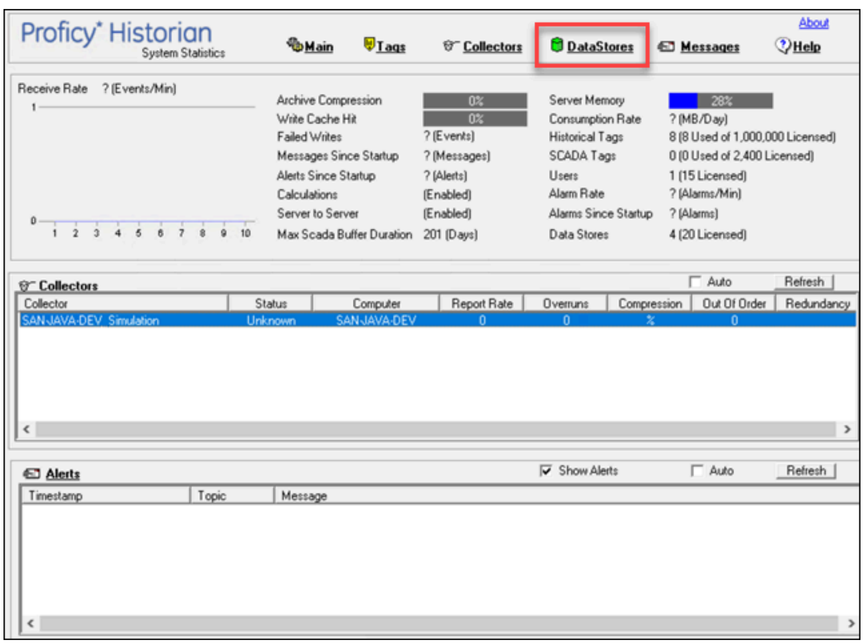
Delete a Data Store

You can delete a data store when it is no longer needed.

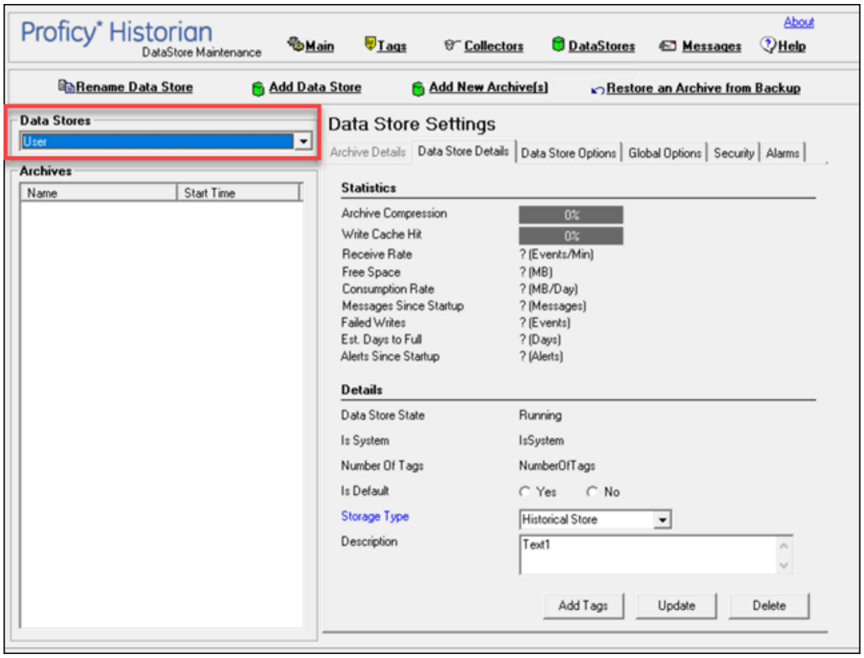
> **Note:**
>
> - You can only delete user data stores. You must not delete the system data store.
> - If you have only one user data store, you cannot delete it.

If there are any tags assigned to the data store, reassign them and manually move the data to another data store.

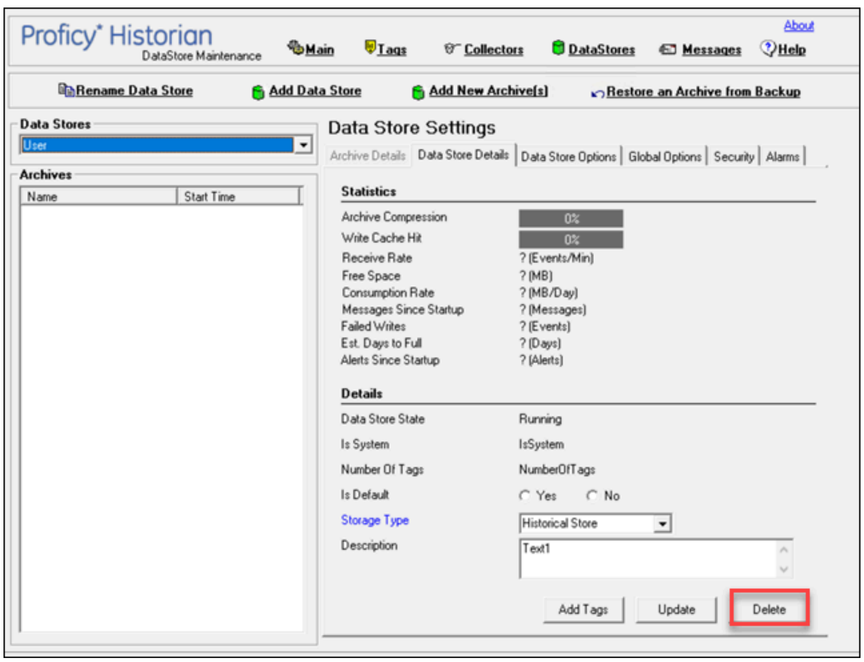
1. Access Historian Administrator *(on page 384)*.
2. Select **DataStores**.



3. In the **Data Stores** field, select the data store that you want to delete.



4. Select **Delete**.



A message appears, asking you to confirm that you want to delete the data store.

5. Select **Yes**.

The data store is deleted.

Managing Archives

About Archives

Historian archives are data files, each of which contains data gathered from all data sources during a specific period of time.

Types of Archive Files:

- ***machine name*_Config.ihc:** Contains information about the archiver, tag configuration, and collector configuration.
- ***machine name*_ArchiveXXX.iha:** Contains tag data, where x is a number indicating the place of the file in a time-based sequence.

Creation of Archive Files Automatically

Archive files grow to a user-configured maximum size as data is recorded by the server. When data starts loading into an archive file, Historian will automatically create a new blank archive file. When the current archive file becomes full, Historian will immediately serve data to the newly created archive file. This significantly reduces archive creation and transition time.

If, however, the option to automatically create archive files is not enabled, you must create an archive file manually *(on page 416)*.

Important:

- If the option to automatically create an archive is not enabled and you do not create a new archive manually, or if the available disk space is less than the required amount of free disk space, a new archive file will not be created.
- Ensure that the number of archive files does not exceed 1024. Otherwise, the archiver will crash. This is because 1024 is the default number of file descriptors a process can open on Linux. We recommend that you create archive files daily or by size so that you can monitor the number of archive files created.

Overriding Old Archive Files

If you enable the **Overwrite Old Archives** option, the system replaces the oldest archived data with new data when the latest archive default size has been reached. Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. Best practice is to create an additional archive to prevent premature loss of data due to overwriting. For example, if you want to save 12 months of data into 12 archives, create 13 archives.

During archiver startup and every 60 seconds while the server is running, Historian verifies that you have configured enough free disk space to save the archives, buffer files, and log files. If there is insufficient disk space, the Data Archiver shuts down and a message is logged into the log file. By default, you can view the Historian archiver log file in `C:\Historian Data\LogFiles`.

```
[03/03/10 15:28:41.398] Insufficient space available in [d:\Historian\Archives\]

        [03/03/10 15:28:41.399] The server requires a minimum of [5000 MB] to continue

        [03/03/10 15:28:41.679] USER: DataArchiver TOPIC: ServiceControl MSG: DataArchiver(DataArchiver)

        Archiver shutdown at 03/03/10 15:28:41.653

        [03/03/10 15:28:41.807] DataArchiver Service Stopped.

        [03/03/10 15:28:41.809] [d:\Historian\LogFiles\DataArchiver-34.log] Closed.
```

Guidelines for Setting Archive Size

Since archived data files can become quite large, you must adjust system parameters carefully to limit data collection to meaningful data only and thus minimize the required size of system storage. You can allocate up to 256 GB per archive.

For each archive, you need approximately 1MB of archive space for every 1000 tags to store tag information. Archive size is a function of the rate at which you archive data and the time period you want the archive to cover. A typical user wants the archive to cover a time period of, say, 30 days.

The following factors affect the rate at which you archive data:

- Number of tags
- Polling frequency of each tag
- Compression settings
- Data types

Based on these parameters, the archive size is calculated as follows:

$$\#Tags \times \frac{Values}{Tag} \times \frac{Tags}{Second} \times \%PassComp \times \frac{Bytes}{Value} \times \frac{Seconds}{Hour} \times \frac{Hours}{Day} \times \frac{MB}{Bytes} = \frac{MB}{Day}$$

Calculating Archive Size

Suppose you want to store data, and you have the following parameters:

- Number of tags: 5000
- Polling rate: 1 value/5 seconds
- Pass compression: 5%.

  Pass compression is the number of data values archived relative to the number of values read.

- Bytes/value: 4
- Duration: 30 days

Based on the preceding formula, for the given parameters, the archive size is calculated as follows:

$$5000 \times \frac{1}{1} \times \frac{1}{5} \times \frac{5}{100} \times \frac{4}{1} \times \frac{3600}{1} \times \frac{24}{1} \times \frac{1}{1024 \times 1024} \times 30 = 494 \frac{MB}{Month}$$

The calculation shows that a file size of 500 MB is adequate for archiving one month of data for this application.

Therefore, we recommend that you set the default archive size to 500 MB for systems with 1000 tags or more. If you believe the computed size is too large for your application, you can modify parameters as follows:

- Decrease the polling frequency.
- Increase compression deadband, reducing the pass percentage.
- Reduce the number of tags.
- Add more disk capacity to your computer.

Archive Size Calculator

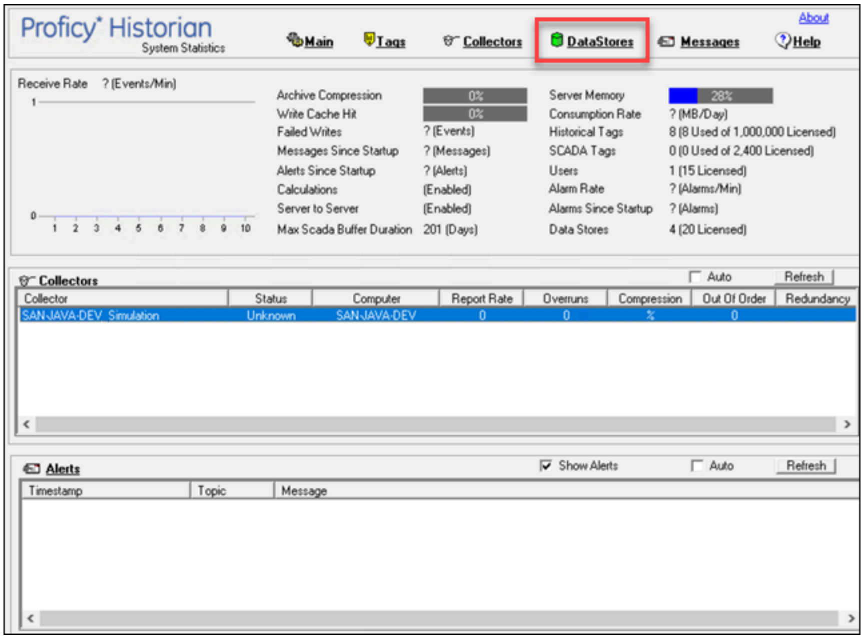
An archive size calculator tool is available to estimate archive size and collector compression based on a tag that has already been configured or based on your inputs. Log on to http://digitalsupport.ge.com to download this tool and other GE Intelligent Platforms freeware product solutions.

Create an Archive Automatically

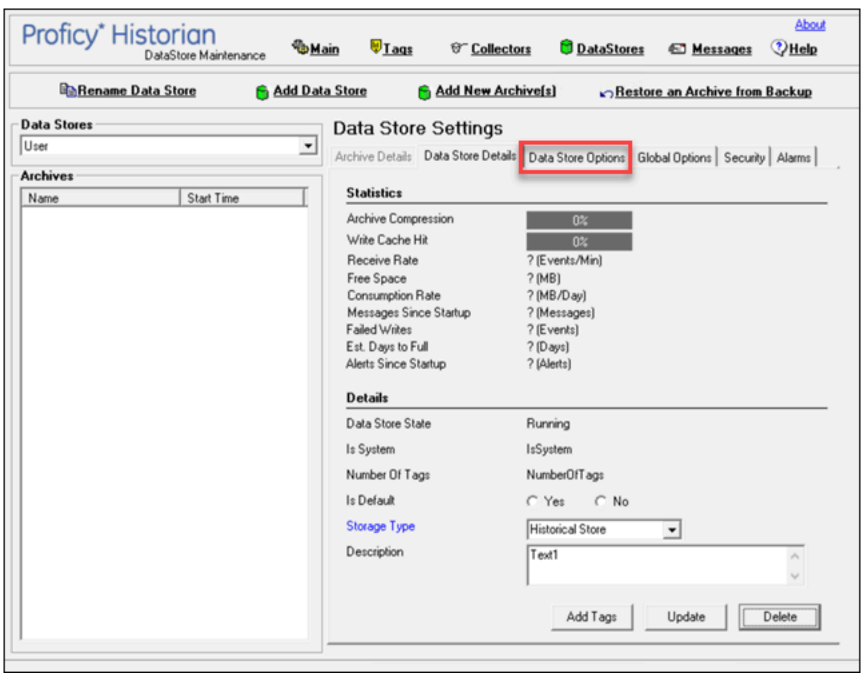
When the current archive reaches a specified size or duration, you can configure Historian to create a new archive automatically. You can also create an archive manually *(on page 419)*. When the current archive is full, the new one is used.

You can allocate maximum 256 GB for an archive.

1. Access Historian Administrator *(on page 384)*.
2. Select **DataStores**.



3. Select **Data Store Options**.



4. Enter values as described in the following table.

Field	Description
Automatically Create Archives	Select **Enabled**.

Field	Description
	Important: ◦ If the option to automatically create an archive is not enabled and you do not create a new archive manually, or if the available disk space is less than the required amount of free disk space, a new archive file will not be created. ◦ Ensure that the number of archive files does not exceed 1024. Otherwise, the archiver will crash. This is because 1024 is the default number of file descriptors a process can open on Linux. We recommend that you create archive files daily or by size so that you can monitor the number of archive files created.
Overwrite Old Archives	Specify whether you want to overwrite old archives with new ones. Exercise caution in enabling this option. We recommend that youback up archives if you want to enable this option.
Default Size	Enter the size of the current archive after which you want to create a new archive. This field is available only if you have selected **By Size** in the adjacent drop-down list box. If, however, you want to create archives after a duration, select **Days** or **Hours**, and then enter the value in the **Archive Duration** field.
Archive Duration	Enter the duration after which you want to create a new archive. This field is available only if you select **Days** or **Hours** in the adjacent drop-down list box. If, however, you want to create an archive when the current one reaches a particular size, select **By Size**, and then enter the value in the **Default Size** field.
Default Archive Path	Enter the path to the folder in which you want to store the archive files.
Default Backup Path	Not applicable. Use AFS to back up and restore archives.

Field	Description
Base Archive Name	Not applicable. Use AFS to back up and restore archives.
Free Space Required	Enter the free space that is required to create the archives.
Store OPC Quality	Specify whether you want to store OPC quality in the archive.
Use Caching	Specify whether you want to use caching in the archive.
Data is Read-Only After (Hours)	Specify the duration, in hours, after which you want to archive to be read-only.
Generate Message on Data Update	Specify whether you want to generate a message when data is updated in the archive.

5. Select **Update**.
   Archives will be created automatically when the current one reaches the size (or after the duration) that you have specified.

Create Archives Manually

If you want to create multiple archives at the same time, access Historian Administrator, and set values for the following fields:

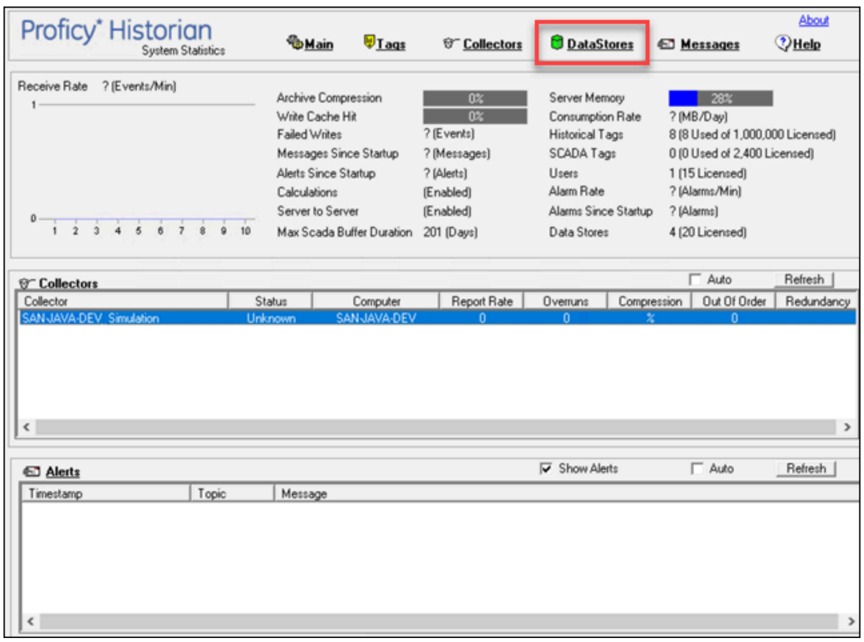
Field	Value
The **Details** Section	
File Attribute	**Read/Write**
The **Global Options** Section	
Maximum Query Time (seconds)	60
Maximum Query Intervals	100000
Automatically Create Archives	Disabled
Overwrite Old Archives	Enabled
Maintain Auto Recovery Files	Enabled
Store OPC Quality	Disabled
The **Security** section	
Data is Readonly After (Hours)	1 month

Field	Value
Security Groups	Use local
Generate Message on Data Update	Disabled
Require Point Verification	Disabled

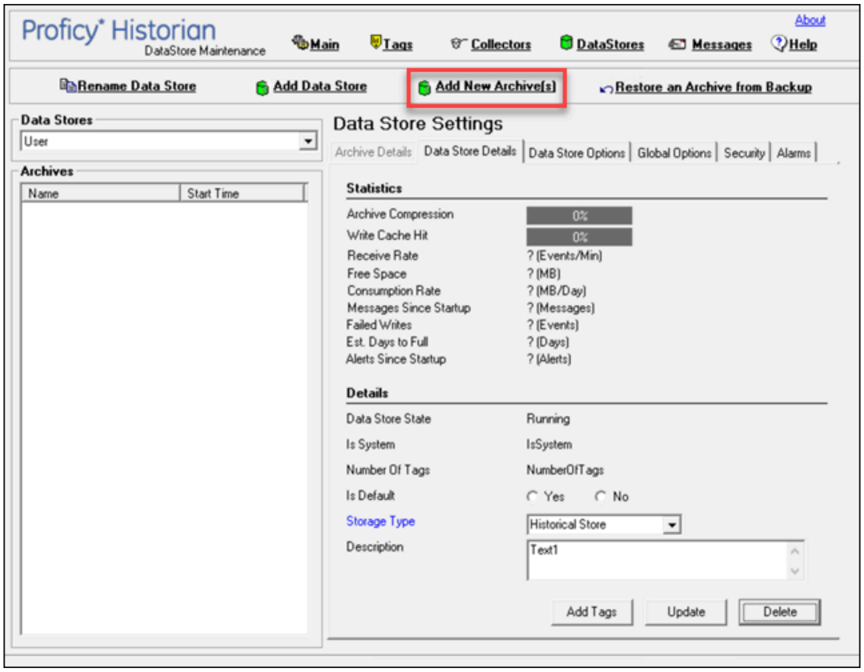
This topic describes how to create archives manually. You can also create them automatically *(on page 416)*. When the current archive is full, a new archive is used (in a sequential order).

You can create multiple archives at the same time.

1. Access Historian Administrator *(on page 384)*.
2. Select **DataStores**.



3. Select **Add New Archive(s)**.



The **Add New Archive(s)** window appears.

4. Enter values as described in the following table.

Field	Description
Archive Name	Enter a unique name for the archives. The value must be the same as the file name. When multiple archives are created, a number is appended to the name to make each name unique (and to maintain a sequence).
Data Store	Select the data store in which you want to create the archives.
File Location	Enter the path to the folder in which you want to store the archives, or specify a UNC path.
EachArchive Size (MB)	Enter the size, in MB, that you want to allocate to the archives.
Number of Archives	Enter the number of archives you want to create. **Important:** Ensure that the number of archive files does not exceed 1024. Otherwise, the archiver will crash. This is because 1024 is the default number of file descriptors a process can open on Linux. We recommend that you

Field	Description
	create archive files daily or by size so that you can monitor the number of archive files created.
Allocate Space	Specify the percentage of the disk space that you want to allocate for archives. As you increase the space, the number of archives increases accordingly. **Note:** The **Allocate Space** field does not display a remote machine's hard disk space; if you are creating multiple archives on a remote machine, you must ignore the ”r;percentage of available disk space will be used” message displayed by the Allocate Space slider.

5. Select **OK**.
   The archives are created.

Managing Tags

About Tags

A Historian tag is used to store data related to a property.

For example, if you want to store the pressure, temperature, and other operating conditions of a boiler, a tag will be created for each one in Historian.

When you collect data using a collector, tags are created automatically in Historian to store these values. These tags are mapped with the corresponding properties in the source.

For example, suppose you want to store OSI PI data in Historian. You will specify the OSI PI tags for which you want to collect data. The OSI PI collector creates the corresponding tags in Historian, and it stores the values in those tags.

You can also choose to create tags manually.

About Collector and Archive Compression

Collector Compression

Collector compression applies a smoothing filter to data retrieved from the data source. By ignoring small changes in values that fall within a deadband centered around the last reported value, only significant changes are reported to the archiver. Fewer samples reported yields less work for the archiver and less archive storage space used.

You can specify the deadband value. For convenience, if you enter a deadband percentage, Historian Administrator shows the deadband in engineering units. For example, if you specify a 20% deadband on 0 to 500 EGU span, it is calculated and shown as 100 engineering units. If you later change the limits to 100 and 200, the 20% deadband is now calculated as 20 engineering units.

The deadband is centered around the last reported sample, not simply added to it or subtracted. If your intent is to have a deadband of 1 unit between reported samples, you must enter a compression deadband of 2 so that it is one to each side of the last reported sample. In the previous example of 0 to 500 EGU range, with a deadband of 20%, the deadband is 100 units; This means that only if the value changes by more than 50 units, it is reported.

Changes in data quality from good to bad, or bad to good, automatically exceed collector compression and are reported to the archiver. Any data that comes to the collector out of time order will also automatically exceed collector compression.

It is possible for collected tags with no compression to appear in Historian as if the collector or archive compression options are enabled. If collector compression occurs, you will notice an increase in the percentage of the compression value in the **Collectors** section of the **System Statistics** page in Historian Administrator. When archive compression occurs, you will notice the archive compression value and status bar change on the **System Statistics** page.

For instructions on setting collector compression, refer to Access/Modify a Tag *(on page 430)*.

Even if collector compression is not enabled, you may notice it in the following scenarios:

- When a succession of bad data quality samples appears, Historian collects only the first sample in the series. No new samples are collected until the data quality changes. Historian does not collect the redundant bad data quality samples, and this is reflected in the collector compression percentage.
- For a Calculation or Server-to-Server collector, when calculations fail, producing no results or bad quality data, collector compression is used. The effect of Collector Compression Timeout is to behave, for one poll cycle, as if the collector compression feature is not being used. The sample

collected from the data source is sent to the archiver. Then the compression is turned back on, as configured, for the next poll cycle with new samples being compared to the value sent to the archiver.

Handling Value Step Changes with Collector Data Compression

If you enable collector compression, the collector does not send values to the archiver any new input values if the value remains within its compression deadband. Occasionally, after several sample intervals inside the deadband, an input makes a rapid step change in value during a single sample interval. Since there have been no new data points recorded for several intervals, an additional sample is stored one interval before the step change with the last reported value to prevent this step change from being viewed as a slow ramp in value. This value marks the end of the steady-state, non-changing value period, and provides a data point from which to begin the step change in value.

> **Note:**
> You can configure individual tags can be configured to retrieve step value changes.

The collector uses an algorithm that views the size of the step change and the number of intervals since the last reported value to determine if a marker value is needed. The following is an example of the algorithm:

```
BigDiff=abs(HI_EGU-LO_EGU)*(CompressionDeadbandPercent/(100.0*2.0))*4.0

If ( Collector Compression is Enabled )

If ( Elapsed time since LastReportedValue>=( SampleInterval * 5 ) )

If ( abs(CurrentValue-LastReportedValue) > BigDiff )

Write LastReportedValue,Timestamp=(CurrentTime-SampleInterval)
```

In the example above, if a new value was not reported for at least the last 4 sample intervals, and the new input value is at least 4 deltas away from the old value (where a single delta is equal to half of the compression deadband), then a marker value is written.

> **Note:**
> These settings are also adjustable from the Registry. Please contact technical support for more information.

Value Spike with Collector Compression

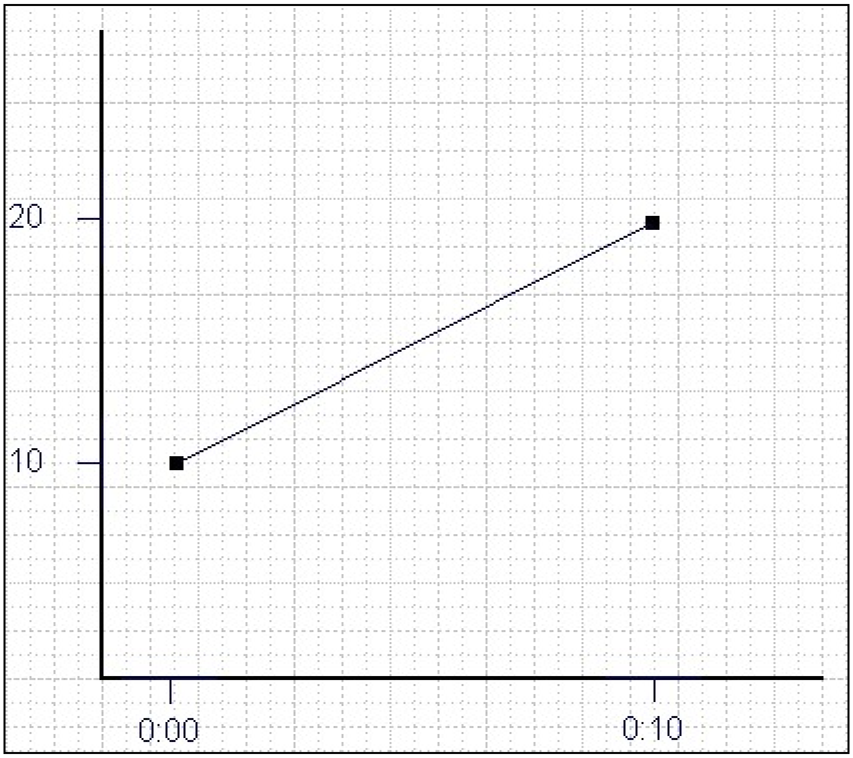
For example, a collector reads a value X once per second, with a compression deadband of 1.0. If the value of X is 10.0 for a number of seconds starting at 0:00:00 and jumps to 20.0 at 0:00:10, the data samples read would be:

Time	X Value
0:00:00	10.0 (steady state value)
0:00:01	10.0
0:00:02	10.0
0:00:03	10.0
0:00:04	10.0
0:00:05	10.0
0:00:06	10.0
0:00:07	10.0
0:00:08	10.0
0:00:09	10.0
0:00:10	20.0 (new value after step change)

To increase efficiency, the straightforward compression would store only 2 of these 11 samples.

Time	X Value
0:00:00	10.0 (steady state value)
0:00:10	20.0 (new value after step change)

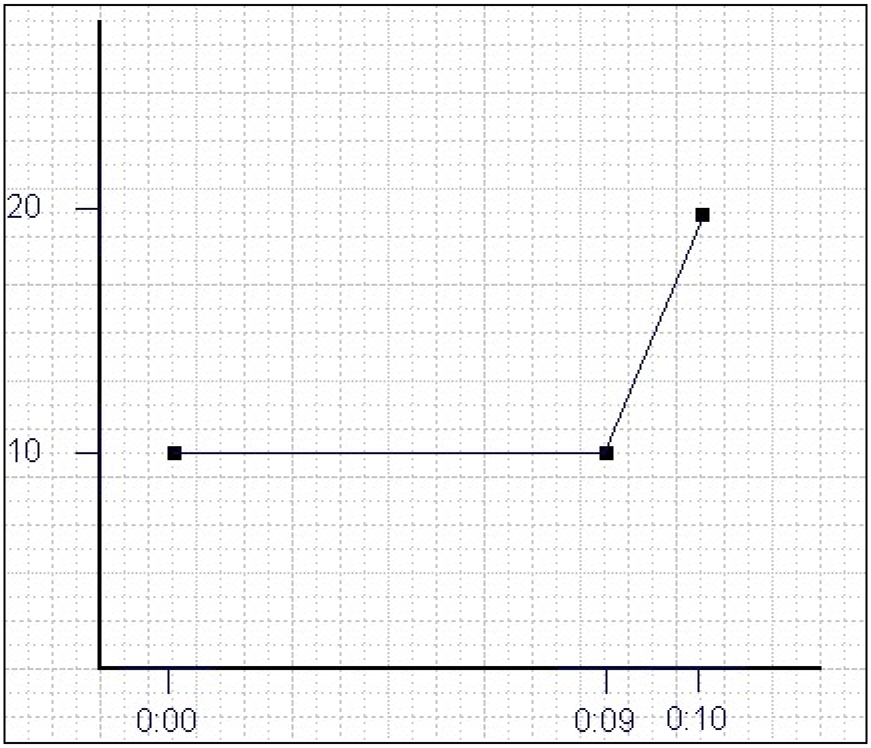
However, without the marker value, if this data were to be put into a chart, it would look like the data value **ramped** over 10 seconds from a value of 10.0 to 20.0, as shown in the following chart.



The addition of a **marker value** to the data being stored results in the following data values:

Time	X Value
0:00:00	10.0 (steady state value)
0:00:09	10.0 (inserted Marker value)
0:00:10	20.0 (new value after step change)

If you chart this data, the resulting trend accurately reflects the raw data and likely real world values during the time period as shown in the following chart.



Evaluating and Controlling Data Compression

You can achieve optimum performance in Historian by carefully controlling the volume of dynamic data it collects and archives. You need enough information to tell you how the process is running, but you do not need to collect and store redundant or non-varying data values that provide no useful information.

Control Data Flow

You can control the amount of online or dynamic data the system handles at a given time by adjusting certain system parameters. The general principle is to control the flow of data into the archive either by adjusting the rate at which the collectors gather data or by adjusting the degree of filtering (compression) the system applies to the data collected.

Adjust the following parameters to *reduce* the rate of data flow into the server.

- Reduce the polling rate by increasing the collection interval for unsolicited and polled collection.
- Enable collector compression and optionally use compression timeout.
- Set the compression deadband on the collectors to a wider value.
- Use the collector compression timeout.

Adjust the following parameters to *increase the filtering* applied by the archiver in the server.

- Enable archive (trend) compression.
- Set the archive compression deadband to a wider value.
- Where possible, use the scaled data type and enable input scaling on selected tags.
- Where possible, select milliseconds or microseconds rather than seconds for time resolution. Seconds is optimum for most common devices. This affects disk space.

Evaluate Data Compression Performance

You can determine how effectively data compression is functioning at any given time by examining the system statistics displayed on the **System Statistics** page of Historian Administrator.

The compression field at the top of the page shows the current effect of archive compression. Values for this parameter should typically range from 0 to 9%. If the value is zero, it indicates that compression is either ineffective or turned off. If it shows a value other than zero, it indicates that archive compression is operating and effective. The value itself indicates how well it is functioning. To increase the effect of data compression, increase the value of archive compression deadband so that compression becomes more active.

Archive Compression

Archive compression is used to reduce the number of samples stored when data values for a tag form a straight line in any direction. For a horizontal line (non-changing value), the behavior is similar to collector compression. But, in archive compression, it is not the *values* that are being compared to a deadband, but the *slope of line* those values produce when plotted value against time. Archive compression logic is executed in the data archiver and, therefore, can be applied to tags populated by methods other than collectors.

You can use archive compression on tags where data is being added to a tag by migration. Each time the archiver receives a new value for a tag, the archiver computes a line between this incoming data point and the last archived value.

The deadband is calculated as a tolerance centered about the slope of this line. The slope is tested to see if it falls within the deadband tolerance calculated for the previous point. If the new point does not exceed the tolerance, it is not stored in the archive. This process repeats with subsequent points. When an incoming value exceeds the tolerance, the value held by the archiver is written to disk and the incoming sample is withheld.

The effect of the archive compression timeout is that the incoming sample is automatically considered to have exceeded compression. The withheld sample is archived to disk and the incoming sample becomes the new withheld sample. If the Archive Compression value on the System Statistics page indicates that archive compression is occurring, and you did not enable archive compression for the tags, the reason could be because of internal statistics tags with archive compression enabled.

For instructions on setting archive compression, refer to Access/Modify a Tag *(on page 430)*.

About Scaling

Scaling converts a data value from a raw value expressed in an arbitrary range of units, such as a number of counts, to one in engineering units, such as gallons per minute or pounds per square inch. The scaled data type can serve as a third form of data compression, in addition to collector compression and archive compression, if it converts a data value from a data type that uses a large number of bytes to one that uses fewer bytes.

For instructions on setting the scaling parameters, refer to Access/Modify a Tag *(on page 430)*.

About Condition-Based Collection

Condition based collection is a method to control the storage of data for data tags by assigning a condition. Data is always collected but it is only written to the Data Archiver if the condition is true; otherwise, the collected data is discarded.

This condition is driven by a trigger tag; a tag collected by the collector evaluating the condition. Ideally, Condition based Collection should be used only with tags that are updating faster than the trigger tag. Condition based collection can be used to archive only the specific data which is required for analysis, rather than archiving data at all times, as the collector is running.

For example, if a collector has tags for multiple pieces of equipment, you can stop collection of tags for one piece of equipment during its maintenance. It is typically used on tags that use fast polled collection but you don't want to use collector compression. While the equipment is running, you want all the data but when the equipment is stopped, you don't want any data stored. The trigger tag would also typically use polled collection. But, either tag could use unsolicited collection.

The condition is evaluated every time data is collected for the data tag. When a data sample is collected, the condition is evaluated and data is either queued for sending to archiver, or discarded. If the condition cannot be evaluated as true or false, like if the trigger tag contains a bad data quality or the collector is not collecting the trigger tag, the condition is considered true and the data is queued for sending.

No specific processing occurs when the condition becomes true or false. If the condition becomes true, no sample is stored to the data tag using that condition, but the data tag will store a sample next time it collects. When the condition becomes false, no end of the collection marker is stored until the data tag is collected.

For example, if the condition becomes false at 1:15 and the data tag gets collected at 1:20, the end of collection marker will be created at 1:20 and have a timestamp of 1:20, not 1:15.

Condition based collection is supported by only archiver and collectors of Historian version 4.5 and above. Condition based collection does not apply to alarm collectors. This condition based collection is applicable to the following collectors only:

- Simulation Collector
- OPC Collector
- iFIX Collector
- PI Collector

For instructions on setting the condition-based collection, refer to Access/Modify a Tag *(on page 430)*.

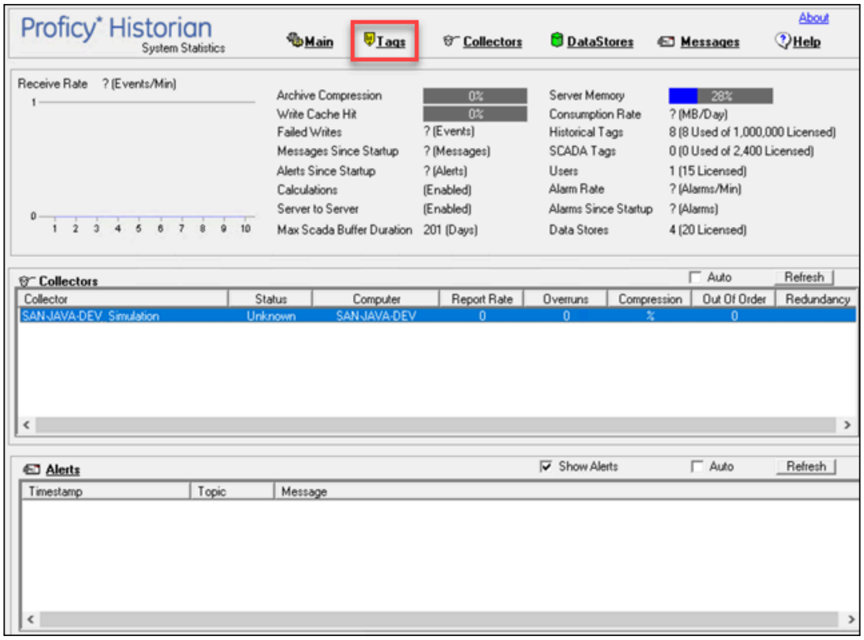
Access/Modify a Tag

T modify a tag, you must be a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).

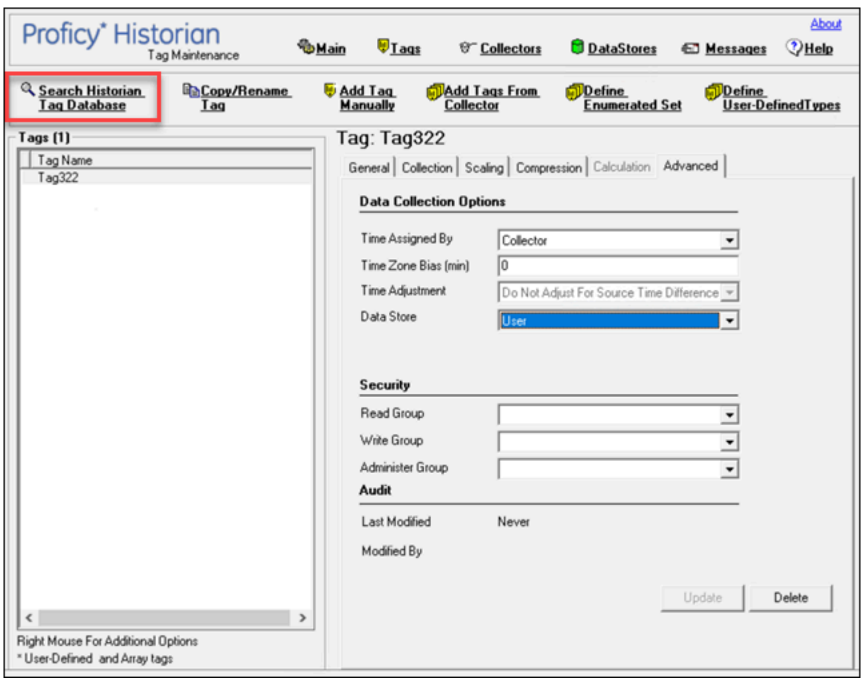
Using Historian Administrator, you can access a list of tags in the Historian database by their name, description, or both.

> **Note:**
> By default, maximum one million tags are retrieved. If the Historian clients are configured to retrieve more than a million tags, to retrieve all of them, add the `MaxTagsToRetrieve` registry key under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\DataArchiver\`, and then set the maximum number of tags that you want to retrieve. Restart the Historian Data Archiver service for the change to reflect.

1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select **Search Historian Tag Database**.

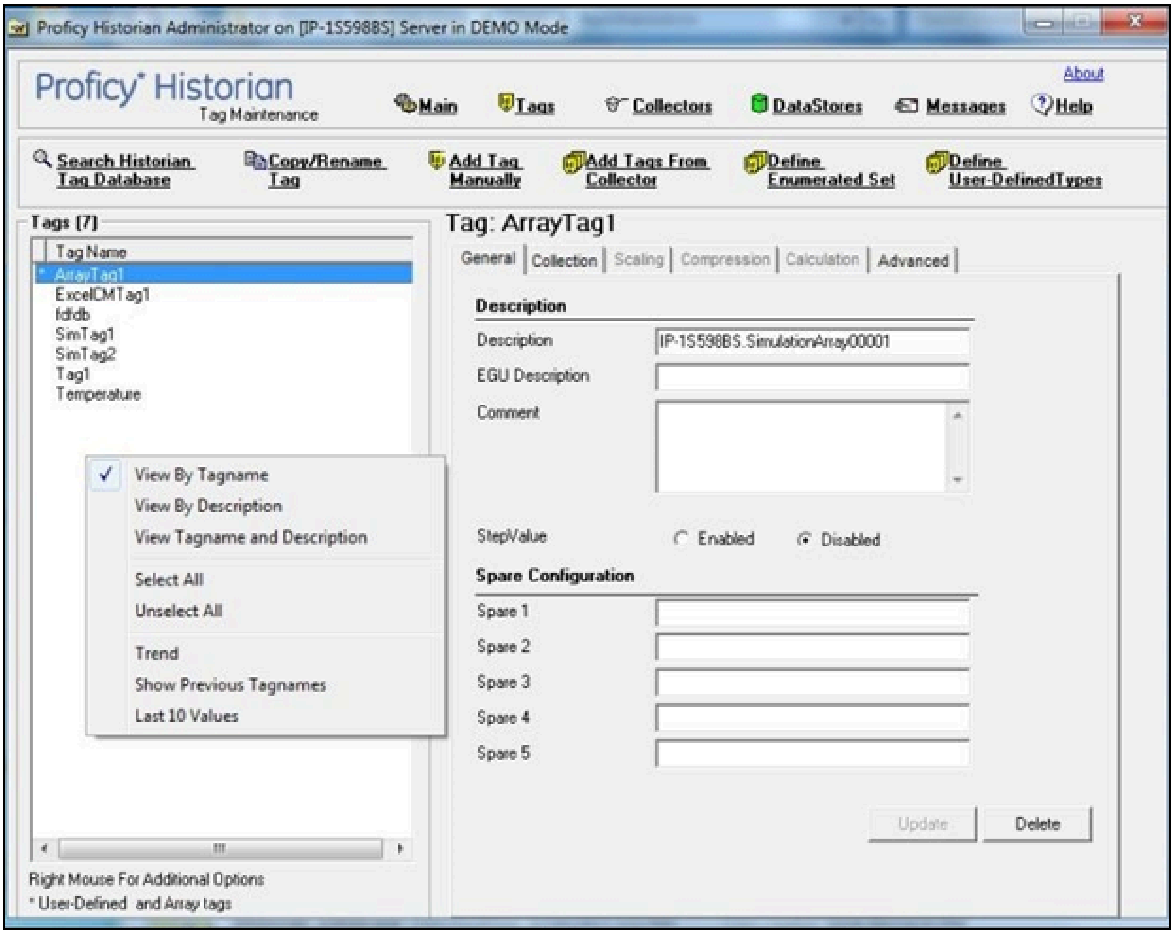


The **Search Historian Tag Database** window appears.

4. Enter values in the available fields to search for the tag, and then select **OK**. You can use the wildcard character asterisk (*).

   A list of tags that meet the search criteria appear in the **Tags** section.
5. Right-click the **Tags** section, and then select one of the following values:

- **View By TagName**: Select this option to view only the names of the tags.
- **View By Description**: Select this option to view only the descriptions of the tags.
- **View Tagname and Description**: Select this option to view both the names and descriptions of the tags.



6. As needed, modify values as described in the following tables, and then select **Update**.

Table 65. The General Section

Field	Description
Description	The description of the tag.
EGU Description	The engineering units assigned to the tag.
Comment	Comments that apply to the tag.
StepValue	Indicates that the actual measured value changes in a sharp step in-stead of a smooth linear interpolation. This option is applicable only for

Field	Description
	numeric data. Enabling this option only affects data retrieval; it has no effect on data collection or storage.
Spare Configuration	The Spare 1 through Spare 5 fields list any configuration information stored in these fields. **Note:** Do not add or update the spare configurations as the data may get corrupted or overwritten. For example, the **Spare 5** field is used by the Server-to-Server collector for internal purposes.

Table 66. The Collection Section

Field	Description
Collector	The name of the collector that collects data for the selected tag.
Source Address	The address for the tag in the data source. Leave this field blank for tags associated with the Calculation or Server-to-Server collector. For Python Expression tags, this field contains the full applicable JSON configuration, which includes an indication of the source address. **Note:** When exporting or importing tags using the EXCEL Add-In, the Calculation column, not the SourceAddress column, holds the formulas for tags associated with the Calculation or Server-to-Server collector.
Data Type	The data type of the tag. The main use of the scaled data type is to save space, but this results in a loss of precision. Instead of using 4 bytes of data, it only uses 2 bytes by storing the data as a percentage of the EGU limit. Changing the EGU limits will result in a change in the values that are displayed. For example, if the original EGU values were 0 to 100 and a value of 20 was stored using the scaled data type and if the EGUs are changed to 0 to 200, the original value of 20 will be represented as 40.

Field	Description
	Note: If you change the data type of an existing tag between a numeric and a string or binary data type (and vice versa), the tag's compression and scaling settings will be lost.
Data Length	The number of bytes for a fixed string data type. This field is enabled only for fixed string data types.
Is Array Tag	Not applicable
Collection	Indicates whether data collection is enable or disable for the tag. If you disable collection for the tag, Historian stops collecting data for the tag, but does not delete the tag or its data.
Collection Type	The type of data collection used for this tag, which can be polled or unsolicited. Polled means that the data collector requests data from the data source at the collection interval specified in the polling schedule. Unsolicited means that the data source sends data to the collector whenever necessary (independent of the data collector polling schedule).
Collection Interval	The time interval between readings of data from this tag. With Unsolicited Collection Type, this field defines the minimum interval at which unsolicited data should be sent by the data source.
Collection Offset	Used with the collection interval to schedule collection of data from a tag. For example, to collect a value for a tag every hour at thirty minutes past the hour (12:30, 1:30, 2:30, and so on), enter a collection interval of 1 hour and an offset of 30 minutes. Similarly, to collect a value each day at 8am, enter a collection interval of 1 day and an offset of 8 hours. **Note:** If you enter a value in milliseconds, the value must be in intervals of 1000 ms. For example, 1000, 2000, and 3000 ms are valid values, but 500 and 1500 ms are invalid. The minimum value is 1000 ms.

Field	Description
Time Resolution	The precision for timestamps, which can be either seconds, milliseconds or microseconds.
Condition-Based	Indicates whether condition-based data collection *(on page 429)* is enabled.
Trigger Tag	The name of the trigger tag used in the condition.
Comparison	The comparison operator that you want to use in the condition. Select one of the following options: ◦ **Undefined**: Collection will resume only when the value of the triggered tag changes. This is considered an incomplete configuration, so condition-based collection is turned off and all the collected data is sent to archiver. ◦ **< =**: Setting condition as trigger tag value less than or equal to the compare value. ◦ **> =** Setting condition as trigger tag value greater than or equal to the compare value. ◦ **<**: Setting condition as trigger tag value less than the compare value. ◦ **>**: Setting condition as trigger tag value greater than the compare value. ◦ **=**: Setting condition as trigger tag value equals compare value. ◦ **!=**: Setting condition as trigger tag value not the same as compare value.
Compare Value	A target value that you want to compare with the value of the trigger tag. If using = and != comparison parameters, ensure that the format of the compared value and triggered tag are the same. For example, for a float type trigger tag, the compare value must be a float value; otherwise, the condition result is an invalid configuration. When the configuration is invalid, condition-based collection is disabled and all data is sent to archiver.
End of Collection Markers	Indicates whether end-of-collection markers are enabled. This will mark all the tag's values as bad, and sub-quality as ConditionCollectionHalted when the condition becomes false. Trending and reporting applications can use this information to indicate that the real-world value was unknown after this time until the condition becomes true and a new sam-

Field	Description
	ple is collected. If disabled, a bad data marker is not inserted when the condition becomes false.

Table 67. The Scaling Section

Field	Description
Hi Engineering Units	The current value of the upper range limit of the span for this tag. Engineering Hi and Lo are retrieved automatically for F_CV fields for iFIX tags; all others are left at default settings. When adding tags from the server using an OPC Collector, the OPC Collector queries the server for the EGU units and EGU Hi/Lo limits. Not all OPC Servers make this information available, however. Therefore, if the server does not provide the limits when requested to do so, the collector automatically assigns an EGU range of 0 to 10,000.
Lo Engineering Units	The current value of the lower range limit of the span for this tag.
Input Scaling	Indicates whether input scaling is enabled, which converts an input data point to an engineering units value. For example, to rescale and save a 0 - 4096 input value to a scaled range of 0 - 100, enter 0 and 4096 as the low and high input scale values and 0 and 100 as the low and high engineering units values, respectively. If a data point exceeds the high or low end of the input scaling range, Historian logs a bad data quality point with a ScaledOutOfRange subquality. In the previous example, if your input data is less than 0, or greater than 4096, Historian records a bad data quality for the data point. **OPC Servers and TRUE Values:** Some OPC servers return a TRUE value as -1. If your OPC server is returning TRUE values as -1, modify the following scaling settings in the **Tag Maintenance** page of Historian Administrator:

Field	Description

	`Hi Engineering Units = 0`
	`Lo Engineering Units = 1`
	`Hi Scale Value = 0`
	`Lo Scale Value = - 1`
	`Input Scaling = Enabled`

Hi Scale Value	The upper limit of the span of the input value.
Lo Scale Value	The lower limit of the span of the input value.

Table 68. The Compression Section

Field	Description
Collector Compression	Indicates whether collector compression *(on page 423)* is enabled.
Collector Deadband	The current value of the compression deadband. This value can be computed as a percent of the span, centered around the data value or given as an absolute range around the data value. **Note:** Some OPC servers add and subtract the whole deadband value from the last data value. This effectively doubles the magnitude of the deadband compared to other OPC servers. To determine how your specific server handles deadband, refer to the documentation of your OPC server. **Example:** Suppose the engineering units are 0 to 200. Suppose the deadband value is 10%, which is 20 units. If the deadband value is 10% and the last reported value is 50, the value will be reported when the current value exceeds 50 + 10 = 60 or is less than 50 - 10 = 40. Note that the deadband (20 units) is split around the last data value (10 on either side.) Alternatively, you could specify an absolute deadband of 5. In this instance, if the last value was 50, a new data sample will be reported when the current value exceeds 55 or drops below 45.

Field	Description
	If compression is enabled and the deadband is set to zero, the collector ignores data values that do not change and records any that do change. If you set the deadband to a non-zero value, the collector records any value that lies outside the deadband. If the value changes drastically, a pre-spike point may be inserted. For information, refer to Enable Spike Logic *(on page 473)*.
Engineering Unit	Converts the deadband percentage into engineering units and displays the result. This value establishes the deadband range that is centered around the new value. This field represents a calculated number created to give an idea of how large a deadband you are creating in engineering units. The deadband is entered in percentage and Historian converts the percentage in to engineering units.
Collector Compression Timeout	Indicates the maximum amount of time the collector will wait between sending samples for a tag to the archiver. This time is maintained per tag, as different tags report to the archiver at different times. For polled tags, this value should be in multiples of your collection interval. After the timeout value is exceeded, the tag stores a value at the next scheduled collection interval, and not when the timeout occurred. For example, if you have a 10-second collection interval, a 1-minute compression timeout, and a collection that started at 2:14:00, if the value has not changed, the value is logged at 2:15:10 and not at 2:15:00. For unsolicited tags, a value is guaranteed in, at most, twice the compression timeout interval. A non-changing value is logged on each compression timeout. For example, an unsolicited tag with a 1-second collection interval and a 30-second compression timeout is stored every 30 seconds. A changing value for the same tag may have up to 60 seconds between raw samples. In this case, if the value changes after 10 seconds, then that value is stored, but the value at 30 seconds (if unchanged) will not be stored. The value at 60 seconds will be stored. This leaves a gap of 50 seconds between raw samples which is less than 60 seconds.

Field	Description
	Compression timeout is supported in all collectors except the PI collector.
Archive Compression	Indicates whether archive compression *(on page 423)* is enabled. If enabled, Historian applies the archive deadband settings against all reported data from the collector.
Archive Deadband	The current value of the archive deadband, expressed as a percent of span or an absolute number.
Engineering Unit	Converts the deadband percentage into engineering units and displays the result. This value establishes the deadband range that is centered around the new value.
Archive Compression Timeout	The maximum amount of time from the last stored point before another point is stored, if the value does not exceed the archive compression deadband. The data archiver treats the incoming sample after the timeout occurs as if it exceeded compression. It then stores the pending sample.

Table 69. The Advanced Section

Field	Description
Time Assigned By	The source of the timestamp for a data value is either the collector or the data source. All tags, by default, have their time assigned by the collector. When you configure a tag for a polled collection rate, the tag is updated based on the collection interval. For example, if you set the collection interval to 5 seconds with no compression, then the archive will be updated with a new data point and timestamp every 5 seconds, even if the value is not changing. However, if you set the **Time Assigned By** field to **Source** for the same tag, the archive only updates when the device timestamp changes. For example, if the poll time is still 5 seconds, but if the timestamp on the device does not change for 10 minutes, no new data will be added to the archive for 10 minutes.

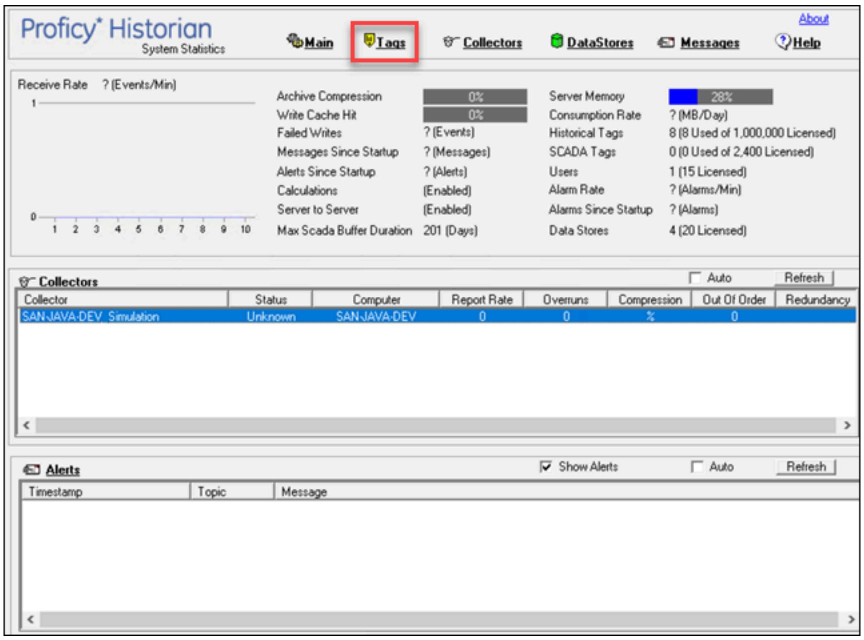
Field	Description
	Note: This field is disabled for Calculation and Server-to-Server tags.
Time Zone Bias	The number of minutes from GMT that should be used to translate time-stamps when retrieving data from this tag. For example, the time zone bias for Eastern Standard time is -300 minutes (GMT-5). This field is not used during collection. Use this option if a particular tag requires a time zone adjustment during retrieval other than the client or server time zone. For example, you could retrieve data for two tags with different time zones by using the tag time zone selection in the iFIX chart.
Time Adjustment	If the Server-to-Server collector is not running on the source computer, select the **Adjust for Source Time Difference** option to compensate for the time difference between the source archiver computer and the col-lector computer. **Note:** This field only applies to tags associated with the Server-to-Server collector that use a polled collection type.
Data Store	Displays the data store to which the tag belongs.
Read Group	The Windows security group assigned to the selected tag.
Write Group	The Windows security group assigned to the selected tag.
Administer Group	The Windows security group assigned to the selected tag.
Last Modified	The date the last tag parameter modification was made.
Modified By	The name of the person who last modified the tag configuration para-meters.

Add Tags from Source

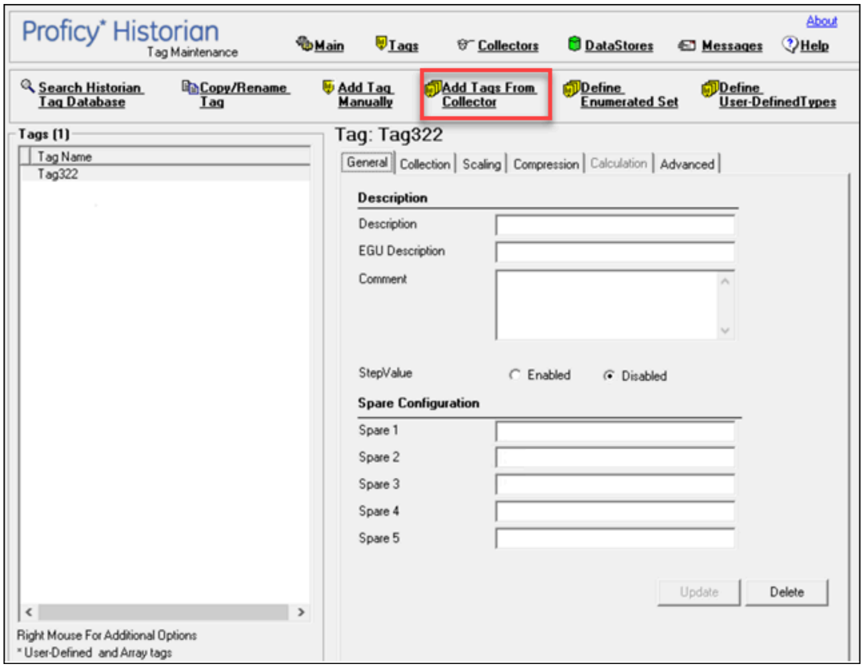
- Ensure that you are a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).
- Create a collector instance *(on page 44)* using which you want to browse the source for tags.

This topic describes how to browse for source tags and add them to Historian. These tags are then created automatically in the Historian database. You can also create tags manually *(on page 443)*.

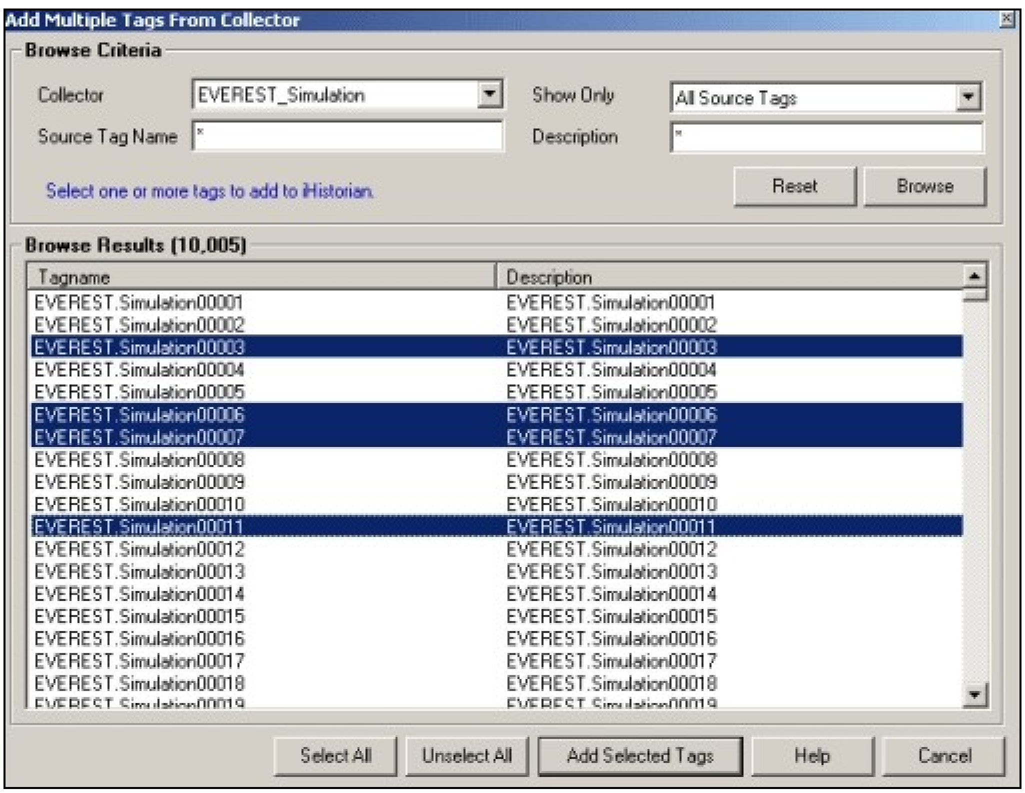
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select **Add Tags from Collector**.



The **Add Multiple Tags from Collector** window appears.



4. Enter values as described in the following table.

Field	Description
Collector	Select the collector instance using which you want to browse the source for tags.
Show Only	Specify whether you want to see all tags or only the ones that have not been added yet.
Source Tag Name	Enter the string to narrow down the search results based on the tag name. You can use wildcard characters.
Description	Enter the string to narrow down the search results based on the tag description. You can use wildcard characters.

5. Select **Browse**.
   A list of tags based on the search criteria appear.
6. Select the tags that you want to add to Historian.
   - Select a single tag by selecting the name of the tag.
   - Select multiple tags by pressing the **Control** key and selecting the tags.
   - Select a contiguous group by pressing the **Shift** key and selecting the first and last tag of the group.
   - Select all tags by selecting **Select All**.
7. Select **Add Selected Tags**.
   The selected tags are added to the Historian database.

Create a Tag Manually

You must be a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).

This topic describes how to create a tag manually. You can also add tags from source *(on page 441)*; these tags are then automatically created in the Historian database.

Whenever you add tags, delete tags, or modify certain tag properties, the following collectors reload only the modified tags without restarting the collectors.

- OPC Collector
- iFIX Collector
- Simulation Collector
- Server-to-Server Collector

- OSI PI Collector
- OSI PI Distributor

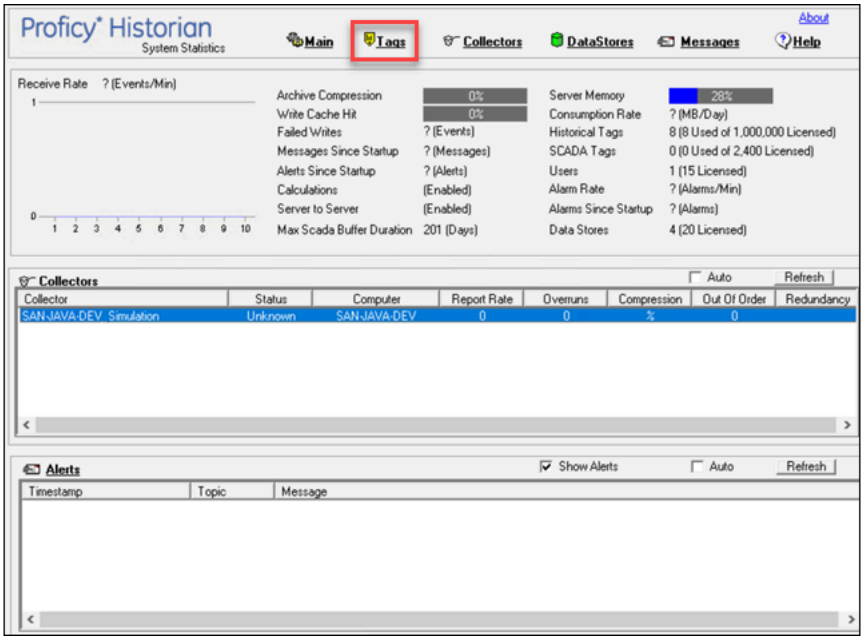
The dynamic collector update feature ensures that any modifications to the tag configuration do not affect all the tags in a collector. Tags that stop data collection may record zero data and bad quality without restarting the collector. Tags that do not stop data collection do not record bad data samples to the collection.

By default, the On-line Tag Configuration Changes option is enabled, which allows a tag to stop and restart data collection without restarting the collector. If you disable the On-line Tag Configuration Changes option, any changes you make to the tags do not affect collection until after you restart the collector. To enable or disable the On-line Tag Configuration Changes option, select **Advanced** on the **Collector Maintenance** page.

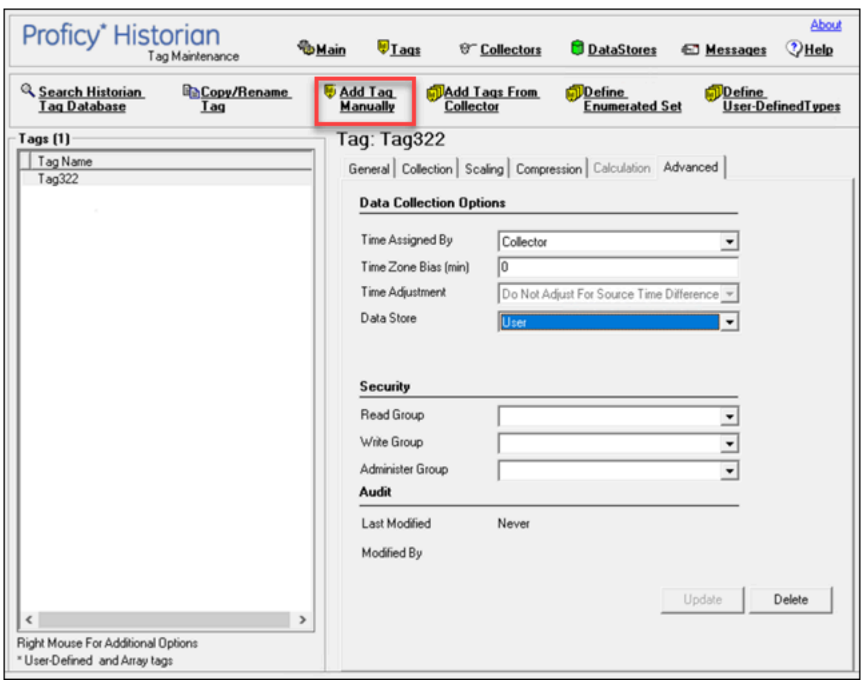
To restart the collector you must stop and start the collector service or executable. Restarting the collector stops and restarts the tag(s) collection and may record bad data samples to the collection. All the collector configuration changes done within a 30 second time frame are batched up together. To collect the modified data faster, update/modify a small set of tags at a time.

> **Note:**
> When updating large sets of tags at the same time, best practice is to disable the On-line Tag Configuration Changes option and restart the collector after modification.

1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select **Add Tag Manually**.



The **Add Tag Manually** window appears.

4. Enter values as described in the following table.

Field	Description
Collector Name	Select the collector using which you want to collect data for the tag.
Source Address	Enter the source address for the tag.
Tag Name	Enter a unique name for the tag.
Data Store	Select the data store in which you want to store the tag data.
Data Type	Select the data type of the tag data.
Is Array Tag	Not applicable
Time Resolution	Select the duration at which you want to collect data for the tag. For example, if you select **Seconds**, data is collected every second.

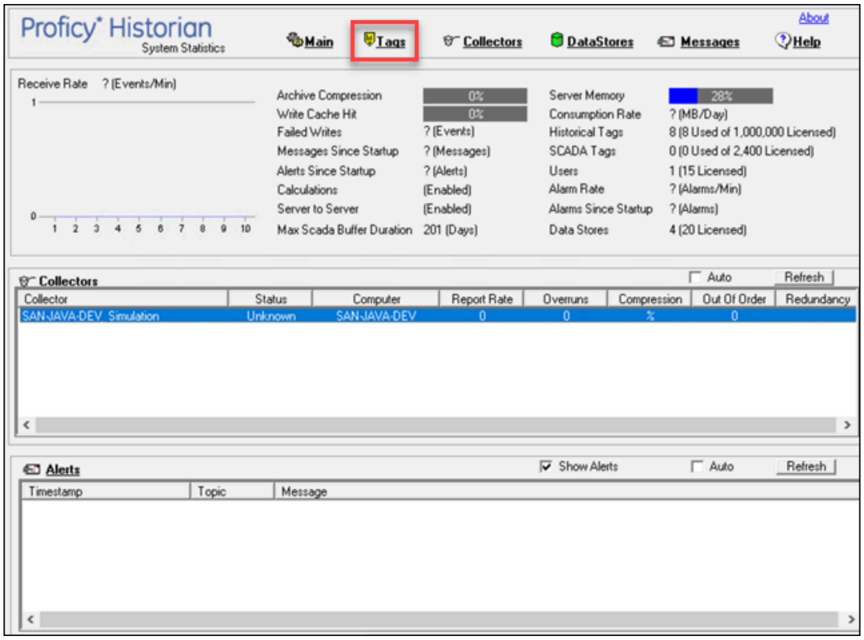
Note:
If you add a tag for a Server-to-Server collector, set the **Time Adjustment** field for the tag to **Adjust for Source Time Difference** after you add the tag. This field is available under **Tags > Advanced**. This is applicable only for polled data collection.

5. Select **OK**.
   The tag is created.

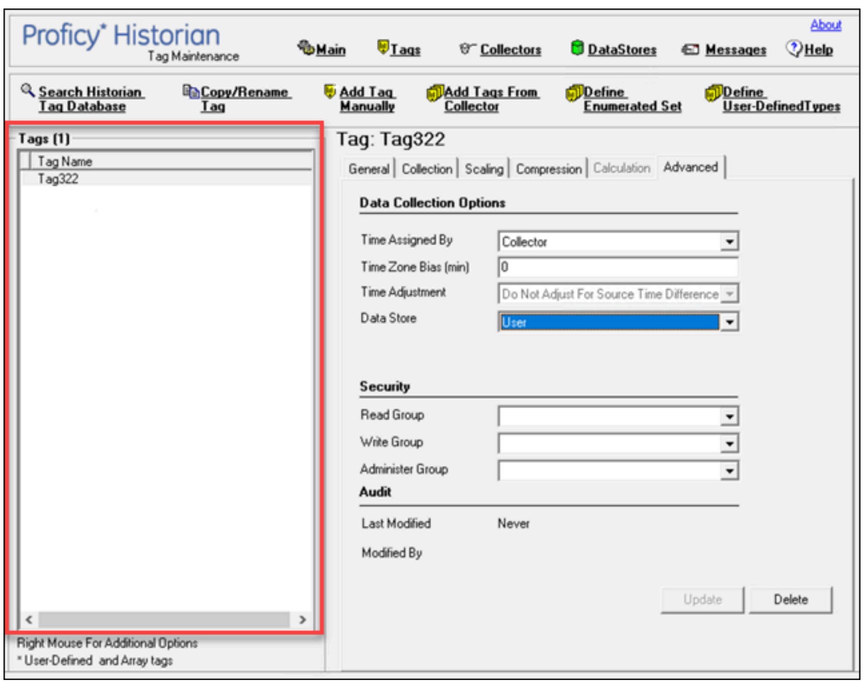
Copy a Tag

You must be a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).

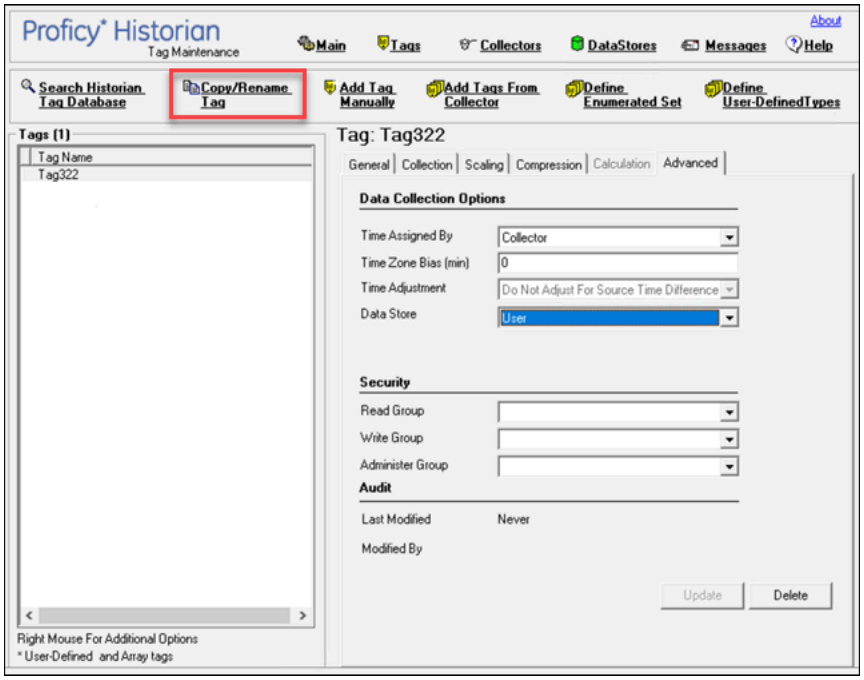
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select the tag that you want to copy.



4. Select **Copy/Rename Tag**.



The **Copy/RenameTag** window appears.

5. Select **Copy**.
6. Enter a new tag name.
7. Select **OK**.

   The tag is copied.

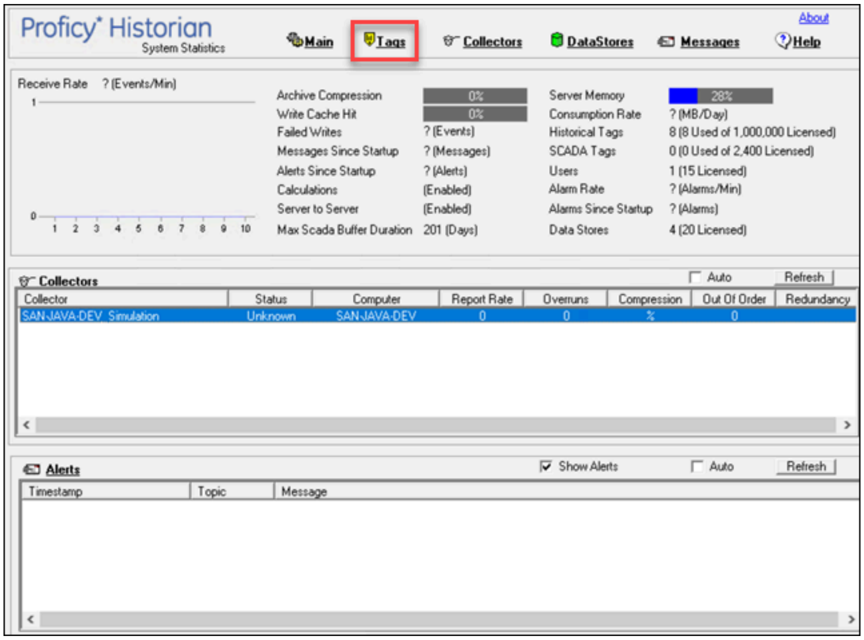
Rename a Tag

- You must be a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).
- If you want to rename a tag permanently, to avoid loss of data, stop the collector instance.

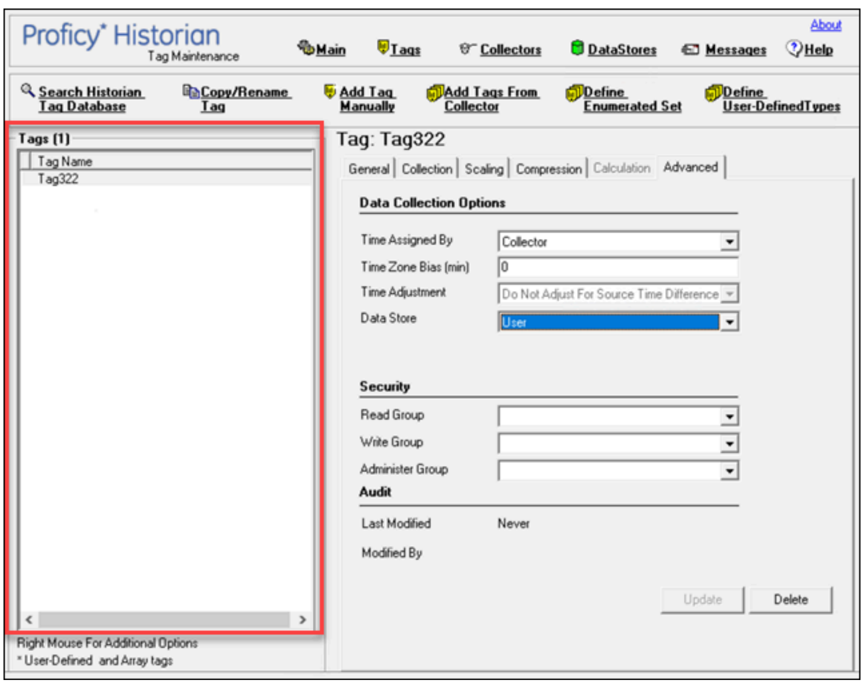
When you rename a tag, you can choose between the following options:

- **Rename using an alias:** In this case, the old name is called the tag alias. You can retrieve tag data using the tag alias as well. When you copy a tag, the tag alias is captured as well to aid in an audit trail.
- **Rename permanently:** In this case, the old name is no longer captured. Therefore, you can create another tag with this old name. You cannot store and forward data using the old name. This implies that data for the tag is collected separately for the new name.

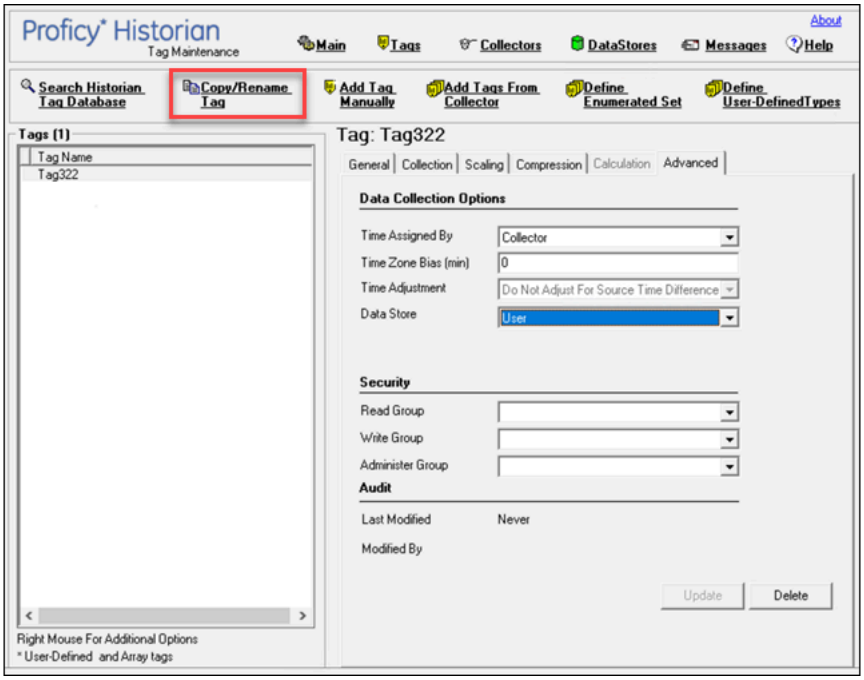
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select the tag that you want to rename.



4. Select **Copy/Rename Tag** .



The **Copy/RenameTag** window appears.

5. If you want to rename the tag using an alias, select **Rename (Alias)**. If you want to rename the tag permanently, select **Permanent Rename**.
6. Select **OK**.

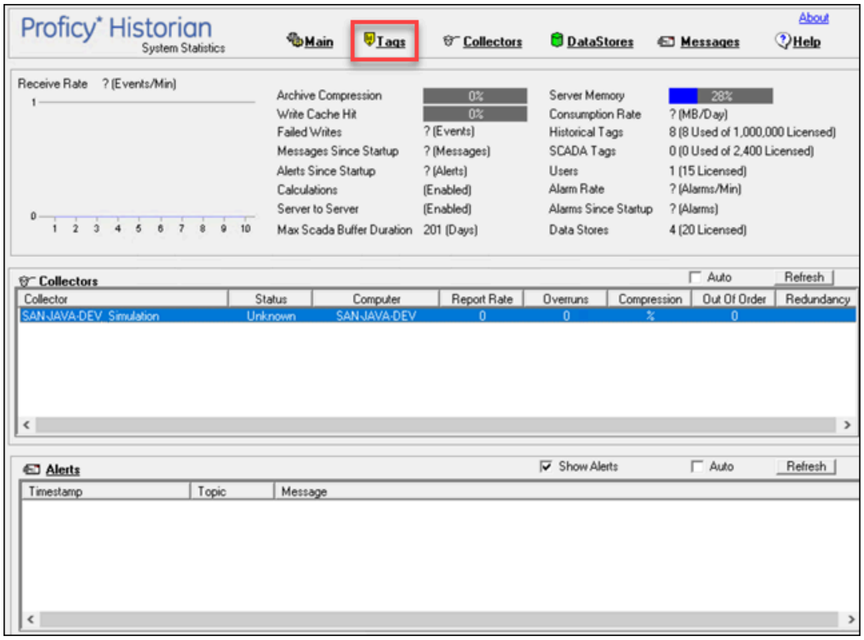
If you have renamed the tag permanently:

- If the tag is used as a trigger, reassign the trigger.
- Restart the collector instance.

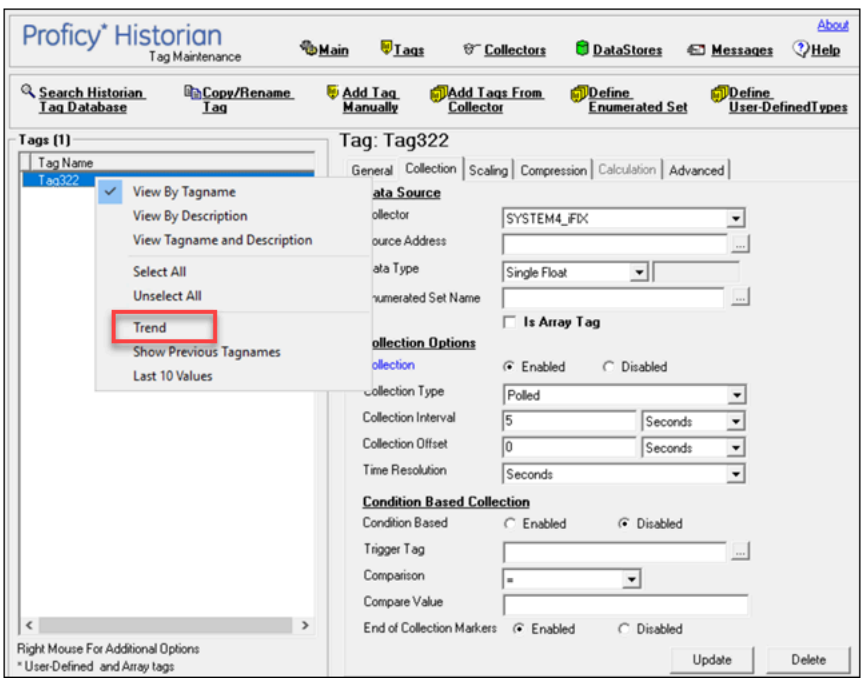
View Tag Trends and Raw Data

This topic describes how to access the trend chart of tag data. Note that the tag trend should not be used for detailed data.

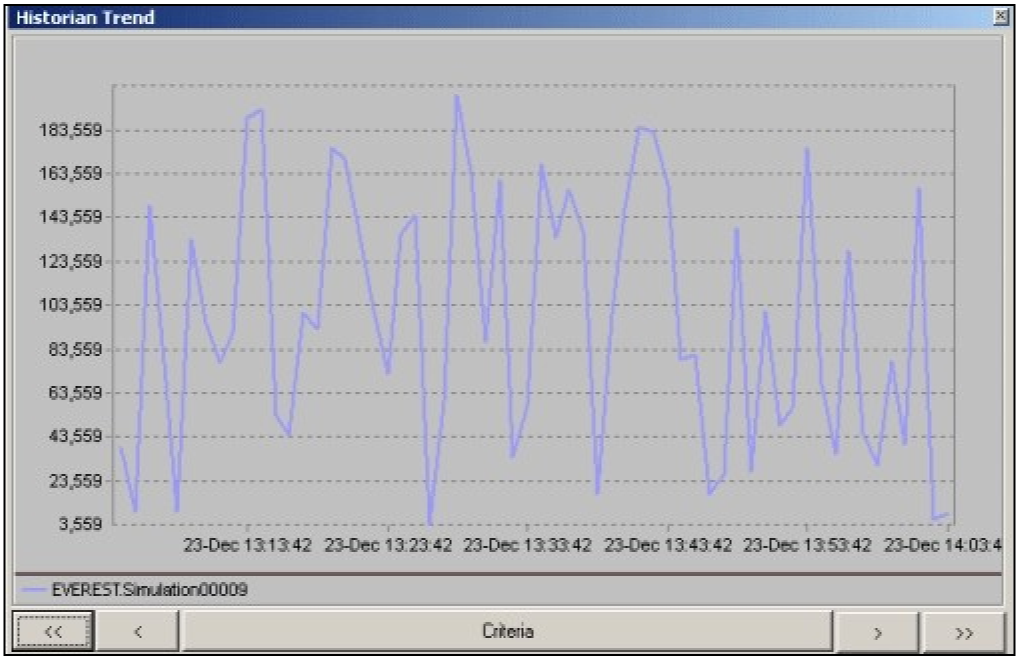
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Right-click the tag whose trend chart you want to access, and then select **Trend**.



The trend chart of the tag values appears.



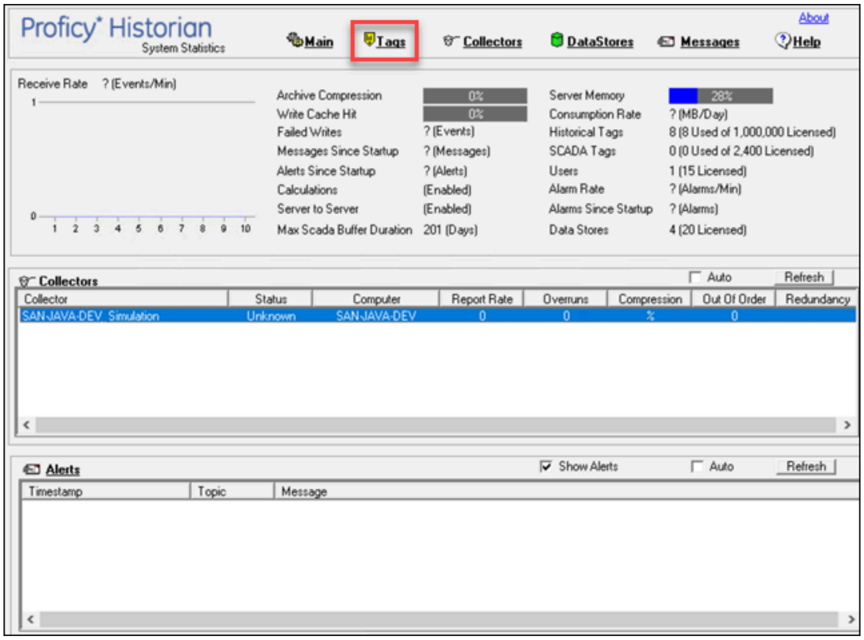
Note:
To change the criteria, select **Criteria**, and then enter values as described in the following table.

Field	Description
Start Time	Select the start time of the trend chart.
End Time	Select the end time of the trend chart.
Sampling	Select the data type that you want to use.
Interval	Enter the interval at which you want to plot the tag data.
Criteria Strings	Enter the sampling mode, calculation mode, and/or query modifiers. Query modifiers are used to specify various ways of retrieving data from Historian. For example, you can request raw data with good quality only by specifying the criteria string as: `RAWBYTIME#ONLYGOOD`. The sampling mode specified with criteria strings takes precedence over the mode specified in the **Sampling** field.

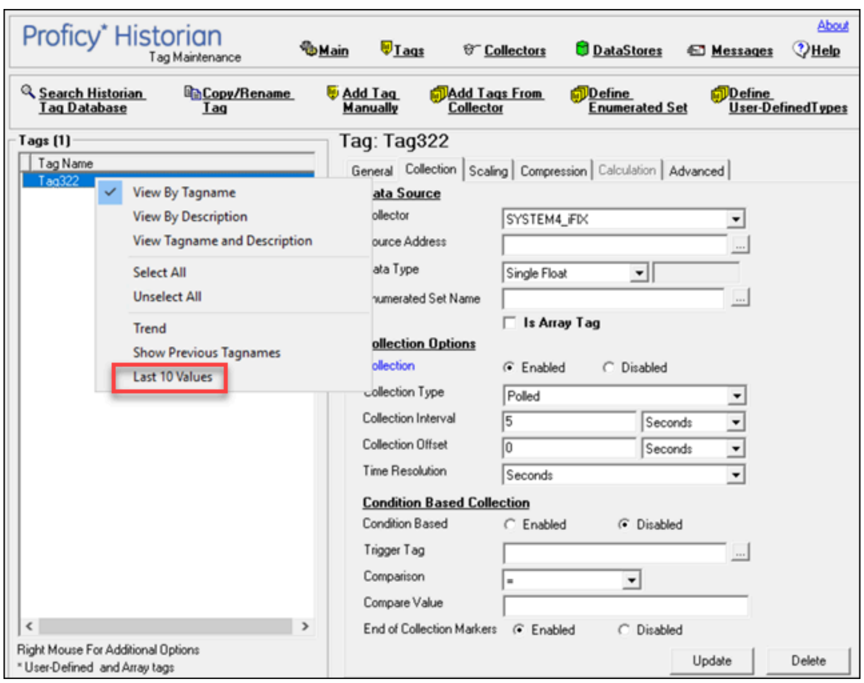
You can also scroll back and forth on the x-axis time scale by selecting on the single and double left and right arrows at the bottom of the page.

View the Last 10 Raw Values of a Tag

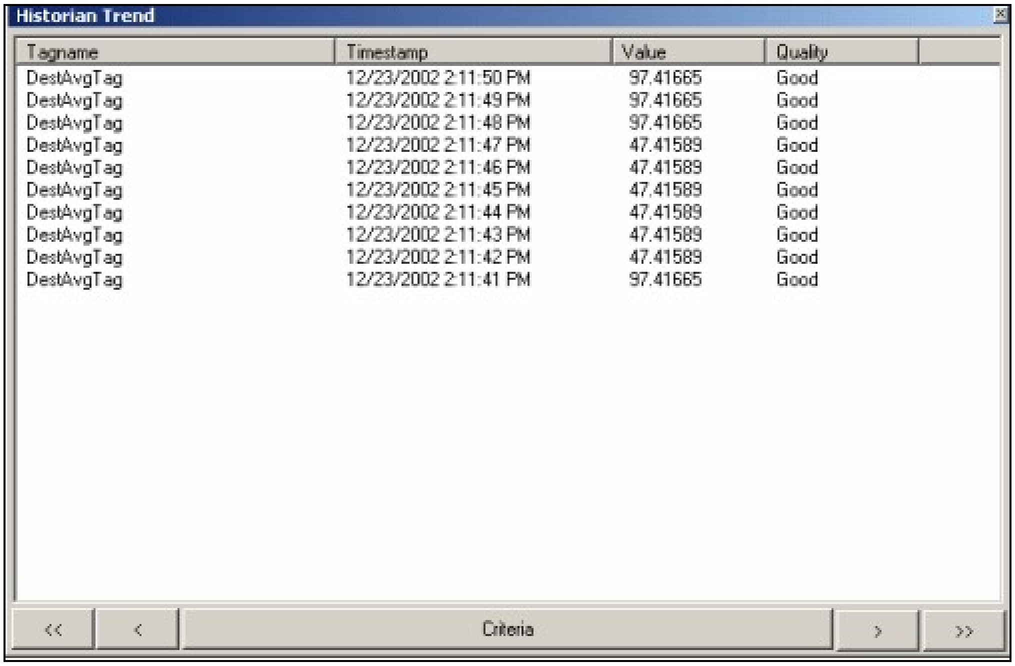
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Right-click the tag whose last 10 values you want to access, and then select **Last 10 Values**.

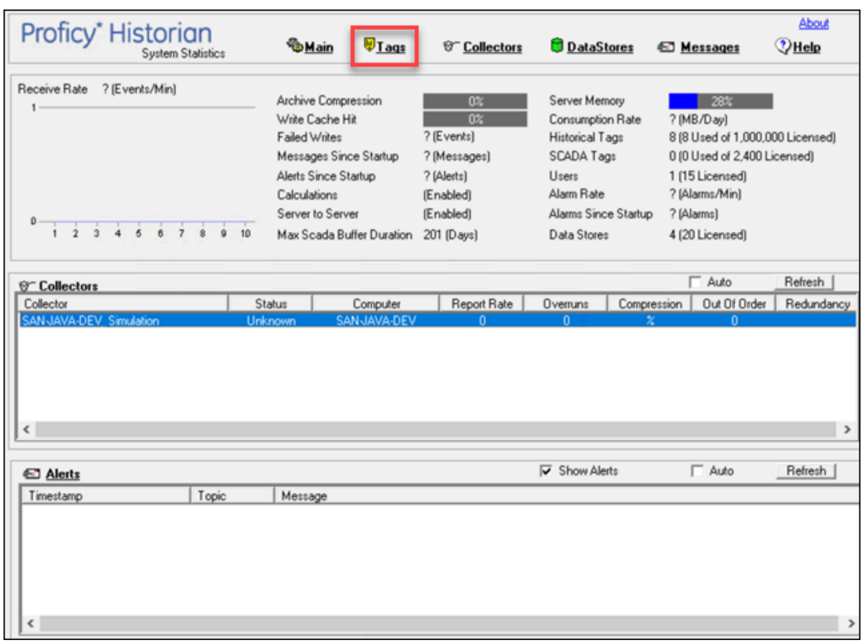


The last 10 values of the tag appear.

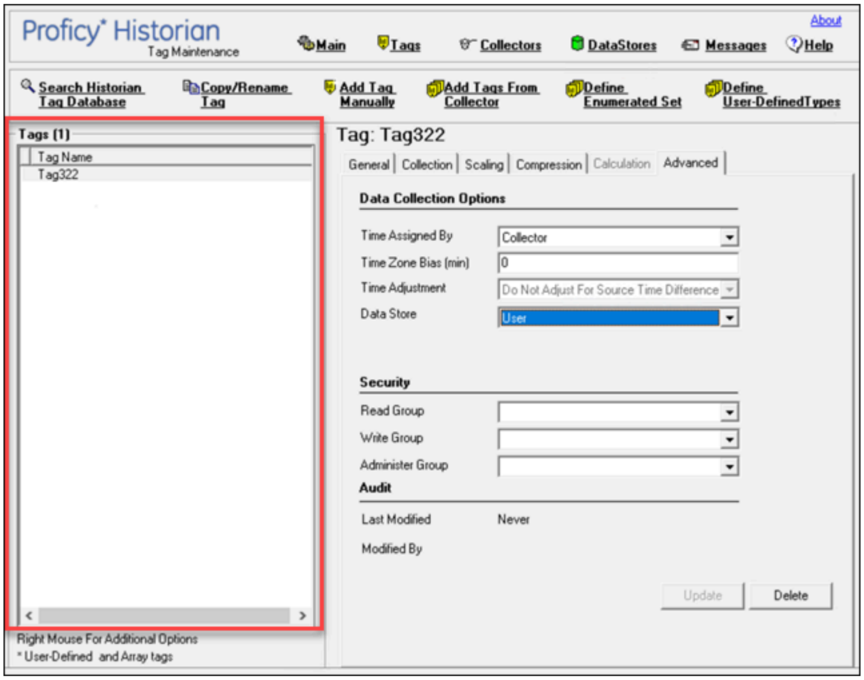


Stop Data Collection

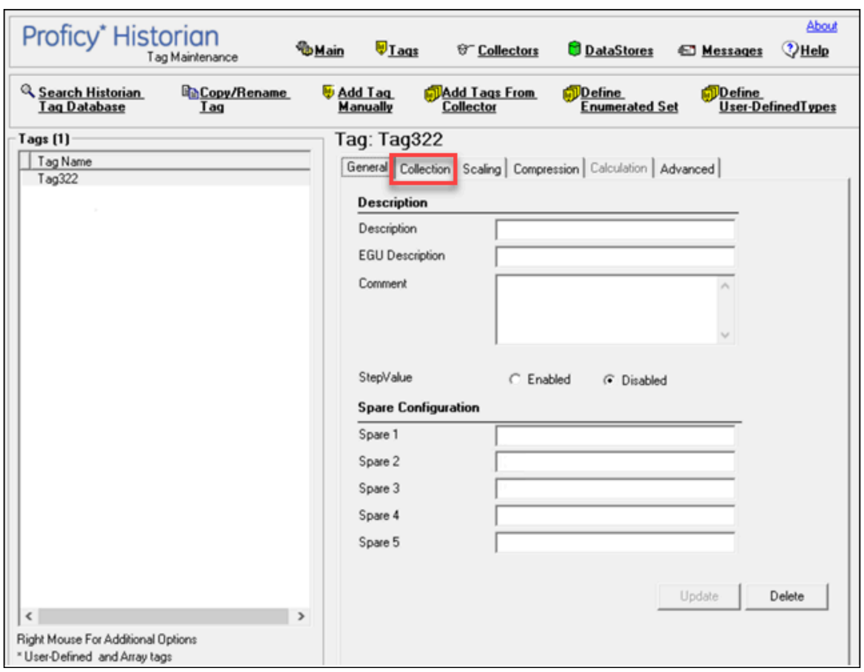
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



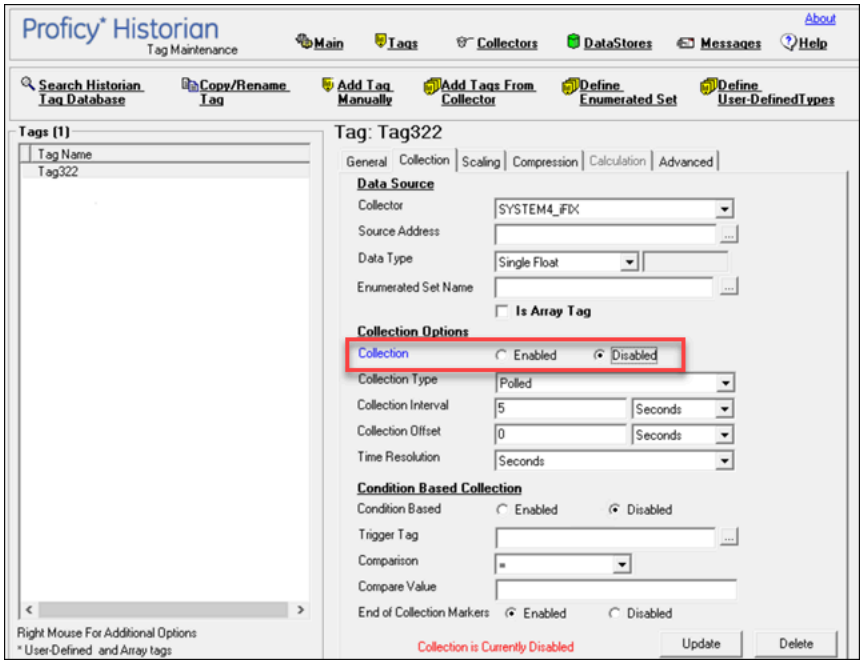
3. Select the tag whose data collection you want to stop.



4. Select **Collection**.

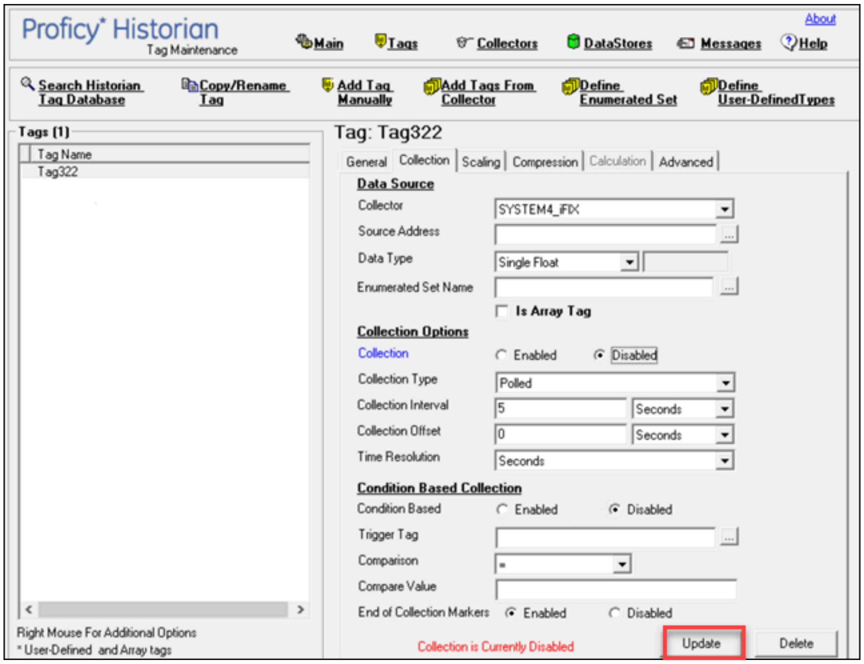


5. For the **Collection** field, select **Disabled**.



6. Select **Update**.
7. To stop data collection on a tag:
    a. From the list in the left-hand window of the page, select a tag.
    b. In the window on the right side of the page, select **Collection**.
    c. For the **Collection** field, select the **Disabled** option.

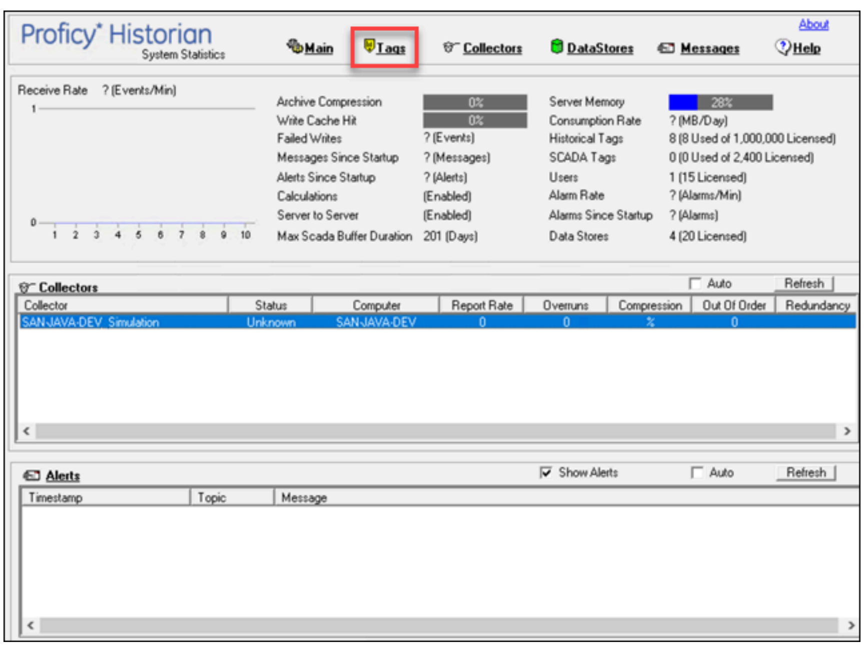
d. Select **Update**.



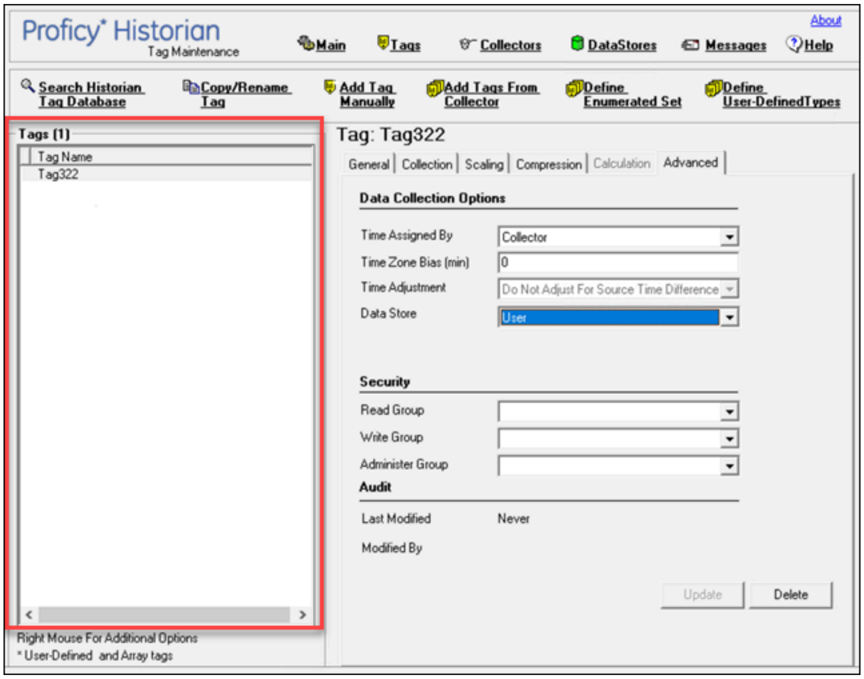
The data collection for the tag is stopped.

Resume Data Collection

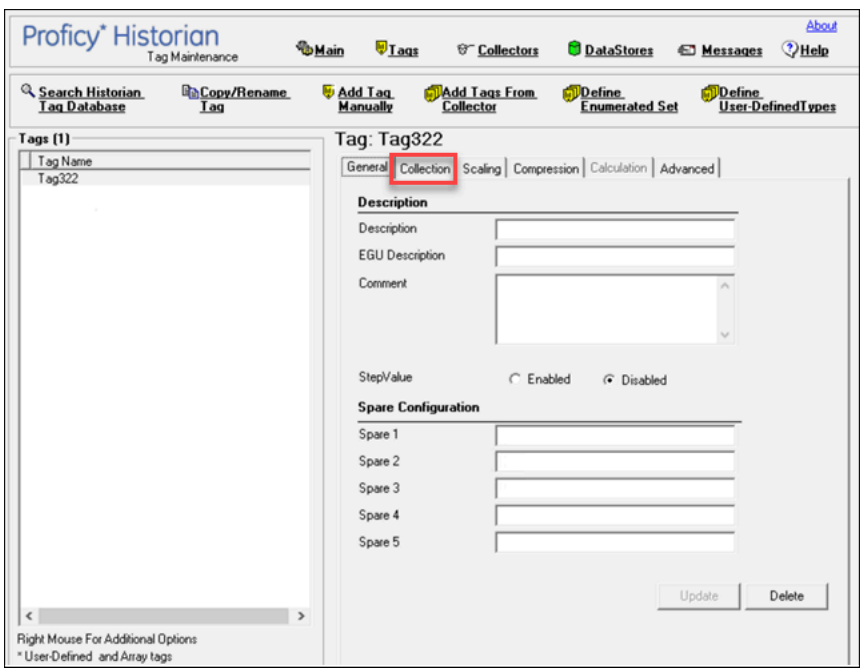
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



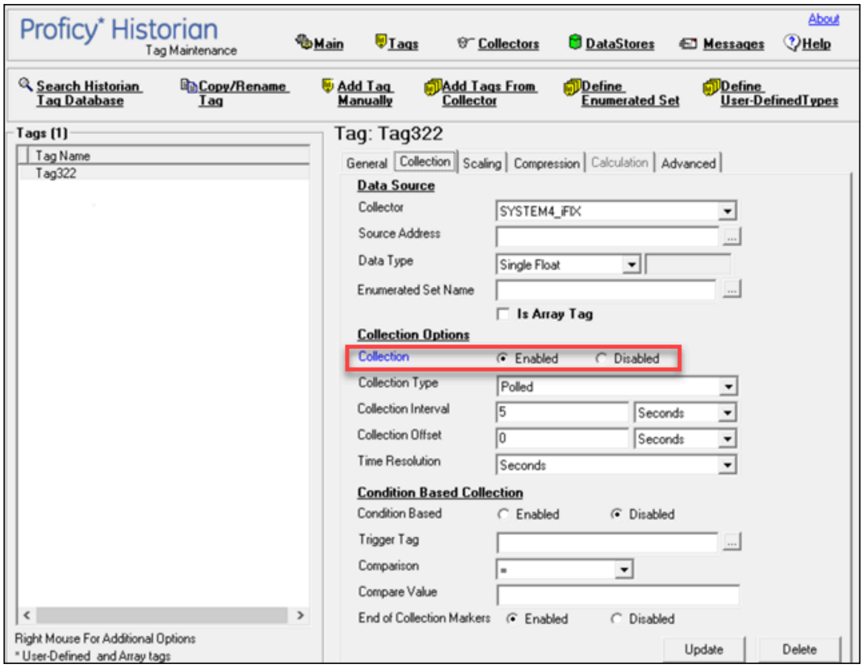
3. Select the tag whose data collection you want to resume.



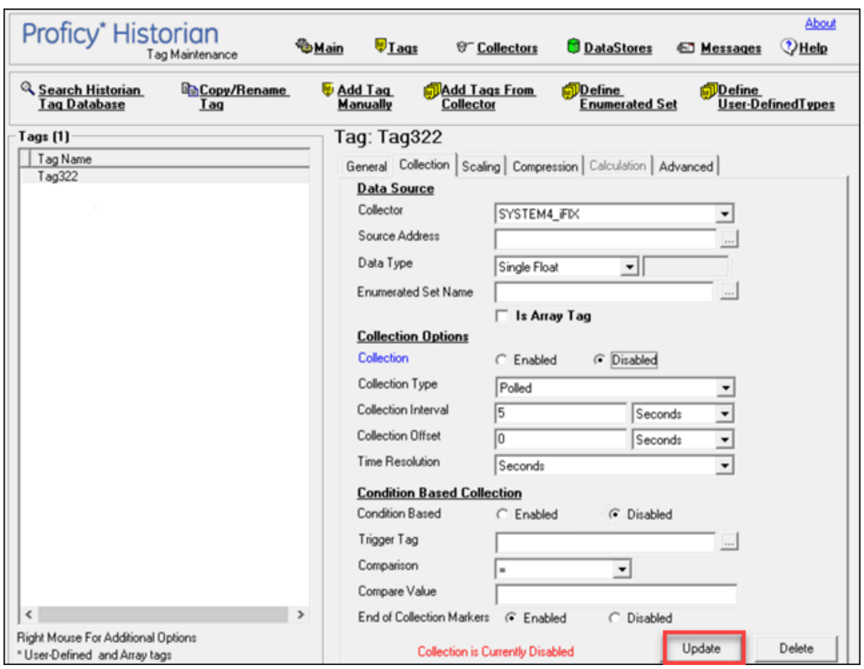
4. Select **Collection**.



5. For the **Collection** field, select **Enabled**.



6. Select **Update**.



The data collection for the tag is resumed.

Get all the Fields Related to a Tag

1. Access Registry Editor.
2. Create a DWORD (32-bit) registry entry named GetAllTagProps for the collector in the following registry path: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\Services\OPCCollector\*_OPC_Intellution_IntellutionGatewayOPCServer`
3. Provide the value 1 for the registry entry.

Remove A Tag

When you remove a tag, it is removed from the tag database, but all data for that tag is retained in the archive and the tag name cannot be reused. Since the tag data is still available from the archive, you can still reference that tag from within a calculation formula, for example, or by using the Excel Add-In.

If, however, you want to remove the tag data as well from the archive, you can delete it permanently *(on page 463)*.

Whenever you delete/remove tags, the following collectors reload only the modified tag(s) without restarting the collectors.

- OPC Collector
- iFIX Collector
- Simulation Collector
- Server-to-Server Collector
- OSI PI Collector

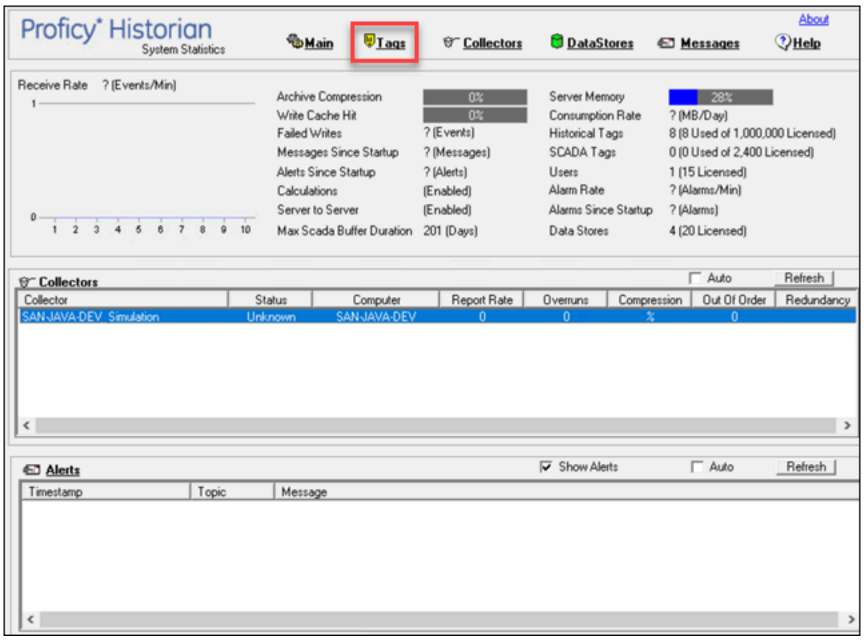
By default, the On-line Tag Configuration Changes option is enabled, which allows a tag to stop and restart data collection without restarting the collector. If you disable the On-line Tag Configuration Changes option, any changes you make to the tags do not affect collection until you restart the collector. To enable or disable the On-line Tag Configuration Changes option, access Historian Administrator, and then select **Collectors > Advanced**.

Restarting the collector stops and restarts the tag(s) collection and may record bad data samples to the collection. All the collector configuration changes done within a 30-second time frame are batched up together. To collect the modified data faster, update/modify a small set of tags at a time. If the modified tags get zero bad markers and available runtime values at the same time, then precedence is given to available runtime values instead of zero bad markers.

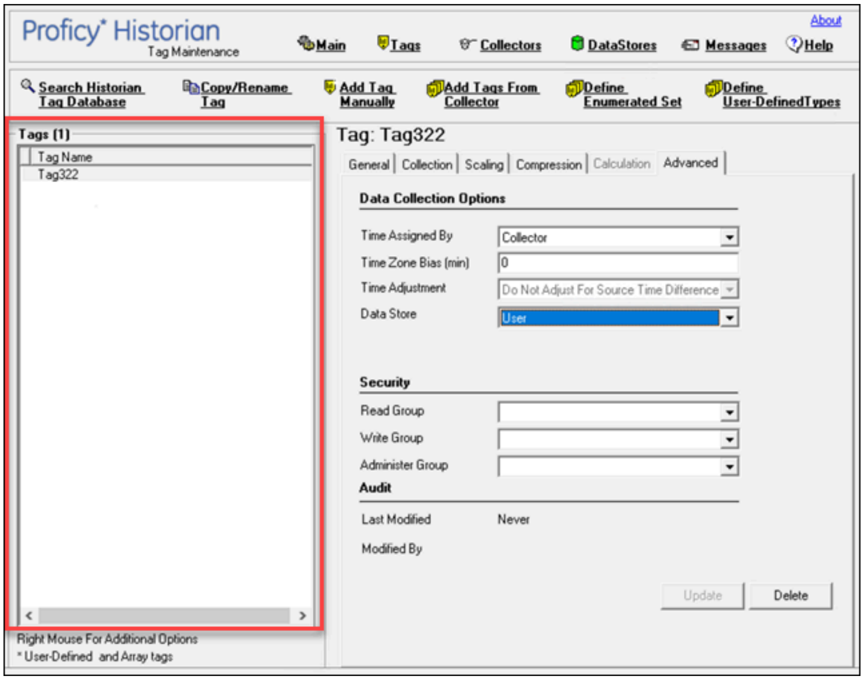
Tip:

- To collect the modified data faster, update/modify a small set of tags at a time.
- When updating large sets of tags at the same time, the best practice is to disable the **On-line Tag Configuration Changes** option and restart the collector after modification.

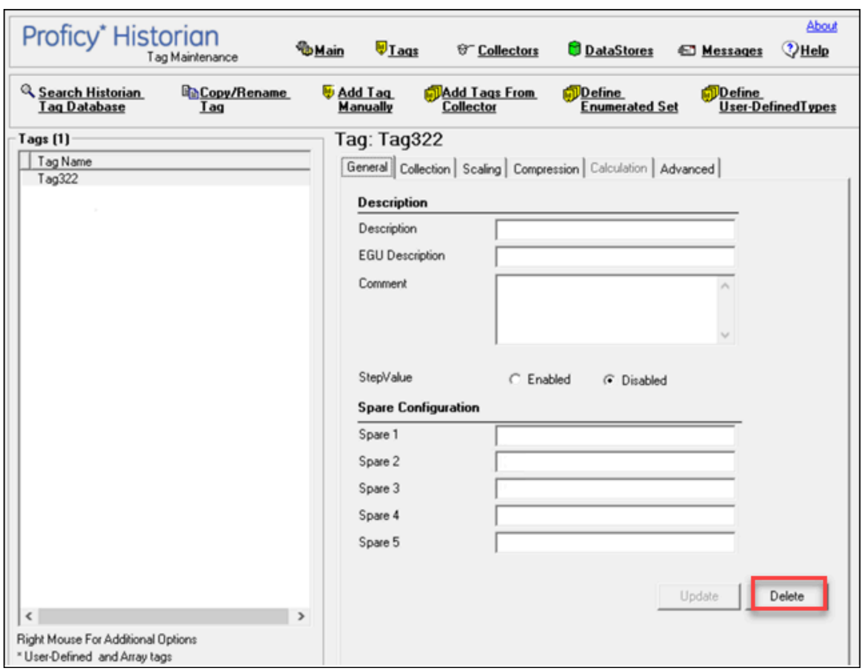
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select the tag that you want to remove.



4. Select **Delete**.



The **Delete Tags** window appears.

5. Select **Remove Tag from System**, and then select **OK**.

   A message box appears, asking you to confirm that you want to remove the tag.

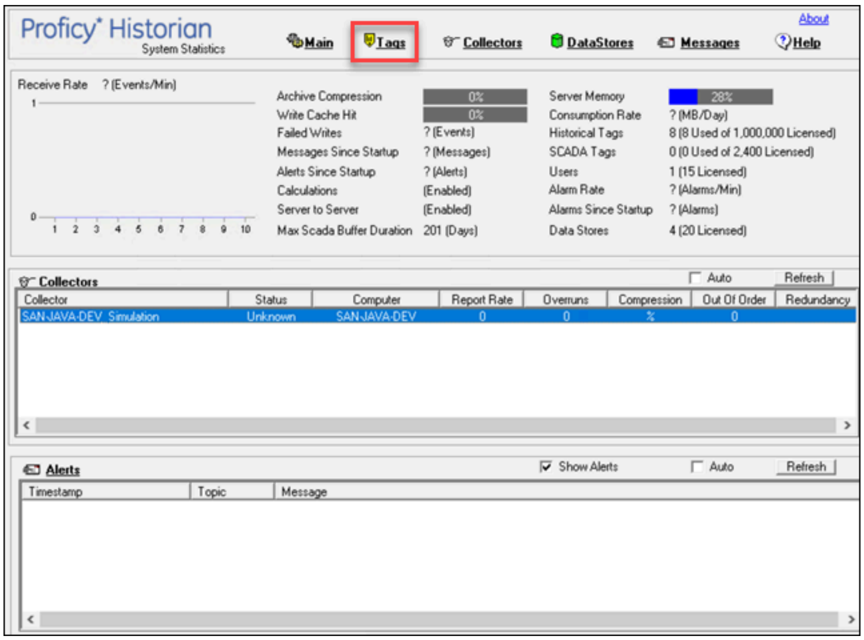
6. Select **Yes**.

   The tag is removed.

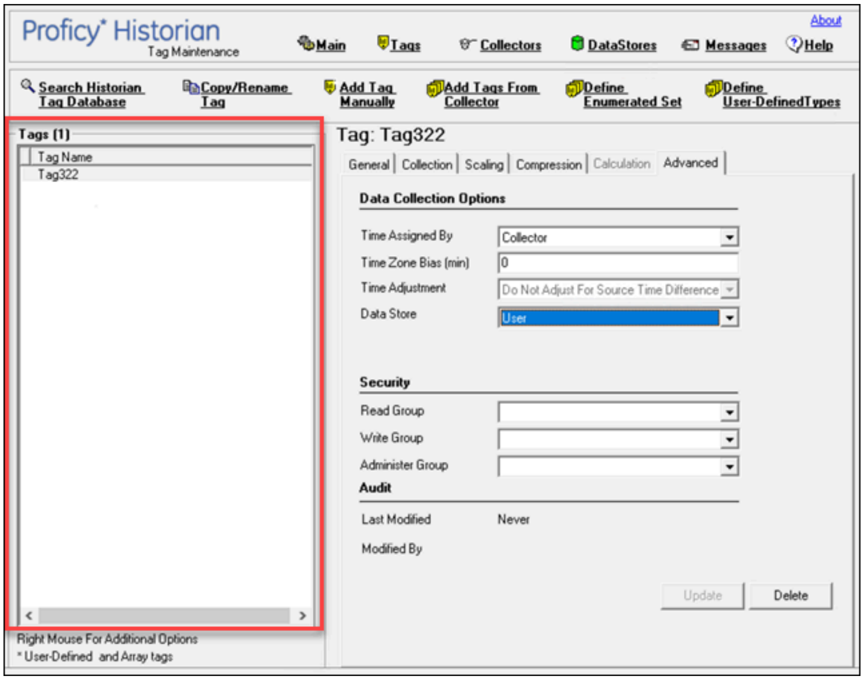
Deleting Tags Permanently

When you delete a tag, the tag as well as all the data for that tag is removed from the archive and the tag name is available for reuse. You can no longer query the data for that tag. If, however, you want to just remove the tag, but retain the tag data, refer to Remove A Tag *(on page 460)*.

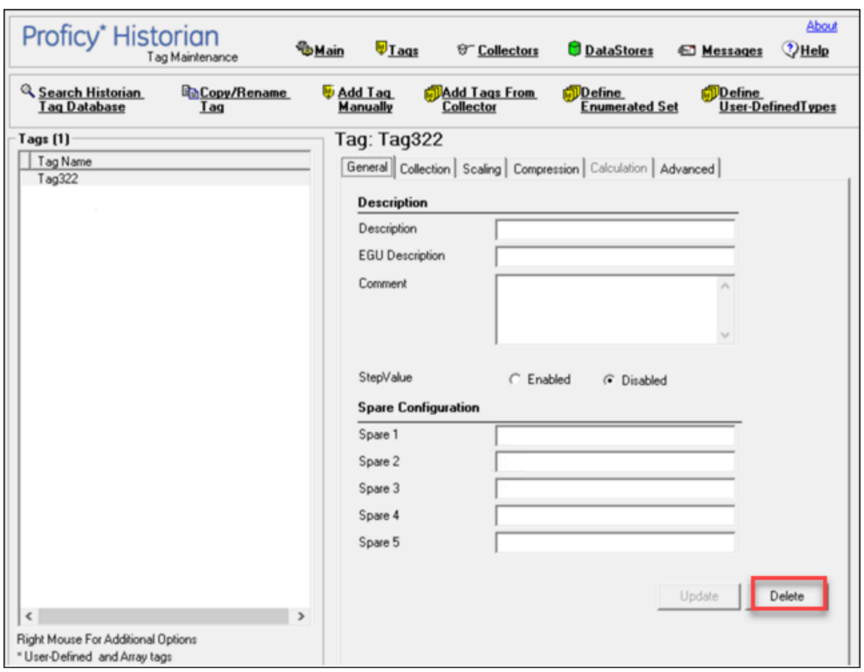
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select the tag that you want to delete.



4. Select **Delete**.



The **Delete Tags** window appears.

5. Select **Permanently Remove Tags From System**, and then select **OK**.

   A message box appears, asking you to confirm that you want to delete the tag.

6. Select **Yes**.

   The tag is deleted.

Managing Collectors

About Collectors

A collector collects tag data from various data sources.

How tag data is stored if using collectors of on-premises Proficy Historian (TLS encryption is not used):

1. Collectors send a request to the Azure Load Balancer to write tag data.
2. Azure Load Balancer sends the request to HA Proxy. HA Proxy routes the traffic to Data Archiver. If user authentication is needed, the Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, Load Balancer confirms to the collectors that data can be sent.
3. Data collected by the collector instances is sent to the Azure Load Balancer.
4. Azure Load Balancer sends the data to HA Proxy and HA Proxy again routes the traffic to Data Archiver. After authentication, the Data Archiver stores the data in the Azure File Share in .iha files.

How tag data is stored if using Historian Collectors for Cloud (TLS encryption is used):

1. Collectors send a request to the Azure Load Balancer to write tag data. Since the request is encrypted, port 443 is used.
2. Azure Load Balancer forwards the request to HA Proxy. HA Proxy decrypts the request and sends it to the Data Archiver. If user authentication is needed, the Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, the Azure Load Balancer confirms to the collectors that data can be sent.
3. Data collected by the collector instances is encrypted and sent to the Azure Load Balancer using port 443. The Azure Load Balancer forwards request to HA Proxy.
4. HA Proxy decrypts the data and sends it to the Data Archiver. After authentication, the Data Archiver stores the data in the Azure File Share in .iha files.

How data is retrieved:

1. Clients (that is, the Excel Addin, the Web Admin console, the REST Query service, or Historian Administrator) send a request to the Azure Load Balancer to retrieve data.
2. The Azure Load Balancer sends the request to HA Proxy, and then HA Proxy forwards requests to the Data Archiver, which retrieves data from Azure File Shares. If, however, user authentication is needed, the Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, data is retrieved from Azure File Share.

To send data using a collector, you must:

1. Install collectors (on page 29).

   You can install collectors on multiple Windows machines. These machines can be on-premises or on an Azure Virtual Network (VNet).

2. Create a collector instance.

Note:
The following collectors are not supported by Proficy Historian for Azure Cloud:

- The File collector
- The HAB collector
- The iFIX Alarms and Events collector
- The OPC Classic Alarms and Events collector
- The Windows Performance collector

What does SQ mean in the cloud output and what are the sub-quality values?

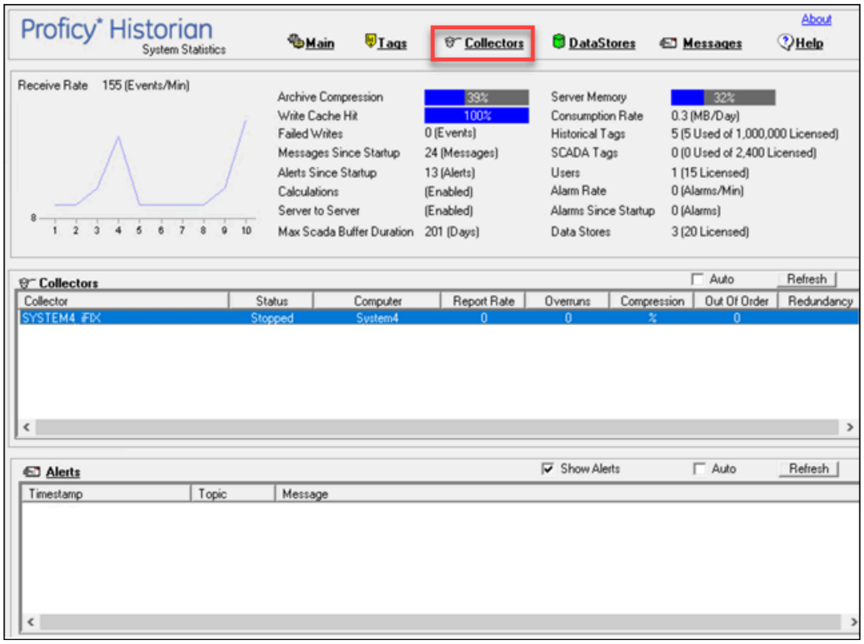
The full form of SQ is Sub Quality; the values range from 1 to 13.

The following are the sub-quality values:

ihOPCNonspecific = 0
ihOPCConfigurationError=1
ihOPCNotConnected=2
ihOPCDeviceFailure=3
ihOPCSensorFailure=4
ihOPCLastKnownValue=5
ihOPCCommFailure=6
ihOPCOutOfService=7
ihScaledOutOfRange=8
ihOffLine=9
ihNoValue=10
ihCalculationError=11
ihConditionCollectionHalted=12
ihCalculationTimeout=13

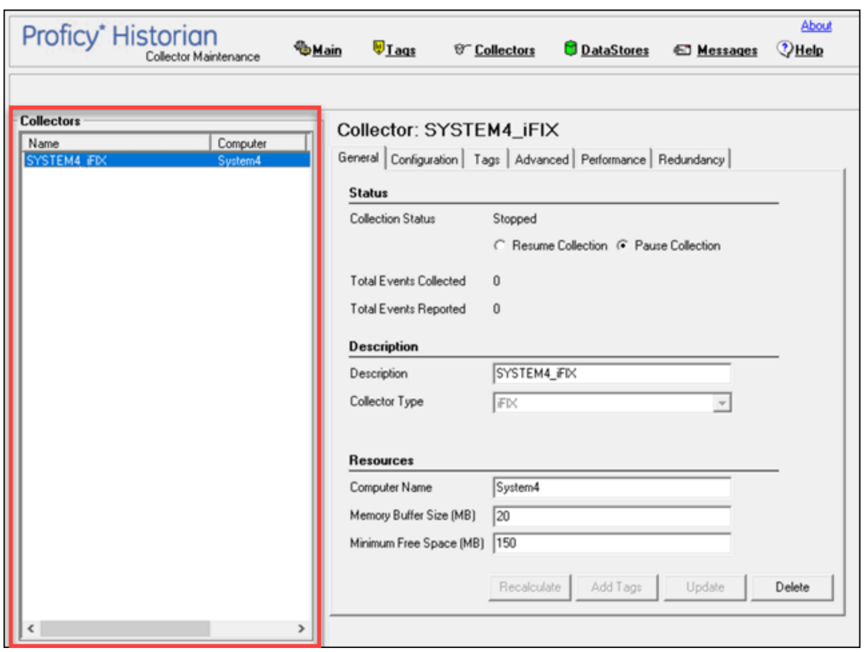
Access/Modify a Collector

1. Access Historian Administrator *(on page 384)*.
2. Select **Collectors**.



A list of collectors appears.

3. Select the collector whose details you want to access/modify.



4. As needed, modify values as described in the following tables, and then select **Update**.

Table 70. The General Section

Field	Description
Collection Status Field	The current operating status of the collector. Contains one of the following values: ◦ **Running:** The collector is operating and collecting data. ◦ **Stopped:** The collector is in pause mode and not collecting data. ◦ **Unknown:** The status information about the collector is unavailable at present, perhaps as a result of a lost connection between collector and server or because the collector was shut down improperly. You can specify whether you want to pause or resume data collection.
Total Events Collected	Not applicable
Total Events Reported	Not applicable
Description	The name of the collector.
Collector Type	The type of the collector.
Computer Name	The machine name of the computer on which the collector is installed.
Memory Buffer Size (MB)	The size of the memory buffer currently assigned to the store-and-forward function. The memory buffer stores data during short-term or momentary interruptions of the server connection; the disk buffer handles long duration outages. To estimate the size you need for this buffer, you need to know how fast the collector is trying to send data to the server and how long the server connection is likely to be down. With those values and a safety margin, you can compute the required size of the buffer. **Note:** If you enter a new value for this parameter, the change is effective the next time you restart the collector.
Minimum Free Space (MB)	The minimum free disk space that must be available on the computer. If the minimum space required is not available when the collector starts, the collector will shut down.

Table 71. The Tags Section

Field	Description
Add Prefix to Tag	A prefix that is automatically added to all tag names when you add tags.
Collection Interval	The time required to complete a poll of a given tag on the collector. It is also used in unsolicited collection. In effect, it specifies how frequently data can be read from a tag. The collection interval can be individually configured for each tag. **Note:** To avoid collecting repeat values with the OPC collector when using device timestamps, specify a collection interval that is greater than the OPC server update rate.
Collection Type	Indicates whether this collector is configured for polled or unsolicited data collection.
Time Assigned By	Indicates whether the timestamp for the data value is provided by the collector or the data source.
Collector Compression	Indicates whether collector compression *(on page 423)* is enabled as a default setting. This option is overridden by tag-level settings.
Deadband	The default setting of the collector compression deadband in absolute or percentage range values.
Compression Timeout	The default setting for the collector compression time-out for tags added through the **Add Multiple Tags From Collector** window. You must enable collector compression to use this field.
Spike Logic Control	Indicates whether incoming data samples for spikes are captured in tag values. If spike logic is enabled, a sample of equal value to the previously archived sample is inserted into the archive in front of the spike value. For more information, refer to Enable Spike Logic *(on page 473)*.

Table 72. The Advanced Section

Field	Description
On-line Tag Configuration Changes	Indicates whether you can make changes to tags without having to restart the collector. If you disable this option, any changes you make to tags do not affect collection until you restart the collector.
Browse Source Address Space	Indicates whether you want the collector to respond to requests to browse the tags in the source. You may sometimes want to disable this feature to reduce processing load on the collector.
Synchronize Timestamps to Server Time	Adjusts all outgoing data timestamps to match the server clock. This option is not applicable when you configure timestamps to be provided by the data source. Note that this does not change collector times to match the server time; it adds or subtracts an increment of time to compensate for the relative difference between the clocks of the server and collector, independent of time zone or day light savings time (DST) differences. If the collector system clock is greater than 15 minutes ahead of the archiver system clock, and the **Synchronize Timestamps to Server** option is disabled, data will not be written to the archive.
Source/Device Timestamps	The time source for the timestamps. This field applies only if you are using source timestamps. The collector uses this field to determine whether the timestamps coming from the data source are in local machine time or UTC.
Delay Collection at Startup	The number of seconds to delay collection on startup (after loading its tag configuration).
Rate Output Address	Not applicable
Status Output Address	Address in the source database into which the collector writes the current value of the collector status, letting an operator or the HMI/SCADA application know the current status of the collector. This address should be connected to a writable text field of at least 8 characters. This value is only updated upon a change in status of the collector. For an iFIX collector, use TX tag for the output address. Enter the address in the following format: NODE.TAG.FIELD (for example, MyNode.MyCollector_TX.A_CV).

Field	Description
	For an OPC collector, use an OPC address in the server. Refer to your OPC documentation for more information.
Heartbeat Output Address	Address in the source database into which the collector writes the heartbeat signal output. This address should be connected to a writable analog field. For an iFIX collector, use an iFIX tag for the output address. Enter the address in the following format: NODE.TAG.FIELD (for example, MyNode.MyCollector_AO.F_CV). For an OPC collector, use the OPC address in the server. Refer to your OPC documentation for more information. The data collector writes the value of 1 to this location every 60 seconds while it is running. You can program the iFIX database to generate an alarm if the Heartbeat Output Address is not written once every 60 seconds, notifying you that the data collector has stopped.

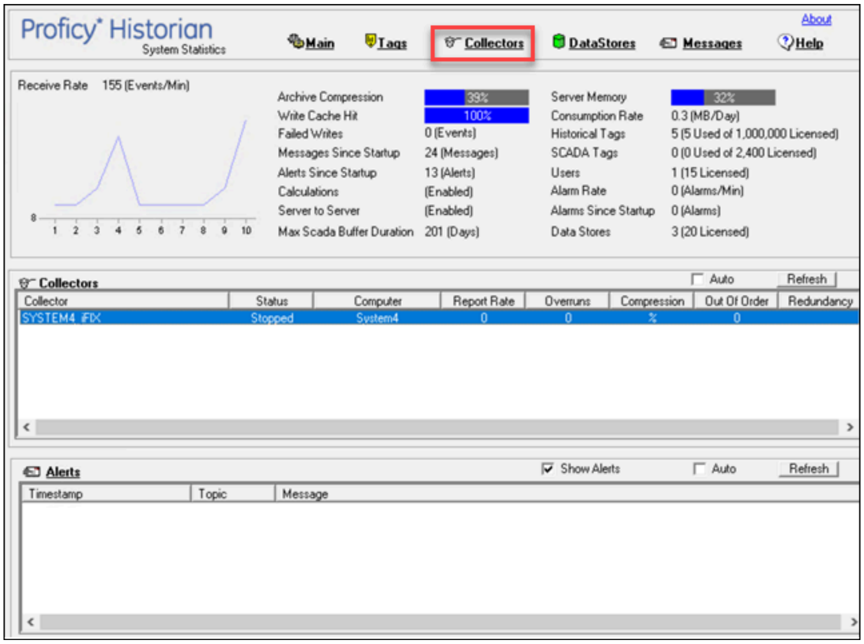
Table 73. The Performance Section

Section	Description
Report Rate	Displays the average rate at which data is coming into the server from the collector. This is a general indicator of load on the Historian collector. Since this chart displays a slow trend of compressed data, it may not always match the instantaneous value of report rate.
Compression	Displays the effectiveness of collector compression. If the chart displays a low current value, you can widen the compression deadbands to pass fewer values and increase the effect of compression.
Overruns	Not applicable

Delete a Collector

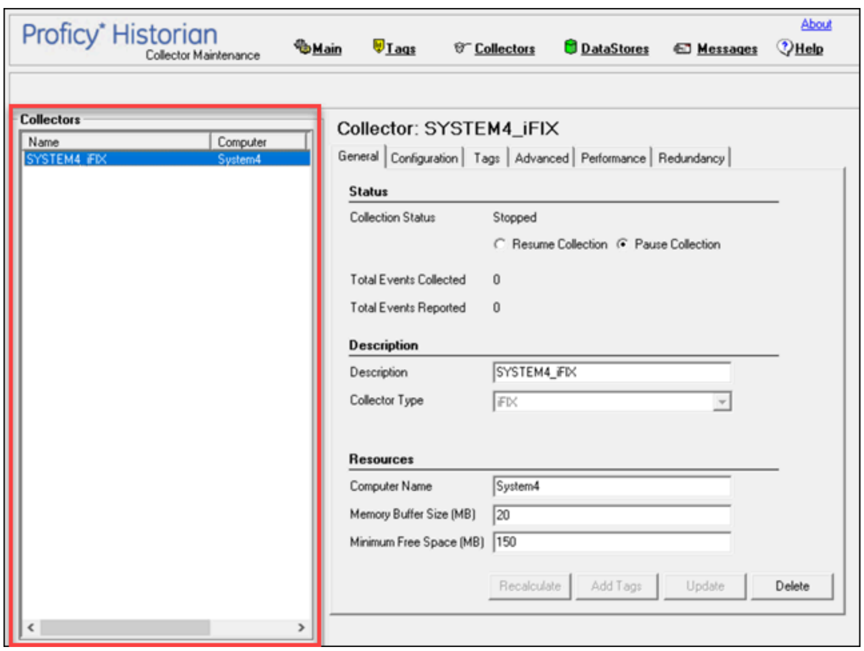
When you delete a collector, all of its tags are deleted from the Historian database.

1. Access Historian Administrator *(on page 384)*.
2. Select **Collectors**.



A list of collectors appears.

3. Select the collector whose details you want to delete.



4. Select **Delete**.

   A message appears, asking you to confirm that you want to delete the collector.

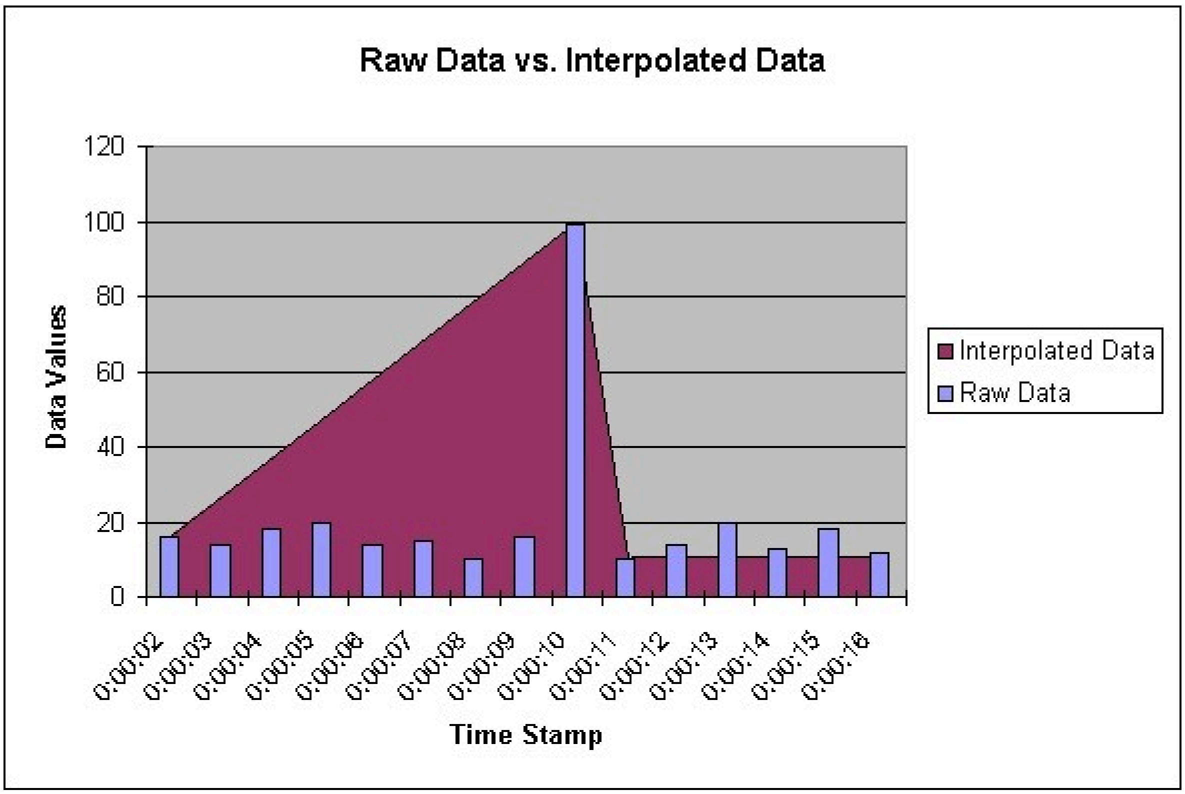
5. Select **Yes**.

   The collector is deleted.

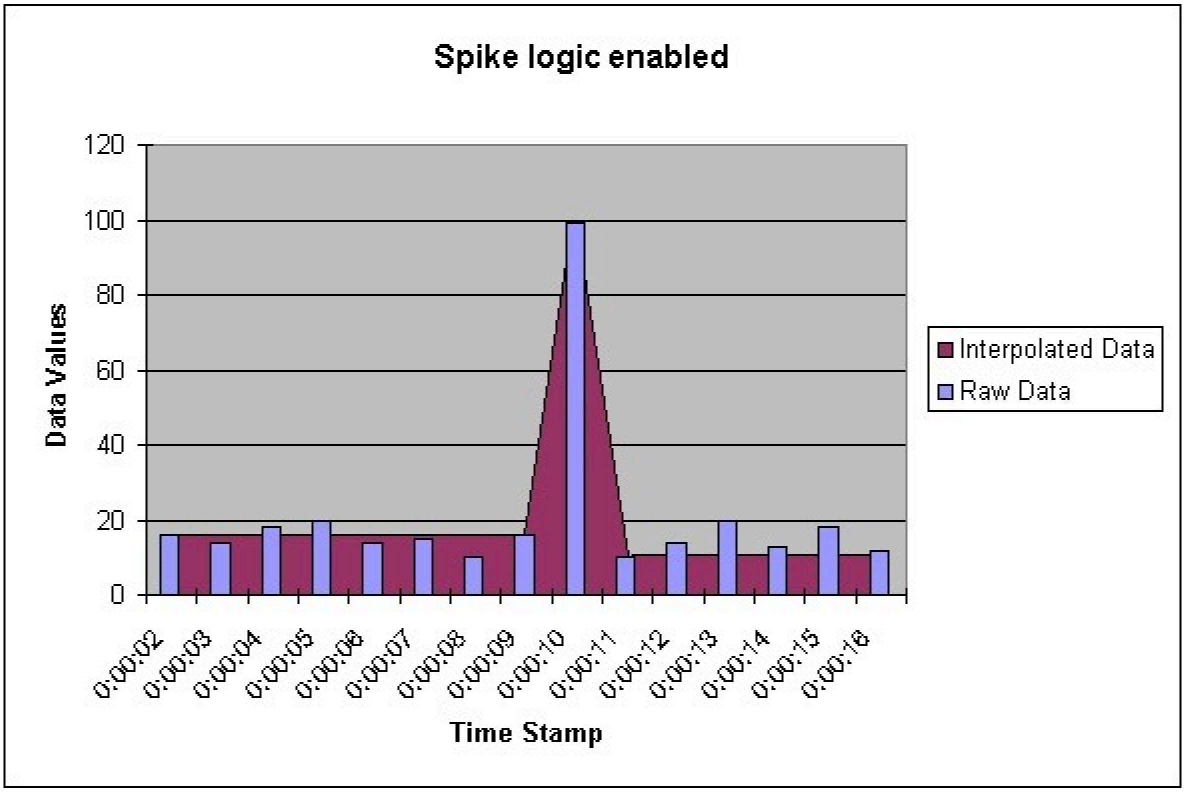
Enable Spike Logic

When compression is enabled in the Historian archive, only the first instance in a series of data falling within a deadband range will be collected to the Historian archive. When that data is plotted using interpolation, false values are inserted into the chart to create a smooth trend between intervals in a given time period. In most cases, interpolation gives a reasonable portrayal of the actual data for a given time period.

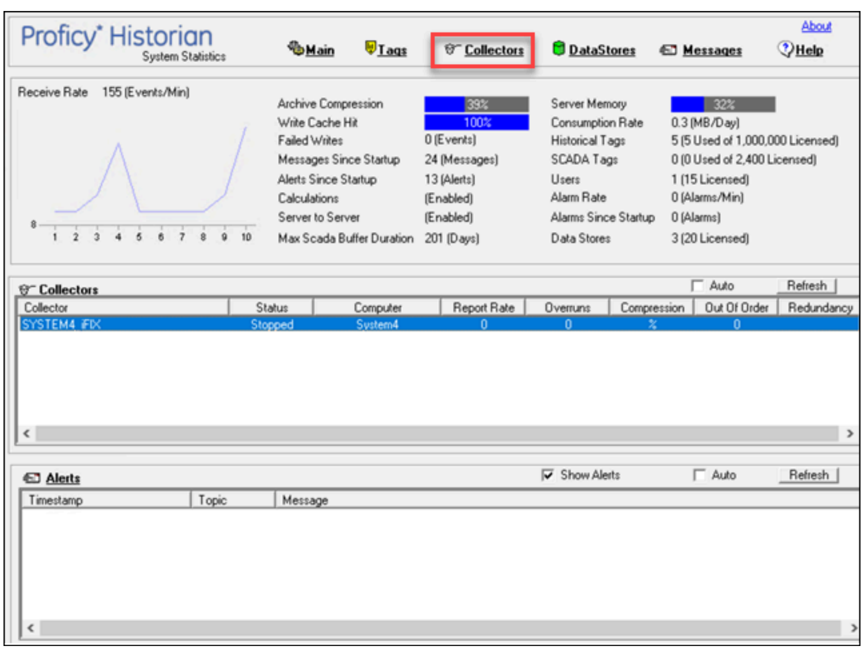
Unfortunately, in the event of a spike in data values, an unrealistic set of samples is created when the data is plotted. Instead of showing the results of compression (the same values over a series of intervals), a rising or falling slope is created in the chart. This gives the impression that values for a given time stamp are higher or lower than they actually were. The figure below shows the difference between the raw data for a series of samples, and how the samples would be plotted if data compression were enabled, assuming all values between 10 and 20 are in the deadband range.



Spike logic monitors incoming data samples for spikes in a tag's values. If spike logic is enabled, a sample of equal value to the previously archived sample is inserted into the archive in front of the spike value. The time stamp of the inserted value is determined by your polling interval. If samples are collected at one-second intervals, the inserted sample's time stamp will be one second before the spike. This helps identify the spike, and retains a more accurate picture of the data leading up to it, as shown in the following image.

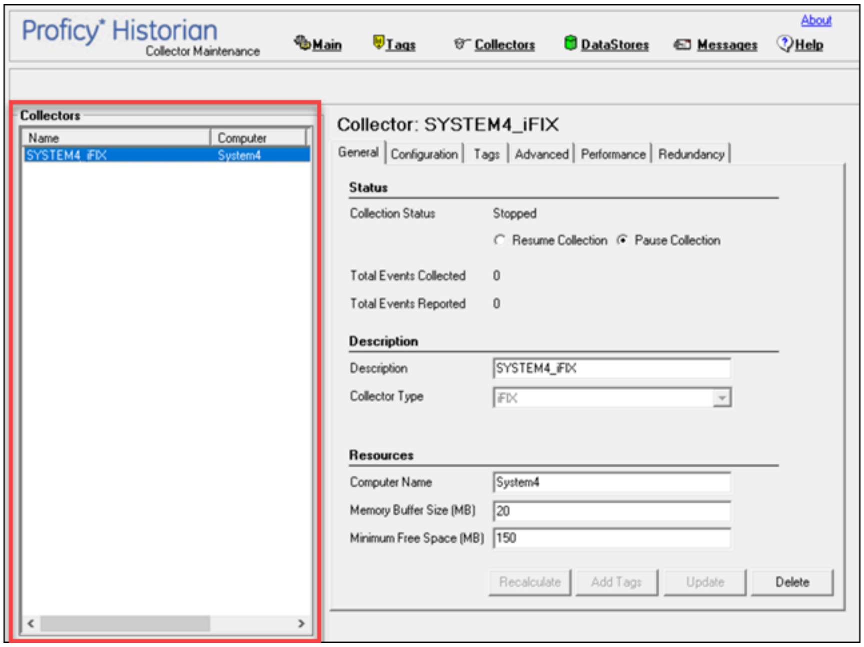


1. Access Historian Administrator *(on page 384)*.
2. Select **Collectors**.

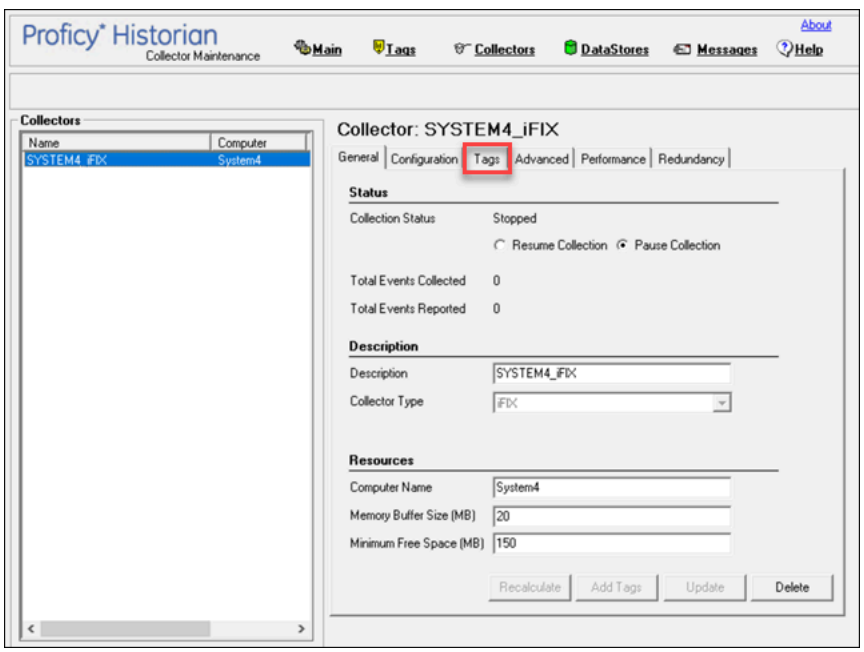


A list of collectors appears.

3. Select the collector to which you want to apply spike logic.



4. Select **Tags**.



5. Under **Default Compression**, in the **Spike Logic Control** field, select **Enabled**, and then select one of the following options:

  - **Multiplier**: Specifies how much larger a spike value must be than the deadband range before spike logic will be invoked. For example, if you enter 3, and the deadband percentage was set to 5%, spike logic will not be invoked until the difference between the spike value and the previously archived data point is 15% of the EGU range.
  - **Interval**: Specifies how many samples must have been compressed before the spike logic will be invoked. For example, if you enter 4, and 6 values have been compressed since the last archived data sample, spike logic will be invoked.
6. Select **Update**.
   The spike logic is enabled.

Maintaining, Operating, and Monitoring Historian

Maintain, Operate, and Monitor Historian

To ensure reliable, error-free operation over a long period of time, develop and execute a consistent maintenance program for the Historian system and the data it collects. The subsequent topics provide guidelines for setting up such a plan and for monitoring and interpreting system performance indicators.

Data Types

Historian uses the following data types.

Data Type	Size	Description	Valid Values
Single Float	4 bytes	Stores decimal values up to 6 places.	1.175494351e-38F to 3.402823466e+38F
Double Float	8 bytes	Stores decimal values up to 15 places.	2.2250738585072014e-308 to 1.7976931348623158e +308
Single Integer	2 bytes	Stores whole numbers.	-32767 to +32767
Double Integer	4 bytes	Stores whole numbers.	- 2147483648 to +2147483648
Quad Integer	8 bytes	Stores whole numbers.	-9,223,372,036,854,775,808 (negative 9 quintillion) to +9,223,372,036,854,775,807 (positive 9 quintillion)
Unsigned Single Integer	2 bytes	Stores whole numbers.	0 to 65535

Data Type	Size	Description	Valid Values
Unsigned Double Integer	4 bytes	Stores whole numbers.	0 to 4,294,967, 295 (4.2 billion)
Unsigned Quad Integer	8 bytes	Stores whole numbers.	0 to 18,446,744,073,709,551,615 (19 quintillion)
Byte	1 byte	Stores integer values.	-128 to +127
Boolean	1 byte	Stores boolean values.	0=FALSE and 1=TRUE (any value other than zero is treated as one)
Fixed String	Configured by user	Stores string data of a fixed size.	0 and 255 bytes
Variable String	No fixed size	Stores string values of undetermined size. This data type is useful if you cannot rely on a constant string length from your data source.	
Binary Object	No fixed size	Stores binary data. This is useful for capturing data that can not be classified by any other data type.	
Scaled	2 bytes	Stores a 4 byte float as a 2 byte integer. The scaled data type saves disk space but sacrifices data precision as a result.	

Note:
Tags associated with Quad Integer, Unsigned Double Integer, or Unsigned Quad Integer data types may suffer a loss of precision value due to a Visual Basic limitation.

Scaled Data Types

Historian uses the high and low EGU values to store and retrieve archived values for the scaled data type. This allows you to store 4 byte floats as 2 byte integers in the archive. Though this saves disk space, it sacrifices data precision. The smaller the span is between the high and low EGU limits, the more precise the retrieved value will be. When calculating the value of a scaled data type, you can use this formula:

```
ArchivedValue = (((RealWorldValue - EngUnits->Low) /

 (EngUnits->High - EngUnits->Low) * (float) HR_SCALED_MAX_VALUE) + .5);
```

For example: A value of 12.345 was stored in a scaled tag whose high EGU was 200 and low EGU was 0. When later retrieved from the Historian archive, a value of 12.34473 will be returned.

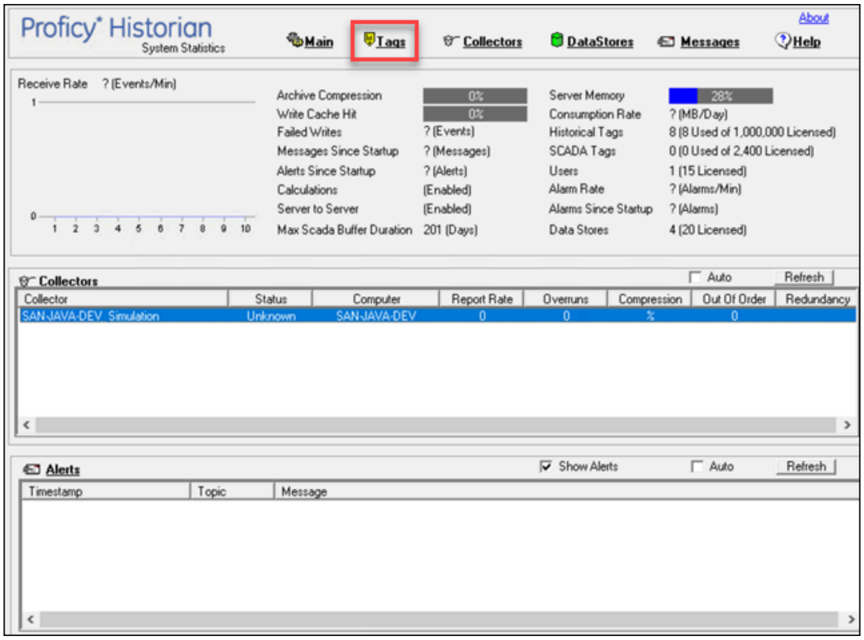
> **Important:**
> Values that are outside of the EGU range of a scaled data type tag are stored as bad, scaledoutofrange in Historian. Changing either the High or Low EGU tags does not affect existing data, but only affects the new data with new timestamps. You cannot correct values for scaled data types that were inserted while EGUs were incorrect. If necessary, contact technical support for additional information.

Quad Integer Data Types: The high and low EGU limits for Quad Integer, Unsigned Single Integer, Unsigned Double Integer, Unsigned Quad Integer are between 2.2250738585072014e-308 to 1.7976931348623158e+308.

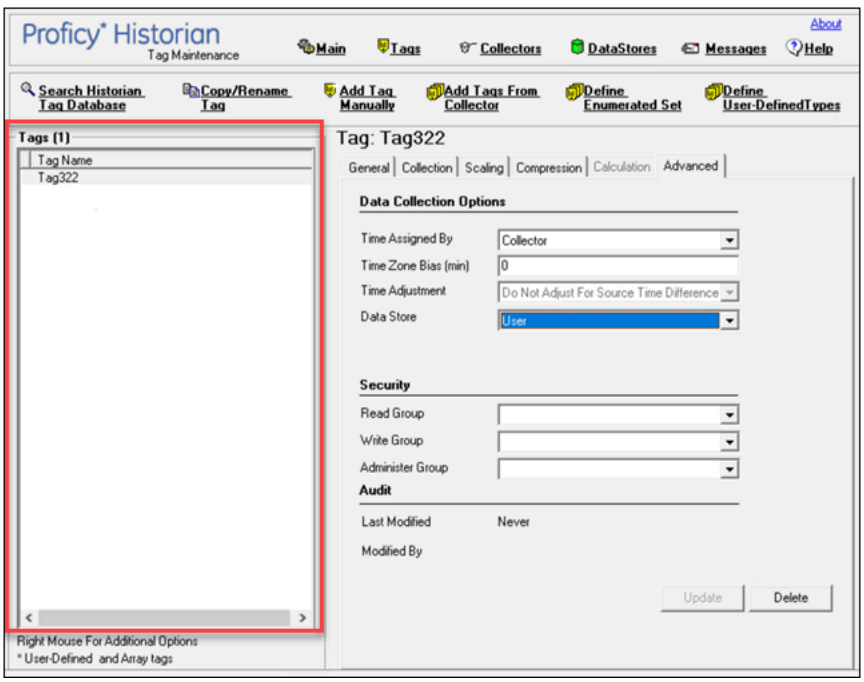
Set the Size of a Fixed String Data Type

Using the fixed string data type, you can store string data of a fixed size. This is useful when you know exactly what data will be received by Historian. If a value is larger than the size specified in the **Data Length** field, it will be truncated.

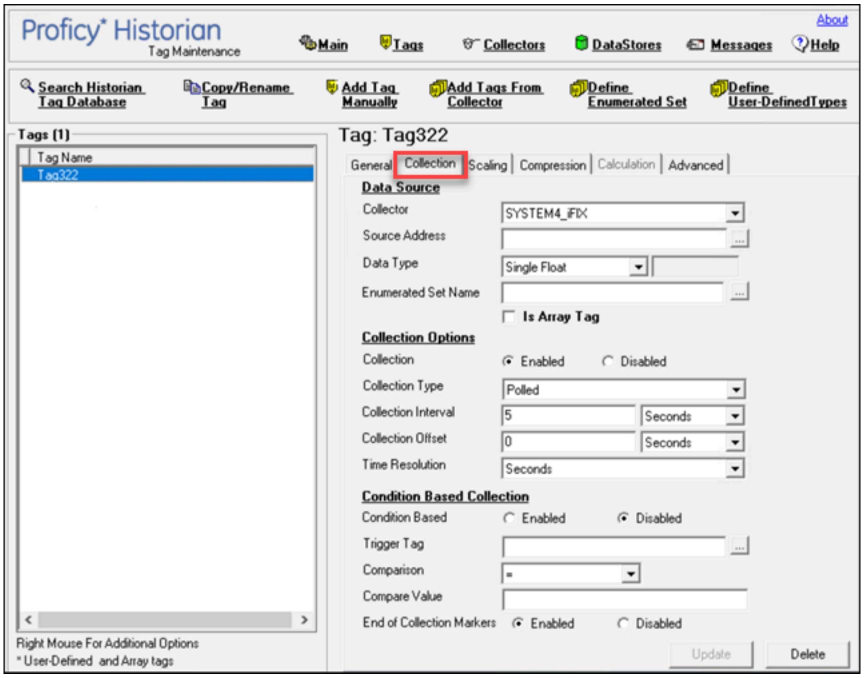
1. Access Historian Administrator *(on page 384)*.
2. Select **Tags**.



3. Select the tag for which you want to set a fixed string data type.



4. Select **Collection**.



5. In the **Data Type** box, select **Fixed String**.
6. Enter a value in bytes in the adjacent field.

   The fixed string data type is set for the tag.

Develop a Maintenance Plan

The primary goal of a maintenance plan is to maintain integrity of the data collected. If you are successful in this regard, you will always be able to recover from a service interruption and continue operation with minimal or no loss of data. Since you can never ensure 100% system uptime, you must frequently and regularly back up current data and configuration files, and maintain non-current archive files in a read-only state, following the guidelines for backup and routine maintenance.

Routine Maintenance: On a regular schedule, examine and analyze the system performance indicators displayed on the System Statistics page of Historian Administrator as follows.

Table 74. System Statistics Performance Indicators

Field	Recommended Action
Consumption Rate of Archive Storage	If the rate is excessively high, reduce the rate at which data flows into the system or increase the filtering applied to the data to lower the rate of archiving. To reduce the collection rate, slow the polling rate on some or all tags. To increase filtering, enable compression at the collector and/or archiver and widen the compression deadbands.

Table 74. System Statistics Performance Indicators (continued)

Field	Recommended Action
Failed Writes	If the display shows a significant number of failed writes, investigate the cause and take corrective action to eliminate the malfunctions. Refer to the `DataArchiver-XX.log` file or query the message database to determine the tags for which failed writes occurred. For example, trying to write values to a deleted archive causes failed writes. Trying to archive data with a timestamp that precedes the start time of the first archive, trying to write to a read-only archive, or trying to write a value with a timestamp more than 15 minutes ahead of the current time on an archiver will produce a failed write.
System Alerts	Not applicable

On a regular schedule, examine and analyze the performance indicators displayed in the **Performance** section.

Table 75. Collector Performance Indicators

Field	Recommended Actions
Avg. Event Rate Chart	Not applicable
Compression Chart	Is compression effectiveness acceptable? If not, verify that compression is enabled and then widen the deadbands to increase the effect of compression.
Overruns Chart	If the value is anything other than zero, determine the severity and cause of the problem and take corrective action.

Troubleshooting

Solve Minor Operating Problems

The following is a table of troubleshooting tips for solving minor operating problems with Historian.

Issue	Suggested Action
After setting the system clock back, browsing the collector from	Delete temporary Internet files and restart Internet Explorer.

Issue	Suggested Action
Historian Administrator produces a Visual Basic script error.	
With the Historian Administrator, switching usernames causes the system to reject the login if the User must change password at next login option is selected at time of user creation.	New users with this setting must log in to the appropriate Windows operating system at least once. If the login attempt fails, run Historian Administrator as an administrator, and log in with a new username and password.
Excel Sample Reports do not display data.	When opening a Sample Excel report, you may receive a message prompting you to update all linked information in the workbook (Yes) or keep the existing information (No). It is recommended that you select No and keep the existing information. The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated.
Need to connect an Historian Server to an Historian Client through a firewall.	Open port 14000 to enable client to server connection through a firewall.
Receiving archive offline failed writes messages.	These occur when the timestamps of data being sent to the archiver are not in the valid time range of any online archives. For example, failed writes occur when the timestamps appear *before* the oldest archive, the archive is offline, or timestamps are more than 15 minutes *past* the current time on another archiver.

FAQ: Run a Collector as a Service

The following list is frequently asked questions about running a collector as a service.

Can all collectors be run as a Windows service? If not, which ones cannot?

The OPC Collector, Simulation collector, and Server-to- Server collector can be run as services. The iFIX collector run as a background task and cannot be run as a service.

Can all collectors be run as an application? If not, which ones cannot?

All collectors can run as applications (console programs). This includes the Simulation Collector. To make a collector run as a console program, pass a RUNASDOS command line parameter.

What does "running as a service" mean?

It means that the collector appears in the Control Panel list of services. It can run at system boot or be run with a different username and password from the currently logged-in user.

How can the iFIX collector be set up to run when no one is logged in?

It can be set up to run without a user login by adding it to the iFIX SCU task list as a background task and by configuring iFIX to continue running after logging off in local startup.

How do you shut down a collector running as console application?

Collectors started as console applications should be shut down by typing S at the command prompt in the DOS window and pressing Enter.

Can a collector be run as a Windows service and then stopped and restarted?

Yes. Collectors that can run as a service can be stopped and started in Control Panel Services. They can be paused/resumed through Historian Administrator.

What is the difference between running a collector to start as a service on boot up using the Services applet in Control Panel versus running iFIX as a service, which starts the collector through the startup task configuration in the SCU?

The collectors that can run as a service would not be started from iFIX. They can be started from the Control Panel start at system boot. Although you cannot run an iFIX collector as a service, you can log off and on while it is running.

Changing the Base Name of Automatically Created Archives

When the IHC file is created, it stores the name of the server inside the IHC file. Automatically created archives use that server name from the IHC file as a base name, not the Base Archive Name configured in Historian Administrator.

When you *manually* create an archive, however, the archive uses the Base Archive Name from Historian Administrator.

If you move an IHC file from one machine to another, you may want to change the default base archive name to match the new server. To change the default Base Archive Name, create a new .IHC file.

1. Export your tags using the Excel Add-in.
   The Fields to Export window appears.
2. Select all tag attributes and select **OK**.

The data is exported to a new Excel worksheet.

3. Examine the **Comments** column in the new worksheet. To ensure a clean import, the Comments column must be completely full or completely empty. If no comments are found, this column must be deleted to ensure a clean import. If only some comments are missing, the missing fields must be filled out with comments.
4. Save the Excel spreadsheet.
5. Stop the Data Archiver service.
6. Open the default archive path in Windows Explorer and rename the .IHC file.
   Rename `MyMachine.IHC` to `MyMachine.OLD`.
7. Restart the Data Archiver service.
   A new, blank IHC file is created for the machine.
8. Import your tags to this new configuration using the Excel Add-In.

Configuring the Inactive Timeout Value

The Non-Web Administrator offers a configurable timeout. This configurable timeout determines how long the Non-Web Administrator will wait before severing its connection to an inactive Historian archive. The default timeout value is 90 seconds.

1. Assuming Historian is already open, double-click on the **Main** button to open Historian Administrator Login window.
2. Select the **Browse for Server** button.
3. Select the Historian server you wish to configure from the **Servers** list.
4. In the **Connection Timeout** field, select the **Use Value** option and enter a timeout value in seconds.

Configuring Deep Data Tree Warnings

Reading and writing to deep trees with large time ranges can be very inefficient. Create a **MaxIndexRecursionDepth** registry key to configure the depth at which the archiver will warn about deep data trees.

1. From the **Start** menu, select **Run**, and then enter `Regedit`.
2. Open the following key folder: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\DataArchiver`
3. Create a value as DWORD called **MaxIndexRecursionDepth**.
4. Set **MaxIndexRecursionDepth** to a number higher or lower than the default value of 900.
   To get more warning messages, set a smaller number if you have an archive which is 10 to 100 deep.
5. Select **OK**.

6. Close the Registry Editor.
7. Restart the archiver for the changes to take effect.

Control Data Flow Speeds with Registry Keys

Configure buffer flush speed with the BufferFlushMultiplier key

Store-and-forward buffering is a key feature of Historian collectors. It prevents data loss during planned or unplanned network outages between a collector and Historian server.

If the collector is disconnected from the archiver for several hours or days, many megabytes of data can be buffered and must be delivered by the collector to the Data Archiver upon reconnect. Since all data is sent from the collector to the archiver in time order, the design goal has been to catch up to real time as quickly as possible by sending data as fast as possible.

If this is not the desired behavior because you want to limit the network load on a slow, shared Wide Area Network (WAN) or you want to limit the CPU load on the Data Archiver caused by the incoming data, you can configure the collector to throttle the data it is sending.

> **Important:**
> Because data is sent in time order (oldest first), you will not be able to retrieve current historical data until the throttled flush is complete.

Configuring the throttle is easy, but it requires modifying a registry key, so it should be done with caution.

A DWORD registry key called **BufferFlushMultiplier** is present under each collector. For Windows 32-bit, it is located under `HKey_Local_Machine\Software\Intellution, Inc.\iHistorian\Services\YOUR_COLLECTOR_TYPE`. For Windows 64-bit, it is located under `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\Services\YOUR_COLLECTOR_TYPE`.

- To **slow** the store and forward throttling, set the value of **BufferFlushMultiplier** to 2. The 2 means that the collector should never send data at more than 2 times its normal rate to limit network and CPU load.
- To **disable** throttling, set the value of **BufferFlushMultiplier** to 0 or delete the registry key.

Control archiver speed with the NumIntervalsFlush registry key

The NumIntervalsFlush registry key controls how quickly the collector sends data to the archiver. The collector collects from the data source at the user configured rate, but for efficiency it bundles data samples in a single write to archive. By default, the collector will send data to archiver every 2 seconds or

10,000 samples, whichever happens first. Most often, it sends every 2 seconds because the collector is not collecting that many samples that fast.

If you need collected data sent to archiver right away, so that it is available for retrieval or for calculations, use the NumIntervalsFlush registry key.

You will have to create the registry key, as it does not exist by default. Create a DWORD value called **NumIntervalsFlush** under the collector, in the same place as HISTORIANNODENAME and INTERFACENAME. On a 64-bit Windows Operating System, all 32-bit component-related registry keys (such as collectors, Client Tools, and APIs) will be located under `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\`.

The preferred setting for **Num Intervals Before Flush** is 5. The intervals are 100 millisecond increments. The default of 20 means `(20 * 100msec) = 2000 msec = 2 seconds`. Set the value to 5 and the collector will send every 500msec.

Note:
Changes to the registry key do not take effect until the collector is restarted. This setting affects the sending of data whether it was collected polled or unsolicited.

Configure Inactive Server Reset Timeout

You can configure inactive server connections to reset automatically with the SocketRecvTimeOut registry key. SocketRecvTimeOut configures a timeout that forces the connection to drop and re-establish if no data is received during the specified time. Consider this configuration when your collector goes to status Unknown for long periods of time even when the connection between collector and archiver is good.

Create a DWORD registry key **SocketRecvTimeOut** under the collector where the problem is occurring and set to a value greater than 90 seconds. A typical value would be 300 seconds. If no bytes are received by the collector for 300 seconds, then the network connection will be closed and re-established.

Historian Errors and Message Codes

When you review errors and messages, for example, in an Historian archiver log file, full descriptions are usually included. If a number appears instead of a description, use the following table to determine the meaning of the error or message.

Table 76. Historian Error Codes and Messages

Number	Description
-32	Operation not permitted
-31	The requested data store was not found
-29	A supplied argument is outside the valid range
-28	A supplied argument is NULL
-27	A supplied argument is invalid
-25	Attempted data delete outside allowed modification interval
-24	Data Retrieval Count Exceeded
-23	Invalid Server Version
-21	Calculation Circular Reference
-20	Not Licensed
-19	Duplicate Interface
-18	No Value
-17	License: Invalid License DLL
-16	License: Too Many Users
-15	License: Too Many Tags
-14	Invalid Tagname
-13	Write No Archive Available
-12	Write Outside Active
-11	Archive Read Only
-10	Write Archive Offline
-9	Write in Future
-8	Access Denied
-7	Not Valid User
-6	Duplicate Data
-5	Not Supported

Table 76. Historian Error Codes and Messages (continued)

Number	Description
-4	Interface Not Found
-3	Not Connected
-2	API Timeout
-1	FAILED
0	OK
0	Undefined
1	Connection Successful
2	Connection Unsuccessful
3	Audited Write
4	Audited Write Update
5	Audited Write Out Of Order
6	Audited Write Update Out Of Order
7	Message On Update
8	Message On Update Out Of Order
9	License Library Function Missing
10	License Library Missing
11	Failed Write
12	Tag Added
13	Tag Modified
14	Tag Deleted
15	Interface Added
16	Interface Modified
17	Interface Deleted
18	Archive Added
19	Archive Add Failure Time Overlap

Table 76. Historian Error Codes and Messages (continued)

Number	Description
20	Archive Deleted
21	Archive Overwritten
25	Archive Five Days Till Closing
26	Archive Three Days Till Closing
27	Archive One Day Till Closing
28	License Key Removed
29	License Max Tags Exceeded
30	License Max Users Exceeded
31	License Max Tags Exceeded Shutdown
32	License Max Users Exceeded Shutdown
33	License Library Invalid
34	Buffer Normal
35	Buffer On Disk
36	Buffer Out Of Space
37	Incomplete Shutdown
38	Archive Modified
39	License Expired
40	Buffer Could Not Create
41	Archiver Startup
42	Archiver Shutdown
43	Audit Status Changed
44	Option Modified
45	Write Processing Stopped
46	Write Processing Resumed
47	Interface Status Unknown

Table 76. Historian Error Codes and Messages (continued)

Number	Description
48	Archive Closed
49	Interface Stopped
50	Interface Started

Scheduled Software Performance Impact

Running continuous disk scan software applications such as anti-virus scans, or any other software that accesses disk drives to a high degree may affect the overall performance of your Historian System by competing with Historian for disk resources.

If your Historian System requires that you need an extremely high throughput (20k/sec or greater), consider **disabling** the scheduled software execution.

Intellution 7.x Drivers as OPC Servers

The ABR and the ABC drivers are OPC v2.0 compliant. All other Intellution 7.x drivers, including the MB1, support OPC v1.0 compliance.

Version 7.x drivers also comply with the OLE for Process Control (OPC) v1.0a standard. Any 1.0-compliant OPC client application can access process hardware data through the I/O Server.

Troubleshooting Failed Logins

The following is a table of error messages, possible causes, and recommended corrective actions for failed logins sometimes experienced with Historian Administrator.

Error Message	Suggested Action
User does not have authority to read messages.	The user is NOT a member of iH Readers nor a member of the iH Security Admins security groups. To access the Main Page, the user must have read access.
[07/18/2001 03:00:46.071 PM] USER: DOMAIN1\administrator TOPIC: Security MSG: DOMAIN1\administrator(administrator) unsuccessfully connected at 07/18/2001 03:00:46.071 PM.	Error in Internet Explorer. Check network connection or IIS on the server.

Error Message	Suggested Action
Not able to establish session to the server from a remote Web-based Clients. Page cannot be displayed.	
[07/17/2001 07:56:06.950 PM] USER: DataArchiver TOPIC: Security MSG: DataArchiver(DataArchiver) Exceeded number of licensed users at 07/17/2001 07:56:06.950 PM (NumUsers=0 MaxUsers=0)	Outdated or failed HASP key is attached. Obtain new key from technical support.
[07/17/2001 07:58:18.980 PM] USER: \baduser TOPIC: Security MSG: \baduser(baduser) unsuccessfully connected at 07/17/2001 07:58:18.980 PM.	Bad password for user account. Enter correct password.
[07/17/2001 07:58:48.712 PM] USER: \administrator TOPIC: Security MSG: \administrator(administrator) unsuccessfully connected at 07/17/2001 07:58:48.712 PM.	DataArchiver service is not running. Results in a Failed to connect to server error. Make sure data archiver service is running and that the user did not enter a bad password.
[07/17/2001 07:23:44.397 PM] USER: DataArchiver TOPIC: Security MSG: DataArchiver(DataArchiver) Exceeded number of licensed tags at 07/17/2001 07:23:44.397 PM (NumTags=1021 MaxTags=0) Must Shutdown	Number of configured tags exceeds number of tags allowed by the key. Delete enough tags to meet the license limit or obtain a license for more tags from technical support.
[07/17/2001 07:35:32.134 PM] USER: DataArchiver TOPIC: Security MSG: DataArchiver(DataArchiver) Licensed expired. Must shutdown 07/17/2001 07:35:32.134 PM. To troubleshoot, refer to the DataArchiver.log file.	License on key has expired (archiver will not start). Obtain a new license from technical support.

Troubleshoot Data Collector Configuration

Troubleshooting Data Collector configuration and/or performance requires a thorough understanding of how Historian works and how the various parameters affect system operation. Armed with this knowledge, you can usually localize a problem to one or more functions or parameter settings and take effective corrective action.

Historian offers several tools to help find the cause of an operating problem.

LOG files

The system creates a new log file each time an archiver or collector is started. You can open these files in Notepad or another text editor. The `-nn` suffix in the file name indicates the place of each log file in the time sequence.

The data collector log files, located in the `LogFiles` folder within the Historian program folder, are a historical journal of every event affecting operation of the collector.
The `DataArchiver-nn.LOG` files are sequential files for the archiver only.
The `iFIX collector-nn.LOG` files contain performance information on iFIX collector functioning.

SHW file

The Data Collector .SHW file shows configuration data for collectors and is also located in the `LogFiles` directory under Historian. Verify that the parameter and configuration settings match what you configured. You can open this file in Notepad or another text editor.

Collector Maintenance Performance Indicators

These performance and status indicators can be a major aid in identifying, localizing, and diagnosing a problem with a collector.

Collector Maintenance pages

The Collector Maintenance pages can provide useful information about settings and selections of various options and parameter values. Examine each field and verify that it is appropriate for the current application.

Troubleshoot Tags

Tag Configuration

To diagnose a problem in tag configuration, examine the .LOG and .SHW files in the Historian/Logfiles directory. Since these files are a journal record of all system events and parameter modifications important to a system administrator, they can be helpful in identifying and localizing the source of a system malfunction or data error.

Stale Tags

If you are not seeing all stale tags in the Historian Web Admin, the ClientManager service may be down. If so, the thread that processes stale tags is suspended until connection is restored.

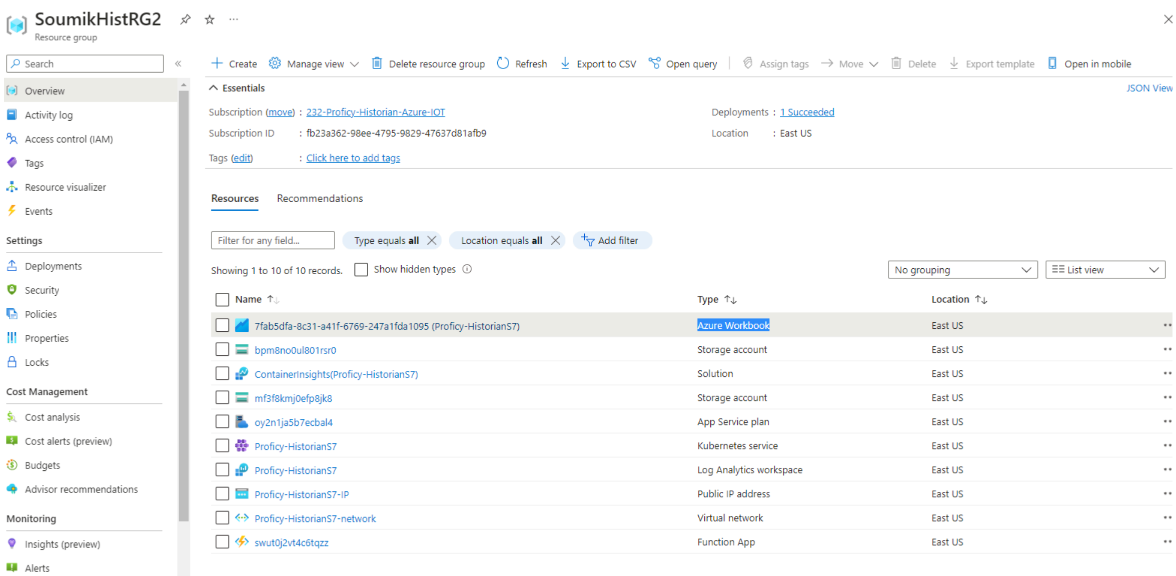
Chapter 10. Monitoring

Access Logs

Deploy Proficy Historian for Azure Cloud *(on page 17)*.

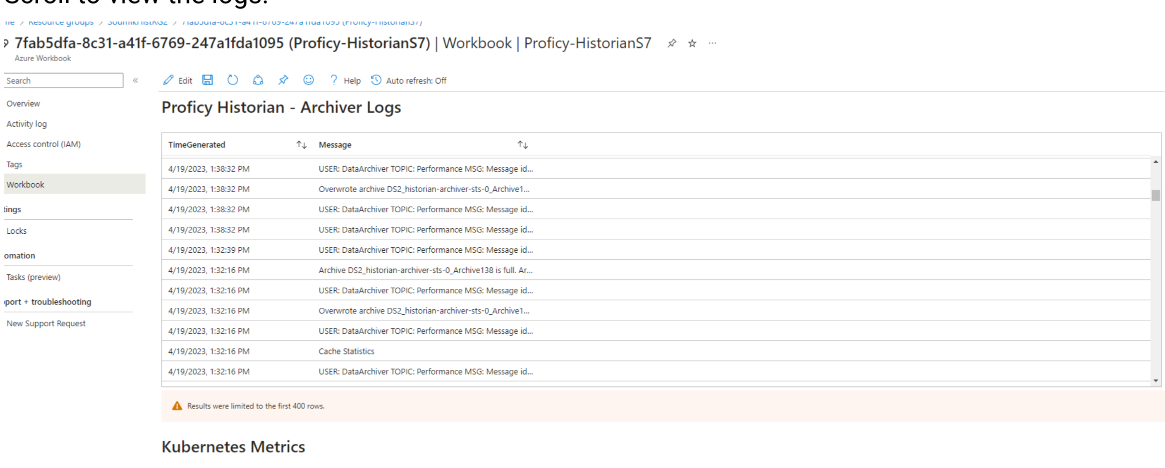
You can access and analyze logs using Azure Workbook.

1. Log in to Azure portal.
2. Under **Resource Group** , select the type Azure Workbook.



The **Workbook** page appears. After clicking on open workbook, Archiver Logs and other metrics will be visible for monitoring purposes.

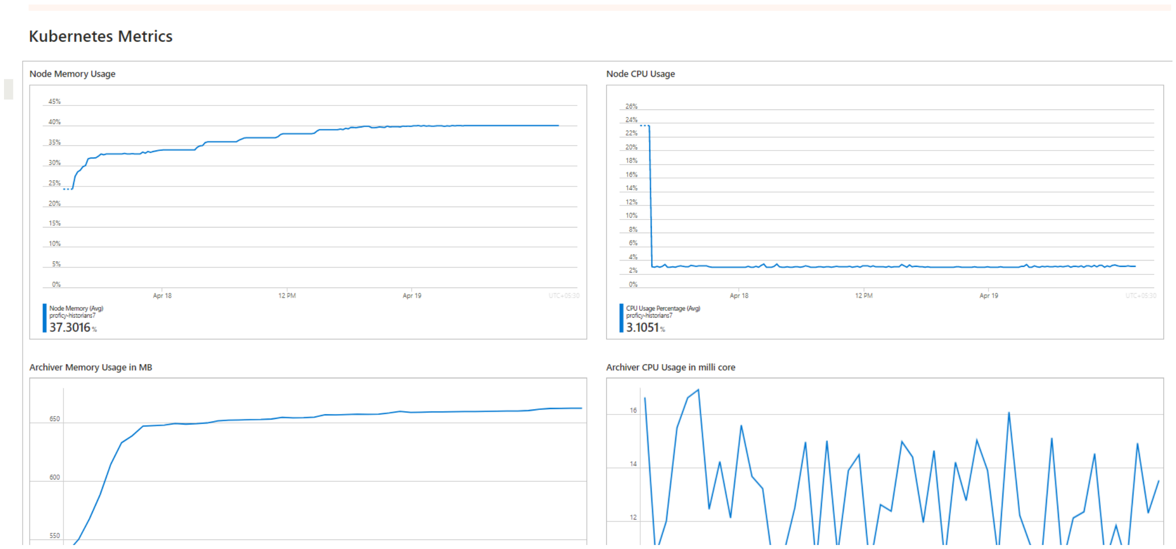
3. Scroll to view the logs.



4. From the list, select the one that contains the Kubernetes Metrics you want to review.

The following widgets appear.

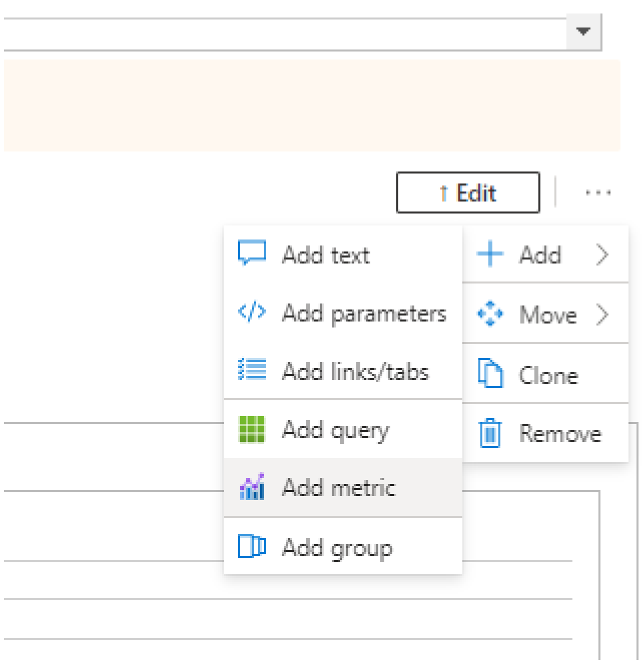
- Node Memory Usage
- Node CPU Usage
- Historian Archiver Memory Usage in MB
- Historian CPU Usage in milli core



You can add more widgets as needed.

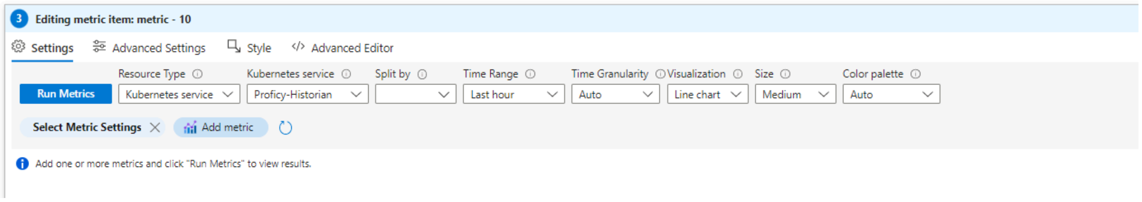
To add more metrics in a workbook:

Next to Edit click **... -> Add -> Add metric**.

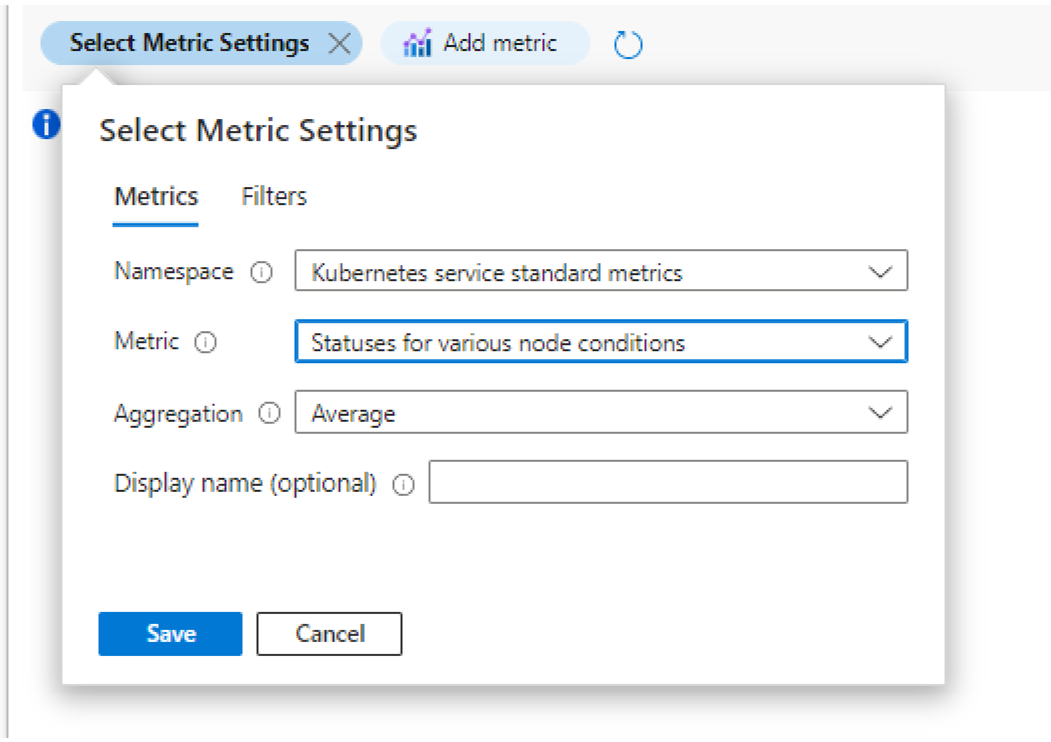


Under Settings, select the following and then **Add metric**.

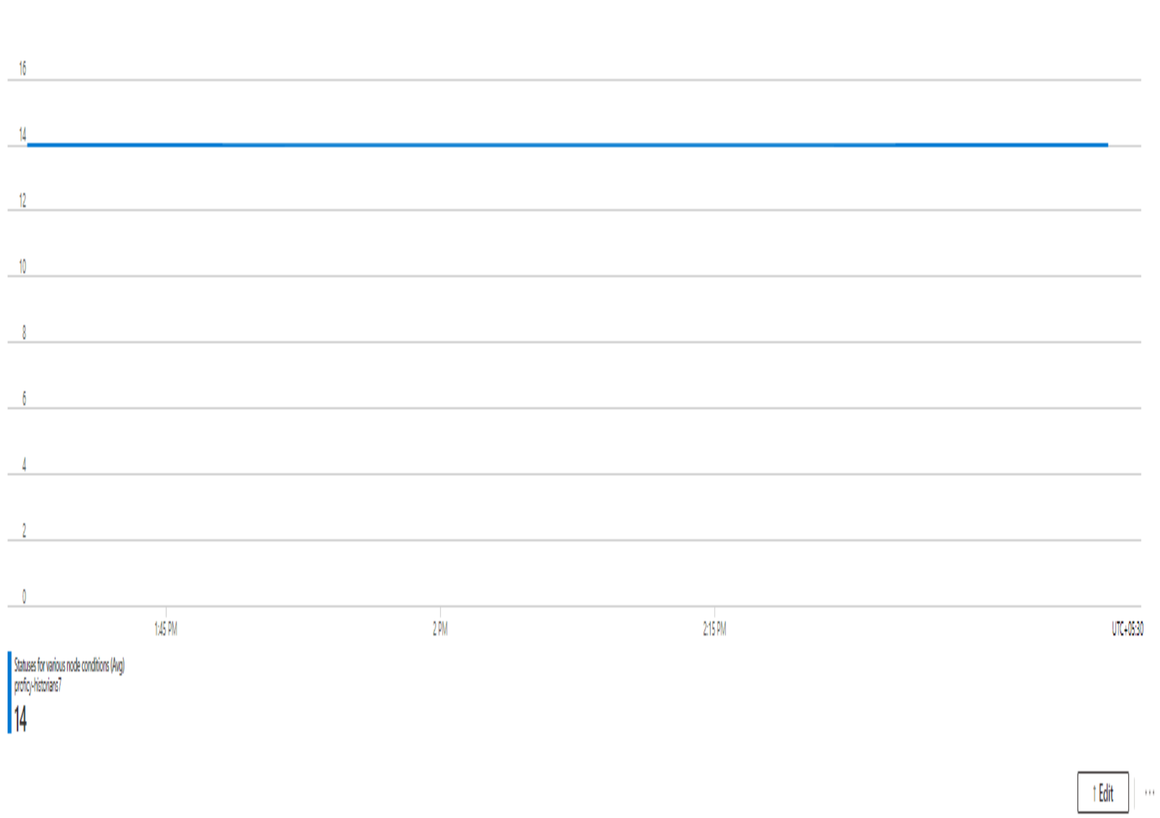
Resource Type: Kubernetes Service, Kubernetes Service: <cluster_name> as given while deploying. If the cluster_name not provided at time of deployment, the default name is: Proficy-Historian.



Under select metrics settings, select Namespace: As required, and Metric: As required.



Finally, click on **Save -> Run Metrics -> Done Editing**, after adding the new



metric.

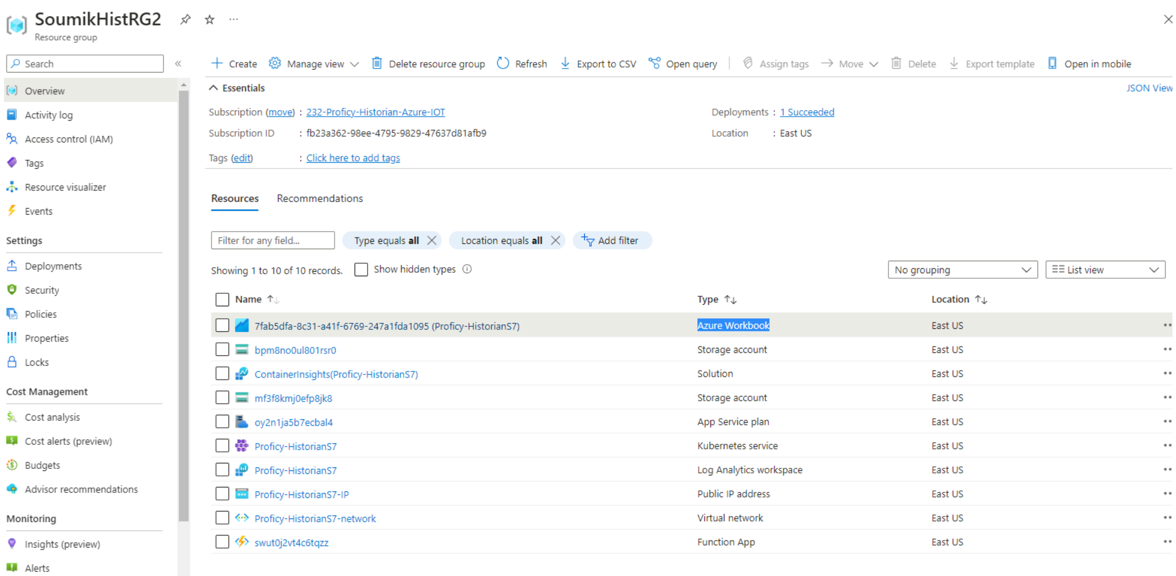
Chapter 11. Troubleshooting

Access Logs

Deploy Proficy Historian for Azure Cloud *(on page 17)*.

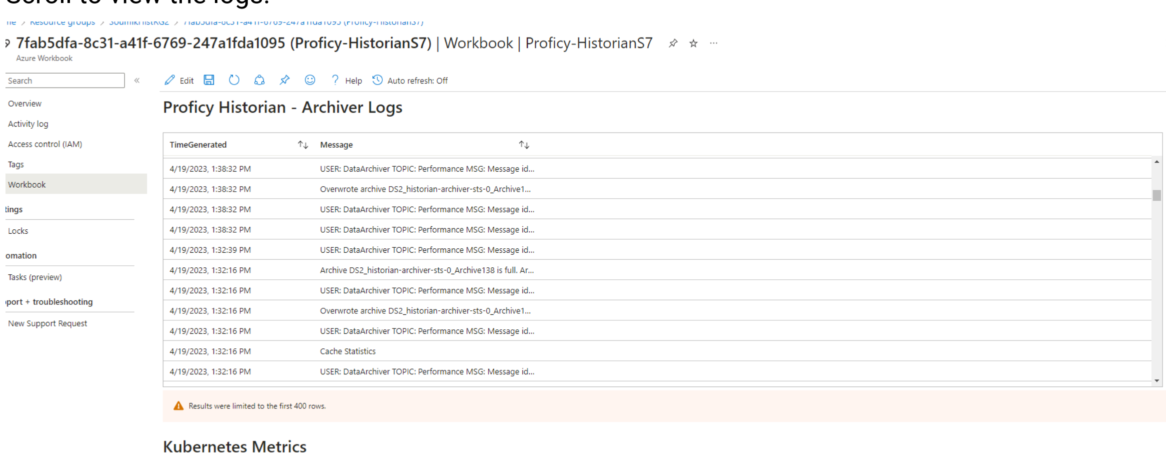
You can access and analyze logs using Azure Workbook.

1. Log in to Azure portal.
2. Under **Resource Group** , select the type Azure Workbook.



The **Workbook** page appears. After clicking on open workbook, Archiver Logs and other metrics will be visible for monitoring purposes.

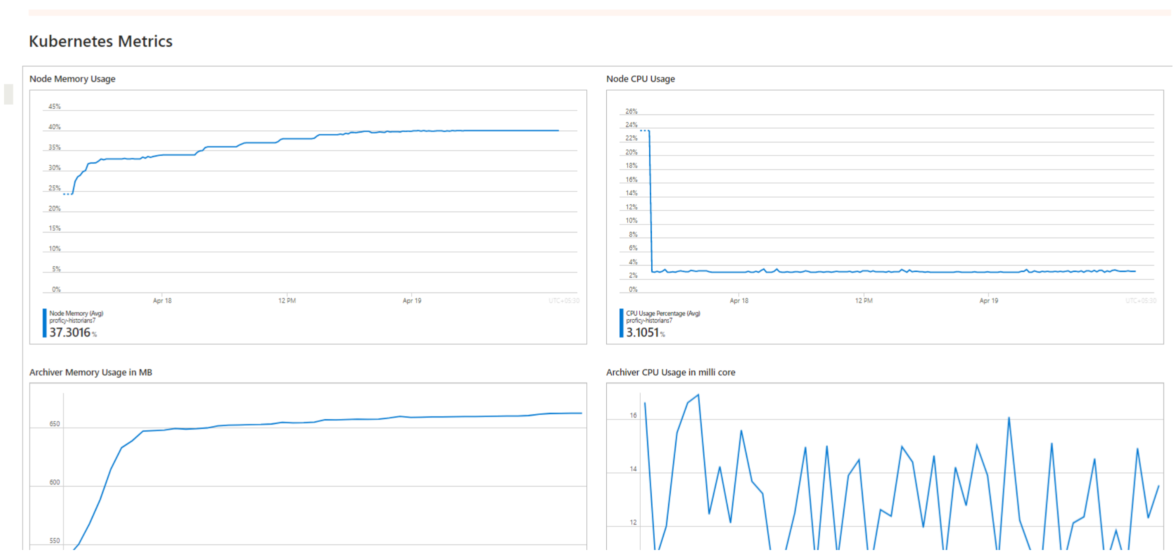
3. Scroll to view the logs.



4. From the list, select the one that contains the Kubernetes Metrics you want to review.

The following widgets appear.

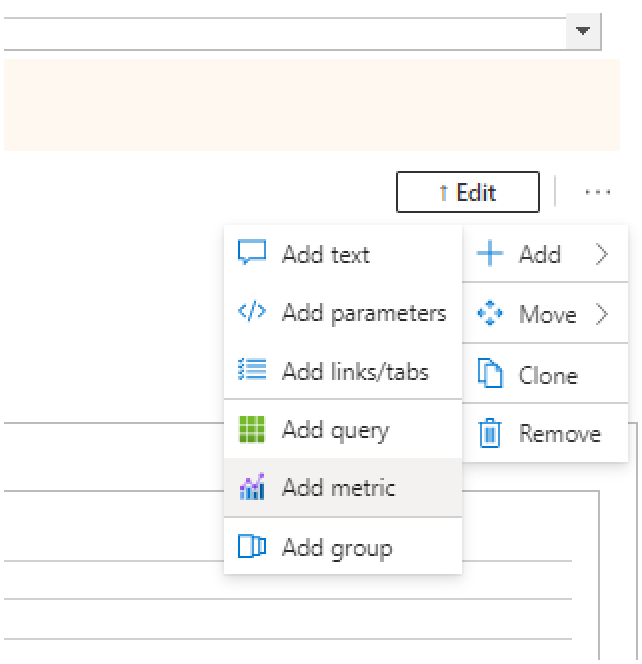
- Node Memory Usage
- Node CPU Usage
- Historian Archiver Memory Usage in MB
- Historian CPU Usage in milli core



You can add more widgets as needed.

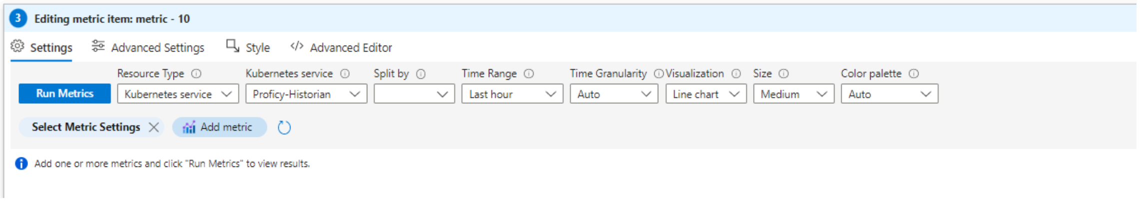
To add more metrics in a workbook:

Next to Edit click **... -> Add -> Add metric**.

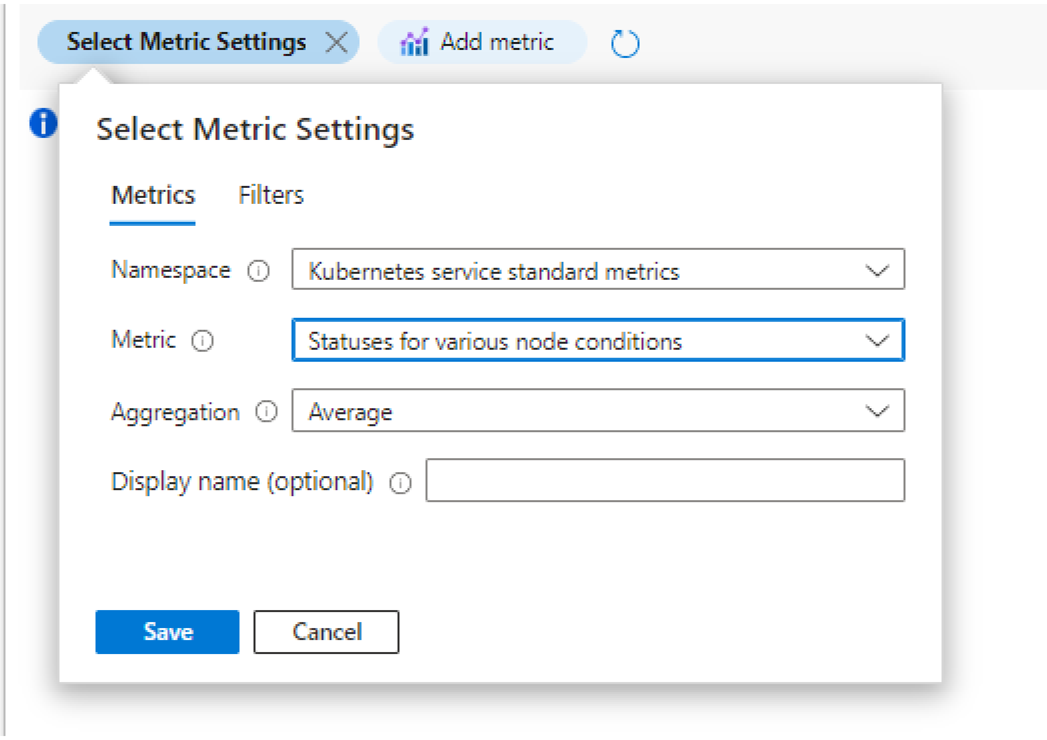


Under Settings, select the following and then **Add metric**.

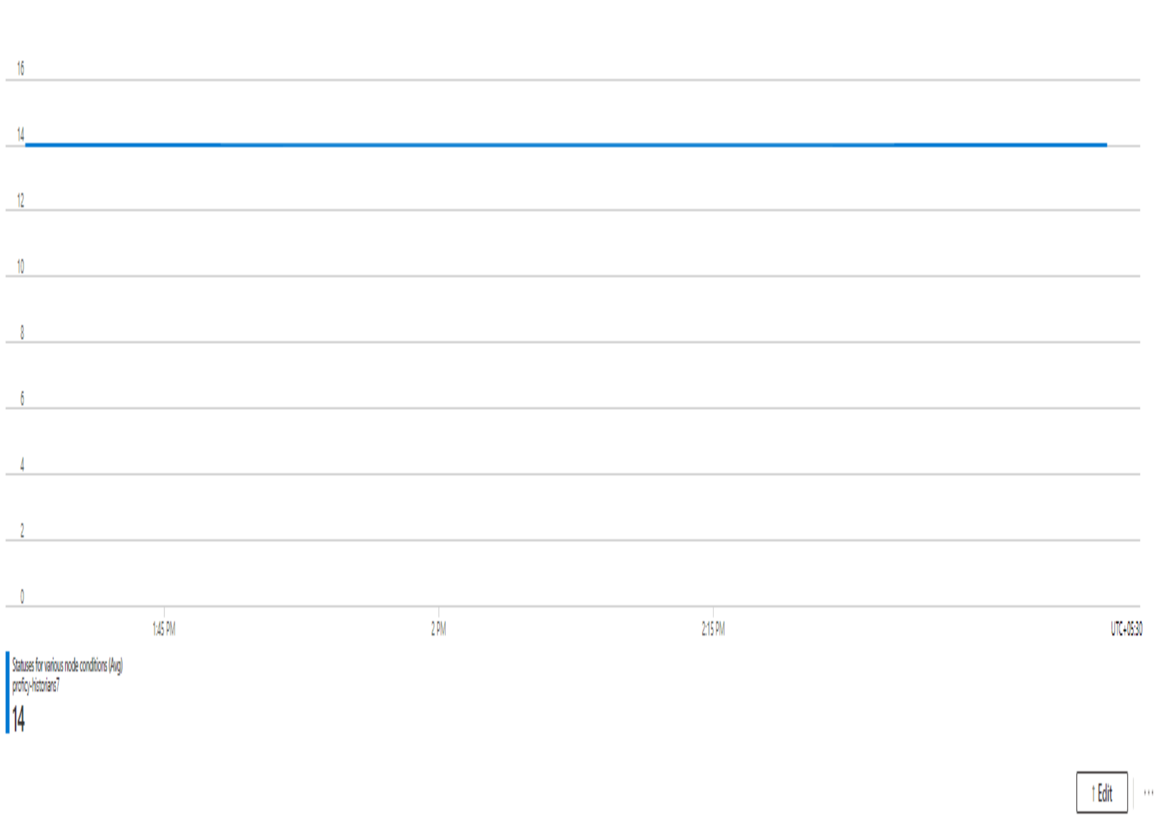
Resource Type: Kubernetes Service, Kubernetes Service: <cluster_name> as given while deploying. If the cluster_name not provided at time of deployment, the default name is: Proficy-Historian.



Under select metrics settings, select Namespace: As required, and Metric: As required.



Finally, click on **Save -> Run Metrics -> Done Editing**, after adding the new



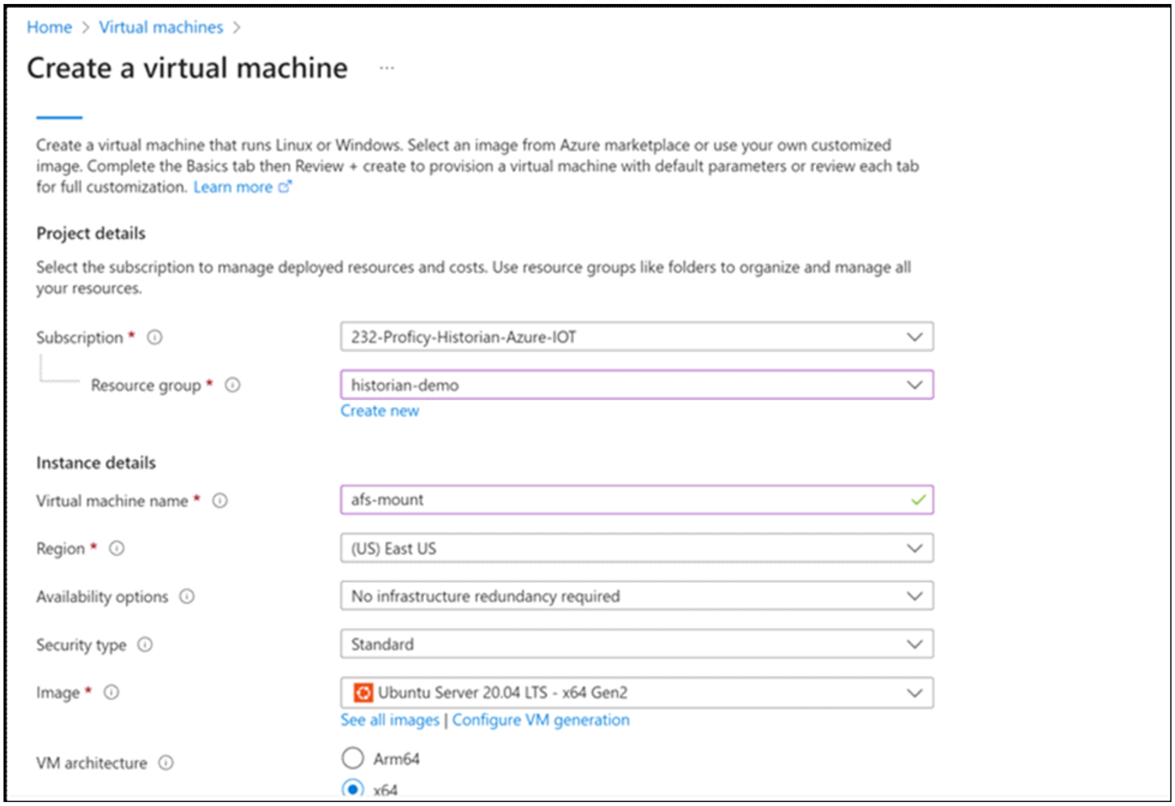
metric.

Access Archives

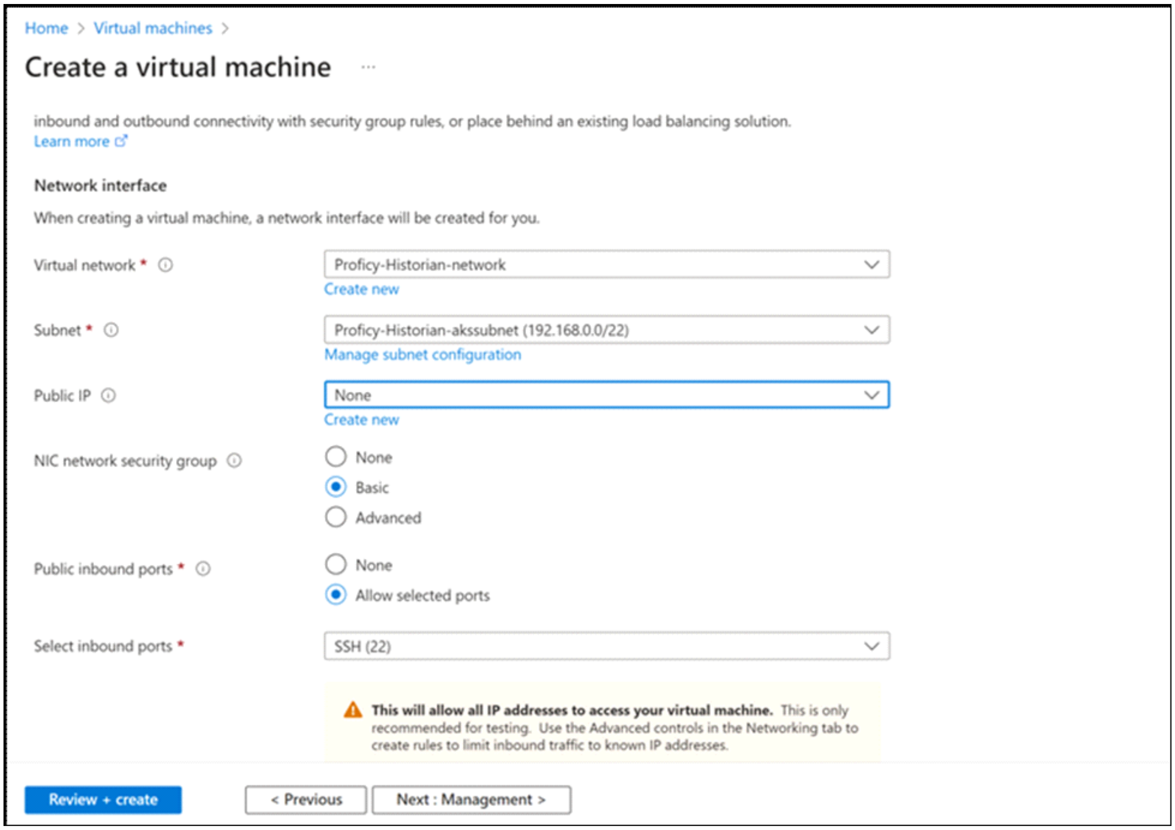
Deploy Proficy Historian for Azure Cloud *(on page 17)*.

Cloud Historian Archive Files on Azure are stored in Azure File Share. To access the files, mount Azure File Share to a Linux VM. The Linux VM should be in the same Virtual Network that was deployed with Cloud Historian. This topic describes how to access them for troubleshooting, monitoring, and so on.

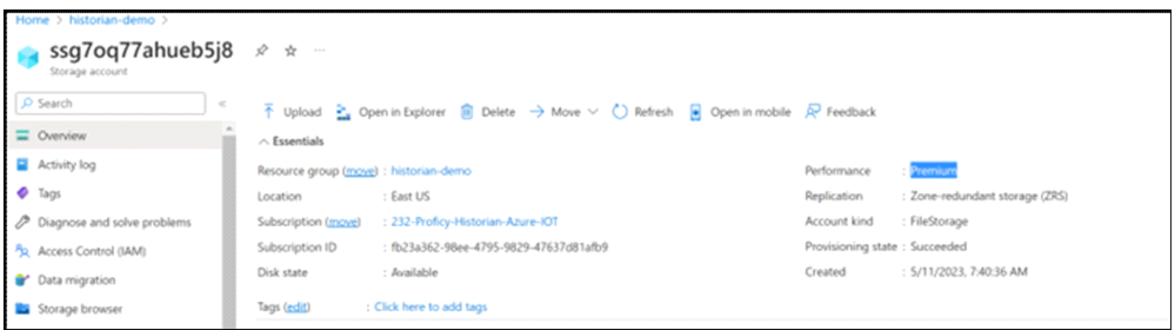
1. Launch a Linux Ubuntu Virtual Machine (VM). The **Resource Group**, **Region**, and **Virtual Network** fields should be the same as of Cloud Historian. Launch an Azure Virtual Machine.



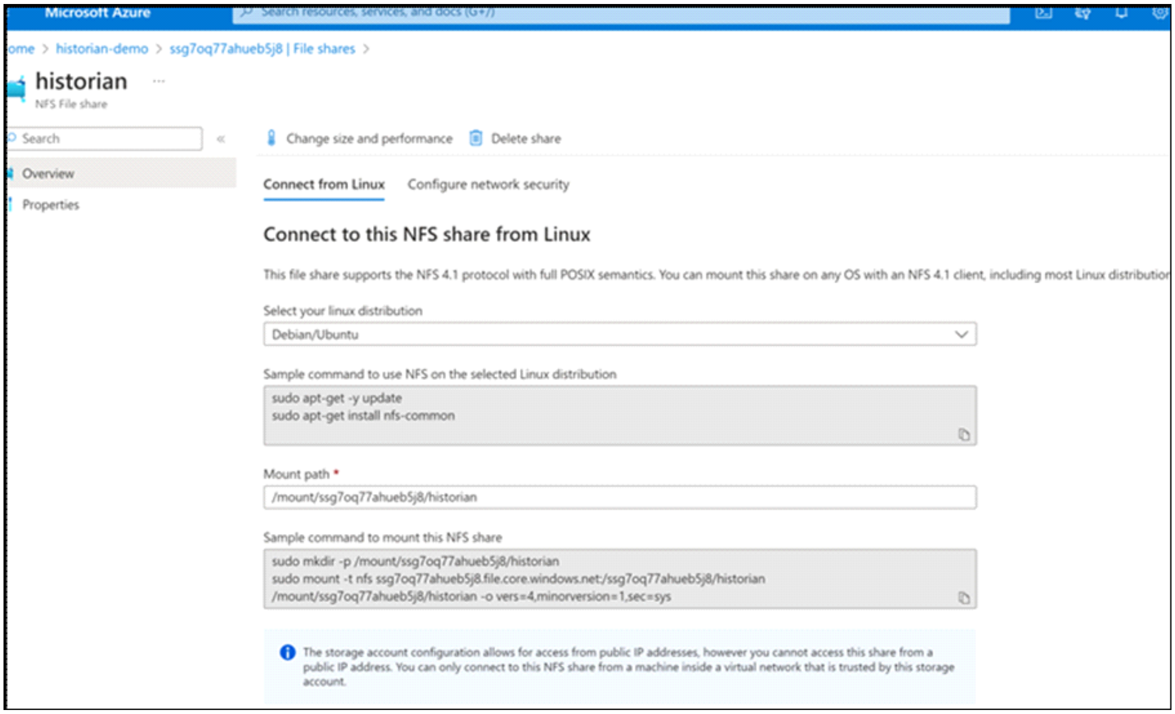
2. Select the VNet in which you have deployed Proficy Historian for Azure Cloud. In addition, in the **Subnet** field, select the public subnet of the VNet.



3. Login to the VM.
4. Go to **<Resource-Group>** -> **Storage Account** with Premium Performance. (Two storage accounts will be deployed with Cloud Historian. The one with premium performance is the one in which Archive files will be present).

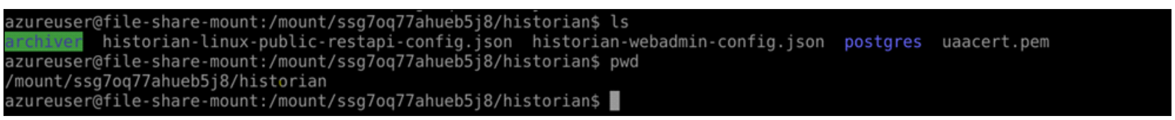


5. Go to **File Shares** -> **Historian**.

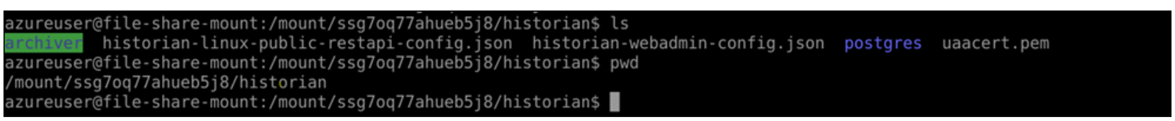


6. Follow the steps in this page to mount the File Share to the VM.

   After the file share is mounted, it looks like this:



7. Use the following command to go to archive files: `cd ./archiver/archives`



8. You can now copy the archive files to the home directory of the VM.

   The directory contains the archives and log files.

Troubleshoot Connection Issues

Unable to Start the iFIX Collector

Description: Sometimes, an error occurs while starting the iFIX collector, stating that the collector fails to start and prompting you to delete the collector. This happens because the iFIX collector service is created with the default path set to `C:\Program Files (x86)\GE\iFIX\ihFIXCollector.exe`, which does not exist.

Workaround:

1. Select No in the error message.
2. In iFIX System Configuration (SCU), set the task parameters as follows:
    - **Filename**: Enter `<installation drive>:\Program Files (x86)\GE Digital \Historian iFix Collector`.
    - **Command Line**: Enter `NOSERVICE REG=<collector name>`.