AVEVA™ InTouch HMI

2023 R2 P01

Contents

Welcome to AVEVA InTouch HMI

InTouch HMI (human-machine interface), an application that provides graphic visualization of your plant operations. This document includes information on:

- how to prepare the development environment, view the application in run time, and connect your application to the physical devices in your plant environment.
- creating and managing InTouch HMI applications locally and in a network environment.
- configuring InTouch HMI NAD and use Managed application at run time.
- viewing application graphics in a web browser, with a focus on language switching, tag viewer, and various alarm components.
- migrating and upgrading InTouch applications, and setting-up the application on tablet PC or multi monitors.
- troubleshooting to understand error messages and resolve issues with the InTouch HMI application and Web Client.
- licensing and security configurations for the InTouch HMI application and Web Client.

Legal Information

Publication date: Thursday, November 7, 2024

Publication ID: 1389207

Contact information

AVEVA Group Limited
High Cross
Madingley Road
Cambridge
CB3 0HB. UK

https://sw.aveva.com/

For information on how to contact sales and customer training, see https://sw.aveva.com/contact.

For information on how to contact technical support, see https://sw.aveva.com/support.

To access the AVEVA Knowledge and Support center, visit https://softwaresupport.aveva.com.

Release Notes

The release notes describes the new and enhanced functionality available for AVEVA InTouch HMI, providing an overview of the most significant changes.

Note: The product documentation is now available on the AVEVA documentation portal. To comply with the portal guidelines, the style and tone of the documentation content has been modified. Hence, the content available on the portal may differ slightly from the documentation delivered with the product release.

InTouch HMI 2023 R2 P01

This Readme provides information about AVEVA InTouch HMI 2023 R2 P01. Readme files from previous releases of AVEVA InTouch HMI are posted on the support site.

New Features and Enhancements

InTouch HMI 2023 R2 P01 includes a number of new features, hot fixes, and provides support for the latest versions of Microsoft products. Refer to the GCS Technology Matrix for the list of supported Microsoft products.

License requirement for InTouch WindowMaker

You can now run WindowMaker without license. For both Standalone and Managed InTouch applications, WindowMaker is allowed to run for any tag count and window count with full capability without any license check.

No tag and window count limitation in demo mode for InTouch WindowViewer

In both Standalone and Managed InTouch applications, if an application has less than or equal to 64 tags, then WindowViewer will launch in free mode (no restrictions).

Note: Free mode is not supported in Operations Control connected experience.

If application has greater than 64 tags and no valid license, then demo mode is offered. While running in demo mode, there is no tag count or window count limit, however, there is 2 hours of runtime limit. After two hours, WindowViewer will exit.

In free mode, if application is not an InTouchVew application and running in non-RDS(Remote Desktop Services) environment, then application will be always running with Read/Write capability regardless the setting of remote access configuration.

Chinese InTouch runs with either Chinese or English InTouch license

You can now run Chinese version of InTouch with English InTouch license as well.

Resolved Issues

InTouch 2023 R2 P01 includes corrections for the issues listed in the following table. These issues are listed by their Defect ID (IMS number), any assigned Service Request (SR) or Case Number with a brief description of the defect.

Defect ID	SR/Case Number	Description
IMS-2985984	960371904	When validating an application object containing Alarm Client .NET client control, the version of the .NET client control incremented in the change log without any changes to the .NET control.
IMS-3000512	960374453	When a user used a UDT with Integer Tag with SIM, the tag value in runtime defaulted to the maximum set value and could not be changed.
IMS-3003376	960375737	When any objects were duplicated using the Ctrl command (Ctrl+D), the Arrange graphics option in the Draw menu was grayed out.
IMS-3003712	960370245	In InTouch HMI 2023 P03, the navigation bar of the WindowMaker disappeared during script opening and closing.
IMS-3005355	960375747	The keyboard shortcuts Ctrl+S and Ctrl+N did not work while editing scripts.
IMS-3020843	960378303	When the function AttemptInvisibleLogon was used in a domain environment, the login process took 10-20 seconds. If the user clicked anywhere during the login process, the WindowViewer stopped responding.
IMS-3021865	960371040	While changing Trend Client properties in runtime, the pen attributes disappeared or changed.
IMS-3021934	960375339	The vers_res.inf file was corrupted

		after several power outages.
IMS-3023929	960341228	In the German installation of InTouch HMI 2023 P03, incorrect alarm bits were displayed in the Web Client, whereas the WindowViewer displayed accurate data.
IMS-3024436	960379857	The AlarmApp showed TimeLCTOAT & TimeLCT in reverse positions when the alarms are shelved.
IMS-3026661	960369259	When importing a graphic in Graphic Access API sample, an error message was displayed "Import failed: Import Failed, Schema validation of xml file failed."
IMS-3031426 IMS-3193589	960371812	Post-migration from InTouch HMI 2014 to InTouch HMI 2023 P03, the 'Me.xxxxx' references were not properly updated in Trend Control.
IMS-3032187	960382317	When using a data change script that included a tagname field, all scripts with the same tagname, but with different tagname fields, displayed the same information.
IMS-3032384	960381310	In InTouch HMI 2020 R2 SP1 and InTouch HMI 2023, users could not search for an application using the keyboard as only one application was selected at all times.
IMS-3034154	960374098	It took a long time to sign in using the smartcard authentication method.
IMS-3036283	960368485	During runtime, the custom layer in OMI Map App did not show data when no graphics were on the map.
IMS-3036416	960383270	The $OperatorName tag did not display the user's full name.
IMS-3038689	960376515	When the user tried to create two

		InTouch applications, one application with discrete tags and another with discrete tags and condition scripts, and executed the DBDump of the latter into the former, the names of the imported scripts were incorrect.
IMS-3043712	960383436	In InTouch HMI 2023 R2, use of the external providers' feature to create new InTouch tags resulted in bad runtime data.
IMS-3044947	960384126	In InTouch HMI 2023 R2, after exporting and importing JSON files, the Access Name disappeared from the Members/Instances list.
IMS-3047289	960386660	In the Japanese installation of InTouch HMI, the WindowMaker application crashed during the printing process.
IMS-3047318	960386660	In the Chinese installation of InTouch HMI, the WindowMaker application crashed during the printing process.
IMS-3048691	960380987	Post-upgrade from InTouch HMI 2023 P03 to 2023 R2, querying a Hist Trend covering a time period which includes midnight resulted in a bad Trend.
IMS-3051120	960385167	On a Windows 11 operating system using standalone InTouch HMI 2023 R2 with NAD, the WindowViewer stopped responding with the error message, "Error getting to set Entries in ACL. Error:80070534".
IMS-3054559	960384212	InTouch HMI was logging null values to Historian for the tags that had initialization value set to 0.
IMS-3055420	960383833	Post-upgrade to InTouch HMI 2023, the OPC DA connection stopped working every 24-48 hours. To resume the connection, the users has to restart Gateway and re-

		initialize their get records on the server side.
IMS-3058936	960380845	In InTouch HMI runtime, when multimonitor screen was used, the maximize option did not cover the complete resolution of the screens.
IMS-3059570	960389252	While creating a new folder or deleting a window, the tree view of Windows malfunctioned.
IMS-3066443	960390208	On a German installation of InTouch HMI 2023 P03 using the "Print project documentation" function, when printing to a text file, the tag names were missing for all variables.
IMS-3067496	960389422	InTouch 2020 R2 SP1 crashed when the Unused Tags tool was used.
IMS-3074940	960389665	The Windows and Scripts pane did not display windows and folders separately.
IMS-3074982	960391187	In a native window of WindowMaker, when using the keyboard shortcut Ctrl+D to duplicate a symbol or other object, if the contextual menu is open, a duplication loop is created.
IMS-3103706	960392721	When using the XML import Method, the WindowMaker creates some windows and stops responding.
IMS-3183859	960392314	The Galaxy Browser window did not show all properties for Trend Client Control and **Show all properties** checkbox was disabled.
IMS-3188212	960392260	When Quick Function is used with a return value, and the script is validated, the Windowmaker stops responding.
IMS-3188498	960368821	In the German installation of InTouch HMI 2023 P03, incorrect

		animation bits and alarms were displayed in the Web Client and Alarm Control Client respectively, whereas the WindowViewer displayed accurate data.
IMS-3190516	960394633	In the English installation of InTouch HMI 2020 R2 SP1 P01 on Windows 10 Pro Mac and Windows server 2019, the string formatting in the Meter Wizard was rounding the numbers differently.
IMS-3198250	960394186	Post-upgrade from InTouch HMI 2023 R2 to InTouch HMI 2023 P03, scrolling in the Windows & Scripts section of the WindowMaker automatically scrolled the canvas section.
IMS-3198575	960394458	When a new InTouch ViewApp derived template was created using a blank application template, the newly created template could not be opened. The template remained checked out, and could not be checked in. A warning message “Not a valid application template package file” was displayed.
IMS-3200524	960395975	In WindowViewer, the control panel at the bottom of the Multi pen trend was not visible when the Hide Top Header option was selected.
IMS-3202978	960390333	Post-upgrade to InTouch HMI 2023 R2, the Alarm DBLogger stopped responding and could not be opened.
IMS-3205397	960397049	Post-upgrade to InTouch HMI 2023 R2, when the network card is activated, InTouch components such as Application Manager, WindowMaker, and WindowViewer took a long time to start.
IMS-3212602	960395848	When a window was opened, the windows properties displayed the

		condition script appended to the name of the window. When the window was renamed, the condition script was replaced with the new name.
IMS-3237984	960395974	When a wizard was created and subsequently deleted in a duplicated graphic object, and this object in embedded into an InTouch window, the wizard that was deleted still appeared.
IMS-3239030	960392252	In InTouch Web Client, when two elements of a graphic overlapped, the Alarm border was still visible.
IMS-3241286	960398473	When the windows of an application are exported and then imported, many windows displayed the same names.
IMS-3241935, IMS-3241972	960400322, 960399792	WindowMaker took a long time to open. UDTs could not be exported with the error message, "UDTSerialization:Save:Exception of type 'System.OutOfMemoryException'".
IMS-3243219	960399542	In the frame windows of InTouch HMI 2023 P04, the embedded .NET controls did not work properly.
IMS-3243526	960400727	The double-click action did not work on the scrollbars of the Alarm Client Control, but worked fine on the grid rows.
IMS-3244606	960398139	On Windows 11 operating system with InTouch HMI 2023 R2, some users could not edit Bitmap embedded on a window in the Window Maker.
IMS-3250867	960341378	The Background color of the Status column in InTouch Web Client did not change dynamically.
IMS-3251060	960390517	During Update Use Counts in Introductory Demo Application, the

		WindowMaker crashed without a message in the Logger.
IMS-3256421	960394633	In the Japanese installation of InTouch HMI 2020 R2 SP1 P01 on Windows 10 Pro Mac and Windows server 2019, the string formatting in the Meter Wizard was rounding the numbers differently.
IMS-3256429	960394633	In the Chinese installation of InTouch HMI 2020 R2 SP1 P01 on Windows 10 Pro Mac and Windows server 2019, the string formatting in the Meter Wizard was rounding the numbers differently.
IMS-3256841	960400283	Some users noticed performance issues when working with UDTs.
IMS-3258646	960385856	The WindowViewer running on the NAD Client was not pushing data to the Historian.
IMS-3260877	960403218	In Alarm Client Control, when an alarm is UNACK_RTN, the ACK color did not change.
IMS-3262526	960403399	When a window is launched in WindowMaker, the CPU consumption is about 5% when the default language is set to English. Switching the language caused 100% CPU usage.
IMS-3264093	960400133	In the OMI Viewapp, the Alarm Client and Query functions didn't work until Historian server was restarted.
IMS-3267821	960401565	Post-upgrade from InTouch HMI 2023 to InTouch HMI 2023 R2, the SQL queries using SQL Access did not work.
IMS-3272910	960400499	When an instance of UDO derived template was created from a graphic with 'Expression or Reference mode' as a default custom property, and when the

		user changed the value of the custom property, the Default value mode displayed as Static.
IMS-3280020	960402885	WindowMaker printout had unusual background if Images/ Graphics were selected as HTML file.
IMS-3287663	960409213	There was a handle leak on Alarm Client Control when the RunQuery() method was called.
IMS-3292969	960406554	Post-upgrade from InTouch HMI 2020 R2 P01 to InTouch HMI 2023 R2, the Web widget could not be opened in the Azure Power Application.
IMS-3302540	960412777	Status element in Industraial Graphic did not expose properties in scripts which resulted in warnings. Also, could not add a 'dot' after the status element name in a script.
IMS-3303422	960407542	The SQLSelect() function did not work if a string was used with the WhereExpr parameter.
IMS-3307715	960414467	On a Chinese install of InTouch HMI, the help file of Industrial Graphic Editor was in German language instead of Chinese.
IMS-3337421	960420385	When tried to zoom-out an Industrial Graphic symbol or set to fit to window, the graphic changed to red-X and it could not be recovered later.
IMS-3341495	960413176	Post-upgrade from InTouch HMI 2017 U3 SP1 to InTouch HMI 2023 R2, there were issues navigating to some of the windows.

Resolved Issues for Web Client

Defect ID	SR/Case Number	Description
IMS-3023929	960341228	In the Web Client, incorrect alarm bits were displayed.
IMS-3036283	960368485	During runtime, some of the map graphics were missing from Web Client.
IMS-3188498	960368821	In the Web Client, expected alarm data intermittently did not display.
IMS-3239030	960392252	In the Web Client, an alarm border visibility issue occureed as one graphic overlapped with another graphic.
IMS-3250867	960341378	In the Web Client, the background color of the Status column did not change when the fill style animation configuration was modified.

Known Issues

This section describes known issues that remain in the release of InTouch HMI 2023 R2 P01.

Issue ID	Description
3298289	Insight screen showed blank when tried to publish data from InTouch HMI. **Workaround:** 1. Go to c:\program files (x86)\Wonderware\Intouch\InsightPublisher\x64 and open the aahCloudConfigurator.exe.config file. 2. Update the following values: • <add key="PublisherProcess" value="C:\Program Files (x86)\Wonderware\InTouch\InsightPublisher\x64\aahInTouchTagImport.exe" />

	• <add key="PublisherPrerequisite" value="C:\Program Files (x86)\Wonderware\InTouch\InsightPublisher\x64\aahPublisherPrerequisites.dll" /> 3. Relaunch the publisher.
2897808	Attempting to sign an alarm acknowledgement fails in managed InTouch with an error message warning of incorrect user credentials. The SignedAlarmAck feature is not currently supported for Managed InTouch configured for AVEVA Operations Control connected experience.
1928318	When the Supertag instances are created by importing from a .CSV file, the imported Supertags are not displayed in the Supertags pane of the WindowMaker. **Workaround:** The Supertag instances created by importing from a .CSV file can be viewed in the tag dictionary.
1826926	When a window containing a symbol with an embedded MapApp widget is viewed in WindowViewer, the MapApp does not load. **Workaround:** To view the embedded MapApp widget upon fast switching to WindowViewer: 1. Navigate to C:\Windows\System32\. 2. Locate the file 'dbghelp.dll'. Copy the file and paste it to C:\Program Files (x86)\Common Files\ArchestrA\. 3. Setup the below registry in windows. [HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ArchestrA\WebApplications\Default\SuiteLink] "V2Server"=dword:00000001 "V3Server"=dword:00000001
1765301	When an existing Custom Client Control or Web Widget is overwritten to the Cloud, the latest version is not available for other users. **Workaround**: To overwrite an existing Custom Client Control or Web Widget to the Cloud, restart the WindowMaker and delete the existing client control or Web Widget in the source repository. This allows successful upload/download of the the client control.

	Similarly, restart the WindowMaker to retrieve the latest Custom Client Control or Web Widget.
TFS-1351507	Language switching is not supported for the Carousel Widget in runtime.
TFS-1369183	A Carousel Widget containing the Web Browser Widget, QR Code Scanner Widget, Trend Client Control, Alarm Client Control or SQL Data Grid Control will not display correctly in WindowViewer, as default browser security options prevent cross-origin requests.
TFS-1372178	Carousel widgets in a managed application migrated from InTouch 2020 to InTouch 2020 R2, do not contain the latest properties. **Workaround**: On migrating a Galaxy from InTouch 2020 to InTouch 2020 R2, run the IDE as an Administrator atleast once, to allow loading the latest InTouch 2020 R2 carousel widget properties.
TFS-1371799	When an InTouch 2020 application (.aapkg) is exported and imported to InTouch 2020 R2, then the default namespace for a tag reference within a Carousel widget is not resolved in runtime. **Workaround**: Open the graphic containing the Carousel widget. Edit any property and save the graphic. The tag reference is resolved, and the graphic is displayed in runtime.
TFS-1374896	After a graphic/toolset is created, the letter case of the graphic name cannot be changed in WindowMaker. For example: INTOUCH cannot be changed to InTouch. **Workaround**: Rename the graphic to a temporary name. Rename the graphic again with the correct letter case. For example: Rename INTOUCH to ChangeName and then rename to InTouch.
TFS-1377672	Connecting many clients to a busy OPC UA server may cause various warnings and errors to be logged from the InTouch OPCUA Host process. Warning messages are for information purposes only and do not indicate any loss in functionality. Error message indicate that the operation was aborted. **Workaround**: It is recommended that client connections to the server be planned and monitored, so not to burden the server and cause connection

	failures.

Additional Information

InTouch HMI Documentation

The InTouch library is offered in three different media:

- AVEVA documentation portal, which can be accessed on your browser using the link https://docs.aveva.com/. This site presents current versions of technical product documentation of the products from AVEVA.
- Online help, which can be accessed on your browser using the AVEVA Products Help Viewer. The Online Help can also be viewed while an InTouch HMI application is running. The help is context-sensitive and is also linked to all online InTouch information.

  To launch the Help Viewer, on the Windows **Start** menu, click **AVEVA**, and then click **AVEVA Help**.

  The AVEVA Help Viewer launches in your default browser.
- Portable Document File (PDF), which can be viewed with Adobe® Reader®. Each user guide is included on the InTouch installation media as a PDF file.
  The InTouch product library consists of a set of Portable Document Files (PDF) available in the following location of the System Platform installation media:

  \AVEVA System Platform\InstallFiles\CD-Intouch\UserDocs\English

  You need Adobe Reader installed on your computer to view InTouch documentation. You can download the latest version of Adobe Reader from the Adobe Corporation web site:

  http://www.adobe.com/products/acrobat/readstep2.html

  After installing Adobe Reader, double-click a PDF file with your pointing device to view the book with Adobe Reader. You can also open a book with the Adobe Reader **Open** command from the **File** menu. Context-sensitive documentation is also available with InTouch online help. Select Help for information about the current InTouch window that you have open on your computer.

Install and upgrade InTouch HMI

InTouch HMI 2023 R2 product compatibility

Product Compatibility refers to the ability of a product to work with various versions of other products running on different network nodes. Product Compatibility also refers to the ability of a product to work with various versions of the same product running on different nodes.

Coexistence refers to same-node installation of a product with other products that support the same hardware, operating system, and SQL Server versions. Other products not listed will be covered by System Compatibility Testing unless specifically noted.

In a **single-node** environment, if you want to install multiple components on a single computer, all components must be at the same versions.

In a **node-to-node** environment, components are compatible between the current licensed version and one licensed version back.

For more information about the list of products and their compatibility and coexistence with InTouch HMI 2023 R2, refer to the GCS Technology Matrix on the AVEVA Global Customer Support (GCS) website.

In some cases, InTouch HMI is supported in newer environments than a compatible product. InTouch HMI has been tested for compatibility with a listed product only in the environments that are supported by that particular product.

System requirements

This section describes the hardware and software requirements to support System Platform products, including InTouch HMI.

Hardware requirements

System sizing guidelines

The following table provides guidelines for hardware configurations suitable for System Platform 2023 R2 software based on the size of the installation. These guidelines are subject to the limitations of your Windows operating system.

- The Intel Itanium 2 processor is not supported.
- We recommend 1280 x 1024 as the minimum display resolution for engineering tools such as the Integrated Development Environment (IDE).
- A Windows Server operating system is required for large installations with more than 50,000 I/O per node.
- SQL Server Express is not supported for installations with more than 25,000 I/O per node (small installations only).

Hardware Requirement	Small Installation (1 - 25K I/O per Node)	Medium Installation (25k - 50k I/O per Node)	Large Installation (More than 50k I/O per Node)
CPU Cores (Minimum)[1]	Greater than or equal to 2	Greater than or equal to 4	Greater than or equal to 8
RAM[1] (GB)	Greater than or equal to 4	Greater than or equal to 8	Greater than or equal to 16
Storage (GB) [2]	Greater than or equal to 100	Greater than or equal to 500	Greater than or equal to 1000
Network (Mbps)	Greater than or equal to 100	Greater than or equal to 1000	Greater than or equal to 1000

Notes:

1 In redundant environments, increase CPU and RAM to maintain a maximum of 40% typical resource utilization.

2 Minimum amount of storage needed to provide sufficient capacity for 1 week @ max frequency.

Hardware requirements notes

Windows operating systems and SQL Server versions may impose hardware requirements that exceed the minimum requirements for InTouch HMI 2023 R2. Refer to the following Microsoft Web pages for Windows and SQL Server hardware requirements:

- Windows Server 2016 System Requirements
- Windows Server 2012 R2 System Requirements
- Windows Server 2012 System Requirements
- Windows 10 System Requirements
- Windows 8.1 System Requirements

Auxiliary hardware requirements

This section describes any optional hardware requirements beyond the specific hardware requirements discussed earlier in this Readme.

Alternative Authentication Using Smart Cards

- Smart Card: Raak Technologies C2-40 Mini Driver Smart Card
- Smart Card Reader: OK 3021 USB Smart Card Readers

Operating System, .NET framework, and virtualization requirements

This section describes additional details about the supporting software for an InTouch HMI system.

Software requirement notes

Windows Operating System notes

- Windows versions 8.1, and 10 support touch screen gestures. In the Windows version 8.1,if the user swipes in from the right edge of a touch screen with a finger gesture, a set of Windows charms appears, which includes Search, Share, Start, Devices, and Settings. Moving the mouse cursor to the upper-right corner of a screen also shows the Windows charms for non-touch screens.

  Showing the set of charms is a standard feature of Windows versions 8.1 that cannot be disabled by software for a touch screen. As a result, operators can access these Windows charms and possibly unlock a dedicated touch screen view node.
- Newer operating system Service Packs (SPs) than those listed do not block the installation of products. A warning message may appear during the installation process.
- The Galaxy Repository (GR Node) can run on a client Windows operating system in a configuration in which System Platform products are installed on up to five nodes. For a system with more than five nodes, the Galaxy Repository must be installed on a computer running a Windows Server operating system.
- Development and application nodes are considered to be clients of the server GR node.
- When an operating system is upgraded on a computer, existing System Platform products must be uninstalled prior to the upgrade and then reinstalled after the upgrade. There are three exceptions. System Platform products do not need to be uninstalled when upgrading from:
  - Windows 8 to 8.1
  - Windows 8.1 to 10
  - Windows 2012 to Windows 2012 R2

.Net notes

- Versions of .NET (other than 4.x versions) can coexist, but all .NET code, including QuickScript.net scripts, run under .NET 4.8.

  For more information about .NET Framework requirements and compatibility, see .NET Framework Requirements and Compatibility .
- .NET 3.5 is installed only because the supported SQL Server versions require it. No other dependencies must exist.

Operating System notes common to AVEVA products

ActiveX Controls behavior on supported Windows Operating Systems

Due to the Data Execution Prevention (DEP) feature of Windows 7 and later operating systems, any ActiveX control built with ATL version 7.1 or earlier will fail to host or will have unpredictable behavior in InTouch 2017 UPDATE 1 either in WindowMaker or WindowViewer. For more information, refer to Tech Note 922: "Some ActiveX Controls NOT Supported in InTouch 2012 R2 (Version 10.6)" available from the Technical Support web site.

Configuring remote alarm retrieval queries

The process to configure remote alarm retrieval queries has changed for interactive applications such as InTouch HMI when running on currently-supported Windows and Windows Server operating systems.

When InTouch WindowViewer is started and generates alarms from an interactive Windows desktop session, an **AlarmViewer** control (running within InTouch HMI) on a remote node must be specially configured to query the alarms. The source alarms will not appear unless the **AlarmViewer** control's alarm query is configured.

This type of query only works for InTouch HMI as an alarm provider running in a Terminal Services session, not for InTouch HMI running in a console session.

Configure the AlarmViewer's alarm query

1. After starting InTouch WindowViewer (alarm provider), open the **Operations Control Logger** and look for the most recent string generated by **AlarmMgr**. For example: "Registering AlarmMgr with SLSSVC as AlarmMgr 253.127.148.120". The indicated IP address will be unique to your alarm-providing node. Note the IP address for Step 2.
2. In the **Alarm Query** tab of the **AlarmViewer** control on the remote computer, configure the alarm query as follows, substituting your nodename of the alarm providing InTouch HMI for "nodename" below and substituting your IP address noted in the previous step:

   \\nodename:ip_address\intouch!$system

   where nodename is the name of the node that is providing the InTouch alarm and ip_address is the IP address that you determined in step 1.
3. Test to validate that the alarms generated from the alarm-providing node are shown accurately in the **AlarmViewer** control.

Using Alarm Manager on a Single Node Running Both InTouch HMI and Application Server Alarm Providers

Starting with Microsoft Windows Vista, the operating system imposes "Session 0 Isolation" as a security enhancement. All Windows services and associated programs are required to run in Session 0, and no GUI applications are allowed to run in Session 0.

Prior to Windows Vista, Application Server and InTouch HMI WindowViewer ran in the same Windows session. Session 0 Isolation requires that Application Server and WindowViewer run in separate Windows sessions. Alarms that are reported by the Galaxy are handled by the Session 0 instance of Alarm Manager (AlarmMgr), which is now different from the Console Session instance of AlarmMgr that handles InTouch alarms. A simple alarm query in an InTouch alarm display such as

\InTouch!$System \Galaxy!Area_001

is now serviced by two separate instances of AlarmMgr -- one running in the Console Session for InTouch, another running in Session 0 for the Galaxy.

This behavior, and related behaviors and error messages resulting from the Windows operating system Session 0 changes, along with procedures to configure the Distributed Alarm System to support alarms from both InTouch and Application Server on the same computer node running with Windows Vista and later operating systems, are described in detail in TechNote 988, "AlarmMgr Support for InTouch and AppServer on Windows Vista and Later". You can download this TechNote from the Global Customer Support (GCS) website.

Remote Desktop Services (Terminal Services) behavior in Windows Server Operating Systems

Windows Server 2008 R2 and later Windows versions no longer support the /console switch as a means of starting the remote desktop (RDP) client, also known as Session 0 or Terminal Server Console session. In Windows Server 2008 or later, Session 0 is no longer an interactive session, and is reserved only for Windows services. From Windows Server 2008 and later, all remote connections are treated as remote RDP sessions regardless of /console, /admin, or any other switches used to make the connection.

This impacts InTouch HMI functionality such as Alarm Manager that depends on the Remote Desktop (Terminal Server Console) session.

In another aspect of Remote Desktop Services behavior, InTouch HMI functions such as TSEGetClientID() can return a null value with InTouch running in a remote desktop (RDP) client session. The cause of this behavior is that the relevant roles are not installed on the RDP client. You must install the "Remote Desktop Host" role in order for TSEGetClientId() and other related functions to work properly.

The impact to Application Server is minimal as most Application Server processes run as services. One impact to

Application Server is to carry forward the restriction introduced with the Windows Vista operating system which permits only one alarm provider. While both Application Server and InTouch HMI can be configured as alarm providers, only one alarm provider can be configured at any one time.

Application Server and InTouch HMI detect when the application is running in the console. In Windows Server, it implies that the application was started by a user physically at the machine. However, this behavior may require you to disable the group policy that enables Fast User Switching.

The software detects when an application is running in the console. All remote connections are treated as a remote RDP session by Windows Server, regardless of /console or /admin switches in the mstsc connection.

To disable fast user switching through the Group Policy interface

1. Click **Start** and then **Run**. The **Run** dialog box appears.
2. Enter **gpedit.msc** and click **OK**. The **Group Policy** dialog box appears.
3. Go to the following location: **Local Computer Policy > Administrative Templates > System > Logon**.
4. Set **Hide Entry Points for Fast User Switching** to **Enabled**. Enabling this policy hides the **Switch User** option in the **Logon** interface, the **Start** menu, and the **Task Manager.**
5. On the **File** menu, click **Exit** to close the **Group Policy** dialog box.

By enabling the policy, Administrators hide the Switch User button in Windows logon, in the **Start** menu, and in the **Task Manager**.

InTouch HMI Operating System notes

- Windows client (starting with Windows 7) do not support a dedicated single-node server configuration that runs one or more databases for an InTouch HMI system.
- The EnableDisableKeys() function writes to the Windows registry to enable or disable some keys on the host computer running an application in WindowViewer. For security purposes, Windows 7 and later versions of Windows prevent Standard or Power users from writing to the registry. Windows Administrators can write to the registry if not disabled by local domain security policies.

  You can configure local Windows security policies that work in conjunction with the EnableDisablekeys() script function to regulate the user's access to keys in a running InTouch application.
- As of System Platform 2014, a computer running Windows 7 or later operating systems can be configured as both an InTouch and an Application Server alarm provider. For more information, see Using Alarm Manager on a Single Node Running Both InTouch HMI and Application Server Alarm Providers on Windows Vista and Later Operating Systems.

  The following InTouch legacy script functions do not operate on 64-bit versions of Windows: WWPoke(), WWExecute(), WWRequest(), ActivateApp() and SendKeys().
- The InTouch Extensibility Toolkit might need to be started by right-clicking and selecting **Run As Administrator** on Windows 7 or later operating systems to function properly.
- The onscreen keyboard options have changed, beginning with Windows 7 and Windows Server 2008 R2 operating systems.
- Hovering to select from the Windows keyboard does not work in supported versions of Windows operating systems.

InTouch HMI View Applications and DDE support

NetDDE is not supported for InTouchView applications.

By design, an InTouchView application does not serve data to any other source, including InTouch HMI itself. When WindowViewer starts, it verifies if the application is an InTouchView application. When WindowViewer detects an InTouchView application, it does not register to become a DDE server. Industrial Graphics make use of the client layer when accessing InTouch tags, and appear as a third-party client trying to access WindowViewer as a data server. As a result, Industrial Graphics cannot communicate with InTouch tags when used with an InTouchView license.

In Industrial Graphics, **InTouch:‹tagname›** is still a valid method of referring to an InTouch tag on a local node.

InTouch HMI support for Windows user account control

System Platform 2023 R2 with InTouch HMI supports User Account Control-enabled operations on run-time nodes.

.NET Framework requirements and compatibility

IMPORTANT: System Platform 2023 R2 installs .NET 4.8 if the currently installed version of .NET is 4.7.2 or lower. If .NET 4.8 or later is installed, no change is made to the .NET Framework. Prior to upgrading your existing applications to System Platform 2023 R2, we strongly recommended that you:

- Back up your applications
- Familiarize yourself with the changes introduced by Microsoft in the .NET Framework
- Review your .NET scripts and .NET controls for any required changes

After upgrading to System Platform 2023 R2, you should perform application testing on application scripts and on script libraries used by the application to ensure they continue to function properly under .NET 4.8. We also recommend you test the upgrade in a staging system prior to upgrading your production system.

System Platform 2023 R2 leverages Microsoft .NET Framework 4. The System Platform installation program will install .NET 4.8 if your system uses version 4.7.2 or lower. No change is made if your system uses .NET 4.8 or higher. Multiple versions of the .NET Framework can coexist. On nodes where SQL Server is installed, .NET 3.5 is also installed by System Platform to support SQL Server. In this scenario, other applications you have on the same machine with dependencies on .NET 3.5 will access .NET 3.5. System Platform 2023 R2 will use .NET 4.7.1, 4.7.2, or later.

All user-supplied .NET code that runs in the context of InTouch HMI and Application Server requires .NET Framework 4.8 or higher. Although .NET Framework 4.5.1 (and later) is highly compatible with applications that are built with earlier .NET Framework versions, you may have to update your scripts, if your .NET scripts were created prior to System Platform 2014. These changes may also affect .NET controls developed with .NET 3.5.

In application scripting, some .Net codes could fail if not using proper text encoding, and could cause a script to exit without completion. The UTF8Encoder is the default BinaryStream decoder in .Net 4.5. To enable an application script to decode ASCII XML data, for example, insert the following snippet:

```
   BinaryReader streamReader = new BinaryReader(ms, new ASCIIEncoding());
```

To learn more about changes introduced in different versions of the .NET Framework, refer to the following Microsoft resources:

What's New in the .NET Framework: http://msdn.microsoft.com/en-us/library/ms171868%28v=vs.110%29.aspx

What's obsolete in the .NET Framework Class Library: https://msdn.microsoft.com/en-us/library/ee461502%28v=vs.110%29.aspx

Migration Guide to the .NET Framework 4.6 and 4.5: https://msdn.microsoft.com/en-us/library/ff657133%28v=vs.110%29

.NET Framework 4 Migration Issues: http://msdn.microsoft.com/en-us/library/ee941656%28v=vs.100%29

Virtualization host support

See the Global Customer Support (GCS) Technology Matrix for supported virtualization environments.

SQL server requirements

SQL server requirements for all components

See the GCS Technology Matrix for SQL Server requirements for System Platform 2023 R2 components, including InTouch HMI 2023 R2.

Note: Match SQL Server version to operating system version.

We recommend that you install and configure the supported SQL Server version before you begin the System Platform installation program, especially if you expect Information Server to coexist, either on initial or subsequent installations, with the InTouch HMI, Application Server, or the Historian Server.

The System Platform installer will install all prerequisites except the SQL Server requirement for installing the Historian Server. If you select the Historian Server for installation, and if the supported version of SQL Server is not already installed, you must exit the installation program, install the supported SQL Server version, and then resume the installation.

Considerations for SQL server

- **Alarm DB Logger** To use the Alarm DB Logger, you must set the default SQL Server authentication mode from Windows-based to Mixed Mode.
- **SQL Server Configuration Rights**: While installing System Platform, configuration changes to SQL Server are required. If the installer does not have SQL Server administrative rights, the proper configuration of SQL Server may not be applied. A post-install aaConfig SQL utility enables you to verify if the SQL configuration has been performed.
- **Maximum Server Memory**: After installing SQL Server, Use SQL Server Management Studio to confirm that the Maximum Server Memory is configured to approximately 65% of the computer's total available RAM. By default SQL Server does not clamp this setting. The System Platform installation process will attempt to adjust it if it has the appropriate rights to configure SQL Server.
- **Restoring a CAB**: You cannot restore a Galaxy .cab file backed up in SQL Server 2012 or SQL Server 2008 to a node with an earlier version of SQL Server. For example, you cannot restore a Galaxy .cab file backed up in SQL Server 2012 to a node with SQL Server 2008. For example, you cannot restore a Galaxy .cab file backed up on SQL Server 2008 to a node with SQL Server 2005. SQL Server database backups from SQL Server 2005 forward are not backward-compatible. Attempting such a restore initializes, and the progress dialog box quickly displays 100% completion of restoring the database, but the Galaxy .cab restore operation does not actually complete, and no Galaxy Repository is created. When you start the IDE, the Galaxy is blank.
- **Migrating SQL Server Versions:** You can migrate a SQL Server database to a later version from any of two prior versions. For example, you can migrate to SQL Server 2012 from SQL Server versions 2008 or 2005. You can migrate to SQL Server 2008 from SQL Server versions 2005 or 2000. You cannot migrate directly from SQL Server 2000 to SQL Server 2012. Such a migration requires an intermediate version installation. For more information and helpful procedures, see the following Microsoft references:
  - Upgrade to SQL Server 2014
  - Migrate SQL Server 2000 to SQL Server 2012

- Migrating to SQL Server 2008 R2
- Migrating to SQL Server 2008
- T-SQL scripts to migrate SQL server 2000 user databases to SQL server 2012
- ALTER DATABASE Compatibility Level (Transact-SQL) SQL Server 2012
- ALTER DATABASE Compatibility Level (Transact-SQL) SQL Server 2008 R2

- **SQL Server Rights Requirements**: SQL Server 2008 and later versions do not automatically create the BUILTIN\Administrators role delivered in SQL Server 2005. Because of this change to SQL Server, the Application Server 2014 installation process will create the necessary operating system user group (aaAdministrators) as well as the necessary SQL Server role. This automated process will provide the rights required to allow operations within the Galaxy Repository without the need for blanket BUILTIN\Administrator rights. The aaAdministrators group must be present and enabled. If you accidentally delete the aaAdministrators group from the Windows operating system, you can run either of two options to restore it:
  - Run the Change Network Utility from the Windows Start menu.
  - Run the aaConfig SQL Utility from the Windows Start menu.

If you accidentally delete the aaAdministrators group from the SQL Server security logins, you must run the aaConfig SQL Utility to restore it. Refer to the Application Server User's Guide, About ArchestrA User Accounts, for further information and procedures to restore the aaAdministrators group.

Considerations for SQL server express

- SQL Server Express is supported for use on a Galaxy Repository node and recommended for use only with small systems.
- If you plan to modify an InTouch HMI installation which has SQL Server Express installed, then add Application Server with full SQL Server, you must install the full SQL Server prior to modifying the InTouch HMI installation or installing Application Server.
- If you plan to use SQL Server Express with Information Server and System Platform on the same node, the following limitations apply:
  - Install Information Server first with the SQL Server Express default instance name set to "SQLEXPRESS". Then, install InTouch HMI. This scenario will work without issues because each component uses a dedicated instance of SQL Server Express.
  - If you install InTouch HMI first, the System Platform installer silently installs and configures SQL Server Express. Installing Information Server will fail during configuration with an error message stating: "SQL Server client components not found."

    To work around this issue, configure Information Server to use an instance of SQL Server Express (or a non-Express edition) on a remote node.
  - The computing capacity of SQL Server Express is limited to the lesser of one CPU socket or four processor cores.
- For InTouch HMI installations, the System Platform installer installs SQL Server Express, if these conditions are met:
  - No other SQL Server elements are installed on the computer at the time of installation.
  - You select only the **InTouch Development and Runtime** System Platform installation option. When you select **InTouch Development and Runtime**, a Galaxy Repository will be installed.
  - The installation program includes an option to not install SQL Server on an InTouch development

computer. When the option is selected, it prevents SQL Server from being installed. Clear this option to automatically install SQL Server on a development computer.

InTouch Access Anywhere requirements

The following sections describe the requirements to run InTouch Access Anywhere Server and supported browsers. For detailed information, see the InTouch Access Anywhere documentation.

You can determine the version of InTouch Access Anywhere Server installed on a computer using **Programs and Features** from the Windows Control Panel.

Requirements

The InTouch Access Anywhere Server can be installed as a System Platform installation option. The InTouch Access Anywhere Secure Gateway is an optional component installed on a separate computer within a network DMZ.

The following requirements must be met before you install the InTouch Access Anywhere Server or a Secure Gateway.

- InTouch Access Anywhere server must be installed on a 64-bit version of Windows that is supported by InTouch HMI 2012 R2 (version 10.6) or later.
- InTouch Access Anywhere must be installed on the same RDP Server computer as InTouch HMI 2012 R2 (version 10.6) or later.
- Remote Desktop (RDP) must be enabled and properly configured on the host computer. For information and guidance, see Tech Note 782, "Installing Remote Desktop Services on Windows 2008 Server R2 for AVEVA Products". You also can refer to http://technet.microsoft.com/en-us/library/dd883253.aspx.
- You must have an InTouch 2012 R2 TSE (RDS) or newer license activated. When InTouch is launched by InTouch Access Anywhere, this RDS license will be consumed per browser session. It will be released when InTouch is closed by InTouch Access Anywhere.
- Per-Device licenses are not supported.

Supported InTouch Access Anywhere browsers

InTouch Access Anywhere can run on smart phones, tablets, laptop computers, or any other device that supports an HTML5-compliant web browser. Browsers validated to support InTouch Access Anywhere include the following:

- Apple Safari
- Google Chrome
- Microsoft Edge
- Microsoft Internet Explorer
- Mozilla Firefox
- Opera

Functionally compatible browsers

Older versions of the listed browsers are functionally compatible with InTouch Access Anywhere. You may be able to use the following older versions of these browsers, but specific behaviors are unknown because no formal testing has been done with InTouch Access Anywhere.

- Apple Safari versions 5 and 6
- Google Chrome versions 11-27
- Microsoft Internet Explorer versions 7 and 8.
- Mozilla Firefox versions 4-21
- Opera versions 11-14

You are strongly encouraged to upgrade to one of the supported browser versions listed in Browsers Tested with WindowViewer. For a complete list of considerations to run an InTouch application with InTouch Access Anywhere on a portable device, see the InTouch Access Anywhere User Guide.

Install InTouch HMI

The major decision you must make when you install InTouch HMI is whether to install the InTouch development and run-time components, or the run-time components alone. The installation program guides you in selecting the features you want, verifying or modifying your selections, installing prerequisite software, and then installing InTouch HMI and the IDE if you chose to install development components.

What you need to install InTouch HMI

You must have the following materials ready to install InTouch HMI:

- Installation media.
- Computer that meets the hardware and software requirements listed in this Readme.

  Important: Use the hardware and software requirements listed in the Application Server Readme if you are going to install the IDE.

Prerequisites

The installation program analyzes the software installed on your computer and lists any required software that is not installed.

Note: At the start of the installation, the prerequisites check is system-specific rather than product-specific.

The following prerequisites, if not already present on your system, will be installed by the System Platform installation program:

- Windows Installer 4.5
- .Microsoft .NET Framework 4.8
- SQL 2022 Express Core

  SQL 2022 Express Core is installed automatically if either of the following conditions are met:

- No version of SQL Server is installed on the host computer.
- You select only InTouch Development and Runtime during the System Platform installation. You must not select any Application Server components.

The installation program installs both system-specific and product-specific prerequisites. You do not have to exit from the installation program to install the prerequisite software. For more information about System Platform installation prerequisites, see the System Platform Installation Guide.

About the ArchestrA user account

The ArchestrA user account is a user name and password combination that enables inter-node communication between all computers in an Application environment. You must specify the same user account on every node when you install the System Platform components for the first time on computers that communicate with each other.

WARNING! The ArchestrA user account is a Windows operating system account located on the local computer or on a domain. Do not delete this account with operating system account management tools. If you do, ArchestrA-enabled software may stop functioning properly.

If the ArchestrA user account has not previously been configured, you will be prompted to configure it during the InTouch HMI installation. You must specify a user name, password, and domain.

If you choose to use an existing user account, it should meet the following requirements:

- User account with a permanent password that does not expire.
- User account with network privileges if InTouch HMI 2023 R2 will be installed on a multi-node system
- User account in which the password cannot be changed.
- User account that is a member of the local Administrators group.

After you install the InTouch HMI, you can use the Change Network Account utility to change or recreate the ArchestrA user account. This utility is accessible from the Start menu in the AVEVA Common folder after you install InTouch. You must have Administrator privileges on the computer to make changes with the Change Network Account utility. For more information, see the Change Network Account utility help.

Note: If you recreate the user account using the Change Network Account utility, the Microsoft Windows security component on the computer can take several minutes to update this information on the ArchestrA Galaxy node. Until that occurs, the ArchestrA component may not function properly. Restarting the Galaxy node updates this information immediately.

Perform the InTouch HMI installation

Before you start installing InTouch HMI, you should have a clear idea whether you want to develop, deploy, and publish your own applications, or require only the InTouch run time, to run applications already created and deployed.

The following procedure will guide you through the installation, with information specific to the InTouch HMI.

To install the InTouch HMI

1. Locate the installer file (Setup.exe) in the installation media. Double-click the **Setup.exe** file or right-click **Setup.exe** and select **Run as administrator**.

2. Follow the prompts to start the installation and install system prerequisites, as necessary.

   The installation program prompts you to select an installation type: either product-based selection or installation by computer roles. As you are only installing InTouch HMI, click the product-based selection as your installation type.
3. Choose the components you want to install.
   a. If you choose **InTouch Run Time Only,** the following will be installed:
   - InTouch Run Time
   - InTouch documentation
   - Alarm DB Logger
   - OI Gateway (as a silent installation)
   - Application Server Bootstrap
   - InTouch Supplemental Components: InTouch Recipe Manager, InTouch SQL Access, and Symbol Factory

     Note: The 16 PenTrend supplementary component is not installed by default. You must select the **Customize Installation** option and select **InTouch 16 PenTrend** from the product list to install it as part of the InTouch HMI installation procedure.
   - AVEVA Enterprise Licensing Platform
   - InTouch Web Client
   - AVEVA System Monitor - Sentinel Agent
   - ASB Runtime Components

   a. If you choose **InTouch Development and Run Time**, the following will be installed:
   - All items listed under step a in these instructions.
   - InTouch Development
   - ASB Service Repository
   - InTouch demo applications
4. When the installation prompts you to verify your selection, select **Customize Installation** if you want to add or remove components of your InTouch installation.
5. After you verify and customize your selection, you are prompted to select the language you want to install. English is selected by default, but you can choose another supported language from the list.
6. Follow the subsequent installation steps to:
   a. Accept the End User License Agreement.
   b. Create a new ArchestrA user account, or specify an existing account.
   c. Install any prerequisite software components not yet installed on the computer.
   d. The installation proceeds to completion. You must restart the computer after completing the installation.

Activate your InTouch license

InTouch uses the AVEVA Enterprise License Server to make licenses available. The AVEVA Enterprise License Manager manages one or more License Servers.

You use the AVEVA Enterprise License Manager interface to make licenses available to InTouch. First, import the entitlement XML file received upon purchase of the license. Then, select which licenses on the entitlement to activate on the License Server. After activation, the license becomes available to WindowMaker or

WindowViewer upon start up. The license is released when InTouch is shut down.

InTouch will run in Demo Mode if it cannot consume a valid license at start up time.

To install AVEVA Enterprise licensing, see the AVEVA Enterprise Licensing Guide.

Antivirus software exclusions

Antivirus software should be implemented as part of a defense-in-depth strategy to secure your industrial control systems.

Visit https://softwaresupport.aveva.com/ to get the latest information on antivirus software recommendations.

For detailed English installation instructions, see the System Platform Installation Guide (SP_Install_Guide.pdf).

Modifying, Repairing, or Removing the InTouch HMI

1. Open the **Modify, Repair or Remove Installation** dialog box by doing either of the following:
   - Run Setup.exe from the installation media.
   - Access the Windows **Control Panel** and select the **Programs and Features** option. The list of software installed on your computer appears. Select the InTouch component
2. Click the **Uninstall/Change** button. The **Modify, Repair or Remove Installation** dialog box appears.
3. Follow the prompts to modify, repair, or remove InTouch. The name of the **Uninstall/Change** button varies depending on the version of Windows installed on your computer.

   Note: Modern applications can no longer be created, edited, or run after Application Server is uninstalled from the same computer that also hosts InTouch run time and development components. Modern applications no longer work because the Galaxy Repository required for Modern applications is removed when Application Server is uninstalled.

Upgrade to the latest version from an earlier version of InTouch HMI

Upgrading refers to installation on a computer that already has a previous version. An upgrade removes the previous version of InTouch HMI and installs the latest version.

After the hardware and software requirements are met, you can upgrade directly to InTouch HMI 2023 R2 SP1 from InTouch HMI Version 11.1 SP1 or later, provided the prior version was installed on a 64-bit operating system.

You can only upgrade System Platform products that are already installed. You cannot install any new System Platform products by an upgrade process.

To upgrade from a previous version of the InTouch HMI

1. In the installation media, run setup.exe to start the set-up program. The upgrade feature dialog box appears.
2. Follow the prompts to complete the upgrade.

Upgrade Notes

- **Important Note**: If you plan to upgrade System Platform on a computer that has InTouch Access Anywhere Server installed, you must first uninstall the InTouch Access Anywhere Server. Then, upgrade System Platform and finally reinstall InTouch Access Anywhere Server.
  Back up any custom configuration of the installed instance of InTouch Access Anywhere before you uninstall it.
- InTouch Development (WindowMaker) is now installed with support for Industrial Graphics. This includes the Integrated Development Environment (IDE), Bootstrap, Galaxy Repository (GR), and FS Gateway.
- If Application Server is installed on the same node as InTouch HMI, both must be upgraded to System Platform Version 20.0.000. You must upgrade Application Server before upgrading InTouch HMI.
- If you are using managed InTouch applications with Industrial Graphics from Application Server 3.1 SP3 or earlier, run the Industrial Graphics Analysis and Repair Utility before installing Application Server to ensure that all issues reported by the utility are addressed.

Migrate applications to InTouch HMI 2023 R2

Migration converts applications created with earlier versions of InTouch HMI to the current version. You can migrate applications to InTouch HMI 2023 R2 that were developed with InTouch HMI version 7.11 P07 or later.

Note: For InTouch Version 10.2 and newer, refer to the System Platform migration table. InTouch cannot be installed or upgraded as a single product since Version 10.2. Therefore all installed products on a node must be upgraded at the same time.

SPCPro was a Statistical Process Control (SPC) utility available with InTouch HMI prior to Version 11.0. SPCPro is no longer supported by InTouch HMI. We recommend that that you migrate your SPCPro databases to the latest version of QI Analyst if you still use SPCPro with InTouch. QI Analyst provides support for the latest versions of operating systems and databases.

QI Analyst also includes a utility to migrate your SPCPro database to a QI Analyst database, leaving the existing SPCPro database intact. Please contact your local sales representative for further information.

Important Note: We strongly recommend that you migrate your current SPCPro databases to QI Analyst before installing System Platform Version 20.0.000 with InTouch HMI. InTouch no longer includes SPCPro and will not open or migrate applications containing SPCPro components.

Get Started

Introduction to AVEVA InTouch HMI

AVEVA InTouch HMI continues the tradition of market leadership in Human Machine Interface (HMI) applications. This material gives you a quick overview of the major features of InTouch HMI and explains the essential tasks to create several types of InTouch applications.

Install InTouch HMI

The simplified installation process makes installing InTouch easier than ever.

The major decision you must make when you install InTouch HMI is whether to install the InTouch development and run-time components, or the run-time components alone.

The installation program guides you in selecting the features you want, verifying or modifying your selections, installing prerequisite software, and then installing InTouch HMI. For detailed information about installation, refer to the System Platform Installation Guide.

InTouch HMI Licensing

A valid product license is required to enabled InTouch functionality. The AVEVA License Server and AVEVA License Manager are automatically selected when you select InTouch during installation.

Note: If you are using a workgroup, the AVEVA License Manager and License Server must be installed on the same node.

You will need to activate your InTouch licenses before using WindowMaker or WindowViewer.

AVEVA Flex Subscription Program has been introduced to reduce the licensable items from hundreds of product combinations to just three: Edge, Supervisory, and Enterprise.

This model is based on two core concepts: Named Users and Entitlement Seats. Each end-user of the system is and known via their AVEVA ID. Access to software within a specific Flex-based offer is controlled by assignment of an AVEVA ID to an Entitlement Seat. You can manage the AVEVA ID and assignment of Entitlement Seat in the CONNECT portal.

For detailed information about Flex license model and license activation, refer to the *AVEVA Enterprise Licensing Guide*. It is also available on the AVEVA License Manager node as a PDF file, under the AVEVA start directory, after installation is complete.

Work with InTouch HMI

An InTouch HMI application shows a graphical representation of a manufacturing or process environment. The tools, materials, and processes used to create a product appear as visual elements in an application's windows. This chapter describes the steps to create the following types of InTouch HMI applications:

- Standalone Applications
- Managed Applications

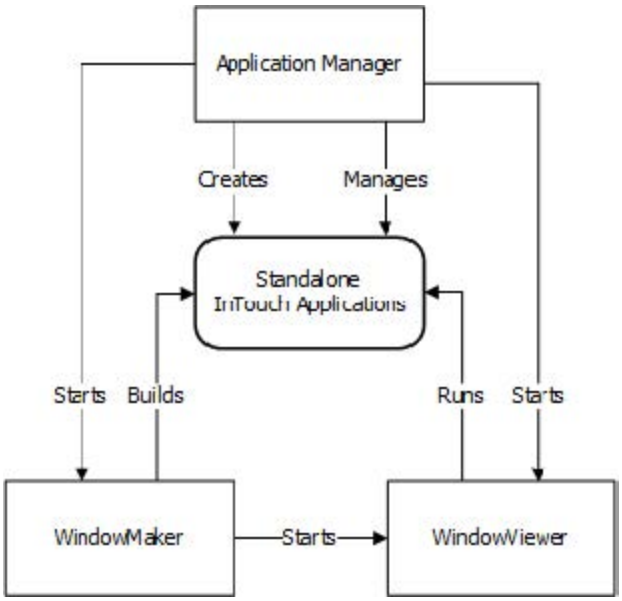
Compare different types of InTouch HMI applications

The following table shows some major similarities and differences between different types of InTouch applications.

Tasks	Types of InTouch HMI Applications	
	Standalone	Managed
Main Use	Tag based, native symbols and Industrial Graphics	Object based and Industrial Graphics
Create an Application	Application Manager	IDE • New applications • Import standalone applications • Import SmartSymbols
Edit an Application	WindowMaker started from Application Manager	WindowMaker started from the IDE
Delete an Application	Delete folder and remove from Application Manager	Delete InTouchViewApp template from IDE
Publish an Application	Yes, from WindowMaker	Yes, from IDE
Create Industrial Graphics	Yes	Yes
Incorporate Industrial Graphics	Yes, can be added, edited, and viewed from WindowMaker	Yes, can be added, edited, and viewed from the IDE
Incorporate Application Objects	No	Yes

Create standalone applications

The following figure shows the components that you would use to create, manage, build and run standalone InTouch HMI applications.



Use InTouch HMI Application Manager to create and manage standalone applications. You can build standalone applications with WindowMaker and run them from WindowViewer.

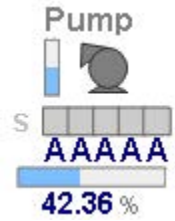
Standalone applications give you the capability to easily integrate Industrial Graphics directly into your applications. You can switch between WindowMaker and WindowViewer to test or run your applications and switch back to make modifications to your applications.

Add Industrial Graphics to applications

After you have created an application, WindowMaker's Industrial Graphic Toolbox includes separate folders containing the Industrial Graphic Library and Situational Awareness Library of predefined symbols. The Industrial Graphic Library contains realistic symbols of standard industrial objects.

Situational Awareness Library symbols are configurable symbols designed to enhance an operator's situational awareness of current process conditions using highly efficient visual techniques and best practices.

Situational Awareness Library symbols have a simplified look and provide minimum visual detail to efficiently convey their functional purpose and status without showing irrelevant information to operators.



Most Situational Awareness Library symbols are designed as Symbol Wizards that incorporate multiple visual and functional configurations in each symbol. Selecting a configuration for a symbol is a simple matter of selecting options from a list without the burden of extensive design work. Also, Situational Awareness Library symbols provide faster application run-time performance because of their lightweight design and simple appearance.

Migrate existing modern applications

The functionality of modern applications has been merged to the new standalone application. For detailed

instructions to migrate existing modern application, see the *AVEVA InTouch HMI Application Maintenance Guide*.

Create managed applications

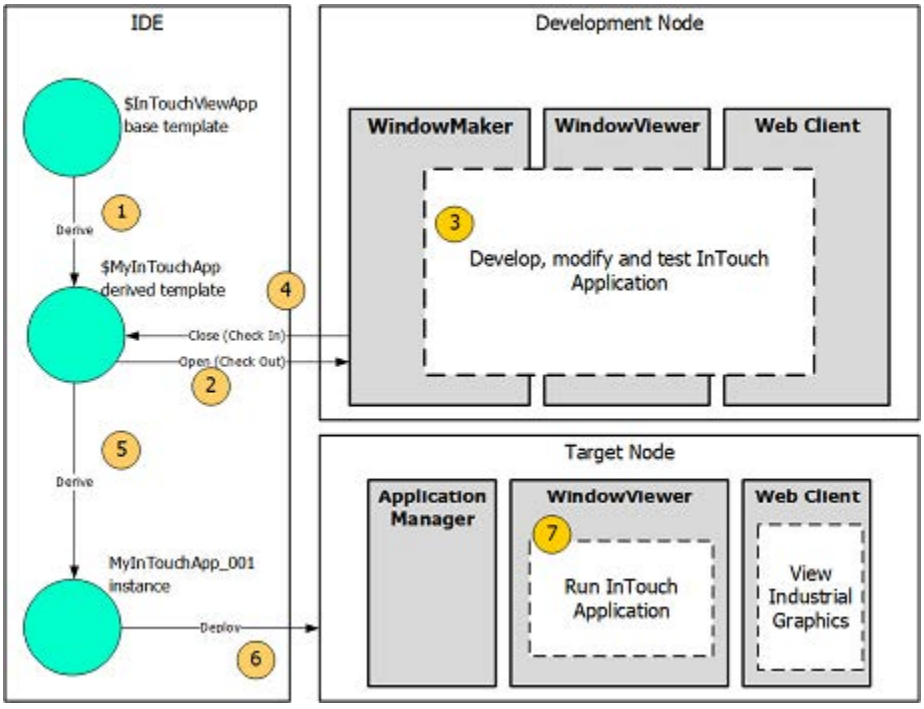
InTouch HMI shares the Integrated Development Environment (IDE) with Application Server. You can also create Managed InTouch applications from the IDE using Industrial Graphics and automation objects.

The IDE includes a suite of graphic tools and automation objects to build simple or complex system environments. Using automation objects, you can integrate your InTouch HMI applications much more easily into the System Platform. Also, you can embed pre-built Industrial Graphics and Situational Awareness Library symbols into your applications or use a wide assortment of tools from the Industrial Graphic Toolbox to create your own symbols.

Using the IDE to manage your InTouch applications, you can:

- View which applications are running on individual Galaxy nodes.
- Use a central repository to manage applications.
- Deploy application changes to WindowViewer nodes running InTouch applications.

The following figure shows the integration of the IDE with traditional InTouch HMI components. The figure shows the steps to create a managed InTouch HMI application with the IDE.



1. Create a managed InTouch application in the IDE by deriving a template from the $InTouchViewApp base template.

   You create a managed application on one node of the Galaxy with WindowMaker. Then, you deploy it to one or more target nodes running WindowViewer.
2. Open the managed application in WindowMaker.
3. Develop your InTouch application in WindowMaker. If needed, switch to WindowViewer to test the

application.

4. Save the changes to the InTouch application.
5. Derive an instance of the managed application and select the nodes to deploy the application.
6. Deploy the InTouch application to the target nodes running WindowViewer in the Galaxy.
7. Run the application in WindowViewer on target nodes.

Integrate application objects with InTouch HMI

A Galaxy is your specific production environment to run your managed InTouch applications. A Galaxy includes all computers and components. It is a collection of platforms, engines, application objects, templates, instances, and attributes you define as the parts of your specific application. This collection is stored in a Galaxy database on a node called the Galaxy Repository (GR).

Application Server manages your InTouch applications with a specific type of application object called the *InTouchViewApp* application object, which is derived from the $InTouchViewApp base template.

After you derive a new InTouchViewApp template from the $InTouchViewApp base template, you can associate the InTouchViewApp template with an InTouch application by:

- Creating a new InTouch application.
- Importing a stand-alone InTouch application.

An InTouchViewApp template represents one specific InTouch application at design time and cannot be executed at run time.

You deploy an *instance* of your derived InTouchViewApp template to a target node to run your InTouch applications. The target node is the node on which the managed InTouch application can run in WindowViewer. To distribute your InTouch application, you can create multiple instances of the same InTouchViewApp template and deploy them to multiple nodes.

Work with the Industrial Graphic Editor

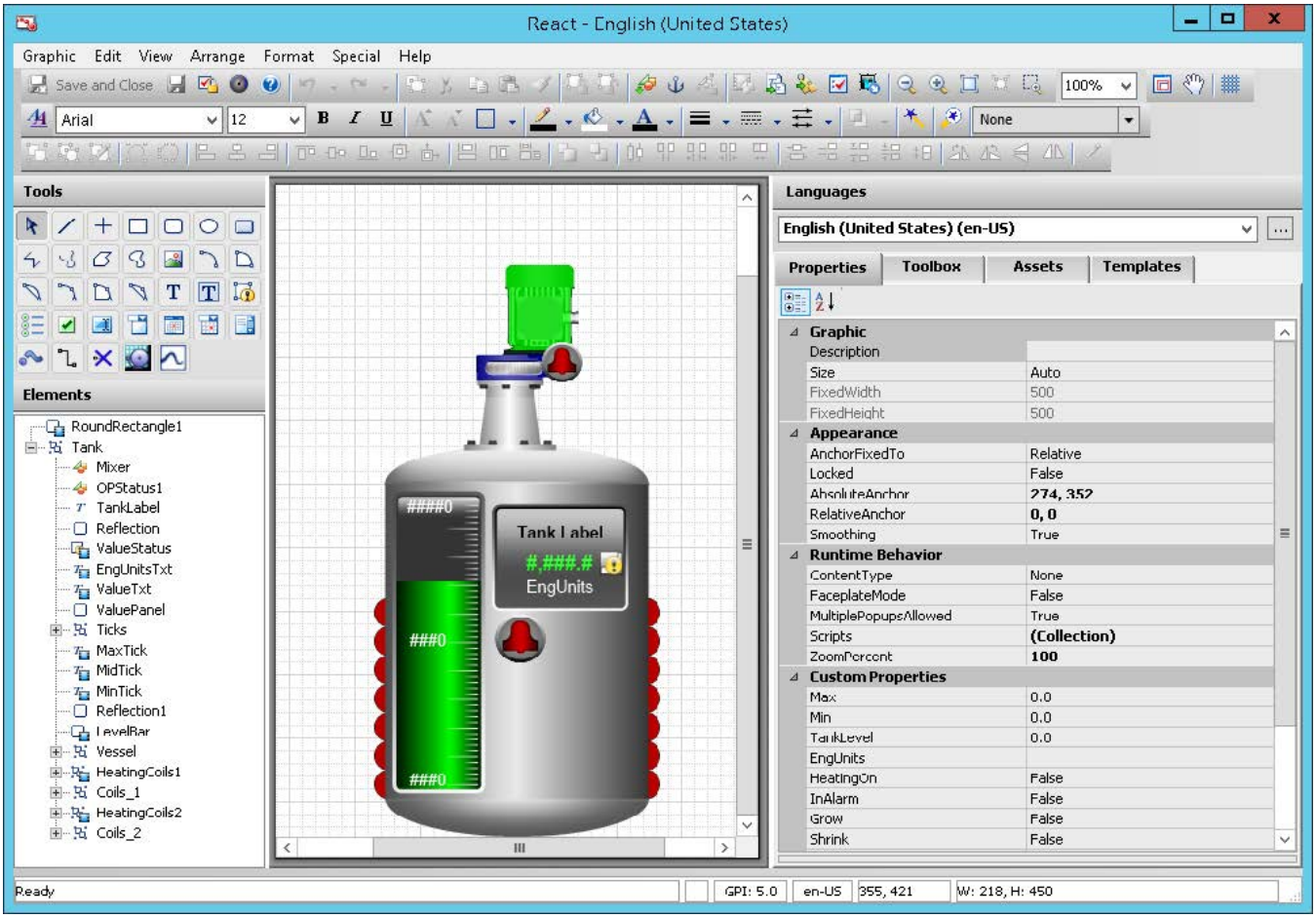
You can create Industrial Graphics from basic elements such as rectangles, lines, circles, or text much like graphics created from WindowMaker. The Industrial Graphic Editor also includes other graphic tools to create more complex drawing elements like closed curves, chords, and Windows controls.

Industrial Graphics are graphics you can add to a Modern application window to visualize data in a production process. You create Industrial Graphics in the Industrial Graphic Editor. You select a basic graphical object called an element from the **Tools** panel and place it on the drawing area called the canvas. Then, you can change the appearance of your drawn elements either by accessing their properties directly, or by modifying their physical

appearance. You can configure the elements or the symbol with animations.

The Industrial Graphic Editor also supports import of Scalable Vector Graphics (SVG) as Industrial Graphics.

The following figure shows the various tools and palettes of the Industrial Graphic Editor that you use to create and customize symbols.



When you embed an Industrial Graphic into an InTouch window and the symbol is contained in an Automation template, you can easily create a new instance of the Automation object. The embedded symbol automatically references the new object.

After you build your managed application from the IDE, you can publish it. A published InTouch application is no longer associated with the InTouchViewApp template and cannot be edited from the IDE. But, a published InTouch application can still communicate with the Galaxy by any embedded Industrial Graphic. You can write data back to the Galaxy or visualize Galaxy data with the Industrial Graphic.

Connectivity with Gateway Communication Driver and OPC

Gateway Communication Driver is included with the InTouch HMI. When the InTouch HMI is installed, an InTouch Access Name is created for Gateway Communication Driver, pointing by default to the localhost.

Gateway Communication Driver streamlines OLE for Process Control (OPC) and OPC Unified Architecture (OPC UA) setup, enhancing device integration. OI Server Manager also is included, providing the Operations Control Management Console (OCMC) as an interface for configuring the Gateway Communication Driver and OPC.

Gateway Communication Driver requires configuration using the OCMC. OPC and OPC UA Servers require a

separate installation.

You can view the list of all the OPC UA server connections configured in the Gateway Communication Driver present on the same machine. You can create the tags by dragging and dropping the tags from the OPC UA server list to the Model - Tagname pane.

InTouch HMI as an OPC UA Server

InTouch HMI can be configured to act as an OPC UA server allowing clients to connect and access tag data. For more information, refer to the InTouch HMI help.

Create a standalone application

All configuration steps to use Industrial Graphics are completed from InTouch WindowMaker. You do not need to use the IDE to create a standalone application. When you are creating a window for an application, you simply drag Industrial Graphics or Situational Awareness Library symbols directly from WindowMaker's Industrial Graphic Toolbox into a window.

Create a standalone application from InTouch Application Manager

1. Select **Start** on your Windows desktop and search for **InTouch HMI Application Manager**.
2. Select **New** by one of the following methods:
   a. Select **New** from the **File** menu.
   b. Right-click within Application Manager and select **New** from the shortcut menu.
   c. Select the **New** icon from the menu bar.
   d. Press the Ctrl + N keys.

   The **Create New Application** wizard appears.
3. Select a **Template**, and select **Next** .

   The **Enter Application Details** page displays the fields to enter the application name, directory name, application path, application target resolution and description.
4. Enter the details and specify a target resolution if different than the default screen resolution. Name the application *Chocolate Milk*.
   a. Select from a list of predefined target resolutions or select **Custom**. The Pixel width and height fields become editable.
5. Select **Finish**.

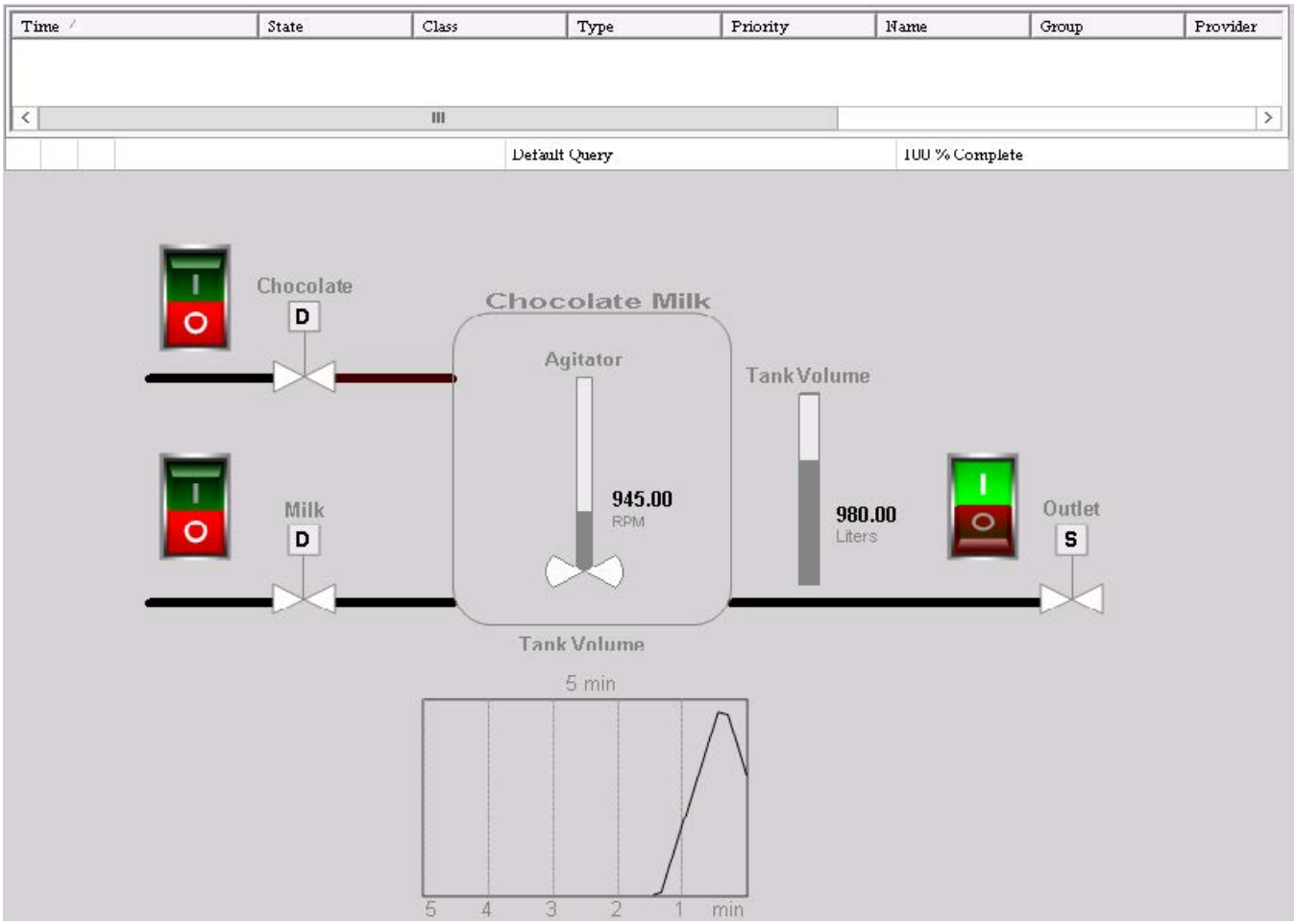
After an application is created, it appears in Application Manager's list of applications. The **Application Type** column identifies the application as Standalone.

Edit a standalone application

You open a standalone application from the InTouch Application Manager and edit it in WindowMaker.

This section demonstrates the basic steps to build an application. The following figure shows a window from a simple application that combines and mixes the ingredients to make chocolate milk. Complete the procedures in

this section to learn the overall workflow to build an application.



In the window, the Alarm Viewer control and the lines that represent pipes are traditional InTouch graphic elements. All other graphics elements shown in the window are Industrial Graphics or Situational Awareness library symbols that can be used in Modern applications.

Edit an application

1. Open InTouch HMI Application Manager.
2. Double-click on the Chocolate Milk application to edit it. If you are in the Icons view, hover over the application tile, and select WindowMaker.

   The application opens in the WindowMaker.

   The first time you open an InTouch application in WindowMaker no windows have been created.
3. Right-click on **Windows** in the **Windows** panel and select **New Window**.
4. Select a template from the **Available templates** section. For this example, we recommend that you select the Native window template which allows multiple graphics to be added to the window. A frame window allows adding only one graphic to the window.
5. In the **Properties** panel, assign "Mixing Station" as the name of the window in the **Name** field.

6. Set the width and height of the window by entering values in the **Window Width** and **Window Height** fields.
7. Change the default background of the window to a lighter color by selecting **Window Color** and choosing a color from the **Color dialog**.
8. Select **OK**.

   The window you created appears in WindowMaker. Continue with the next procedure to add symbols to the window.

Add graphics to a window

Drag and drop graphics from the Industrial Graphic Toolbox to add symbols to a window.

This procedure explains how to add the following symbols to the window you created earlier:

- Situational Awareness Library
  - 3 valves (Valves folder, SA_Valve_2Way)
  - 1 vessel (Equipment folder, SA_Tank_Vessel)
  - 1 meter (Meters folder, SA_Meters)
  - 1 simple trend (TrendPen folder, SA_Trend)

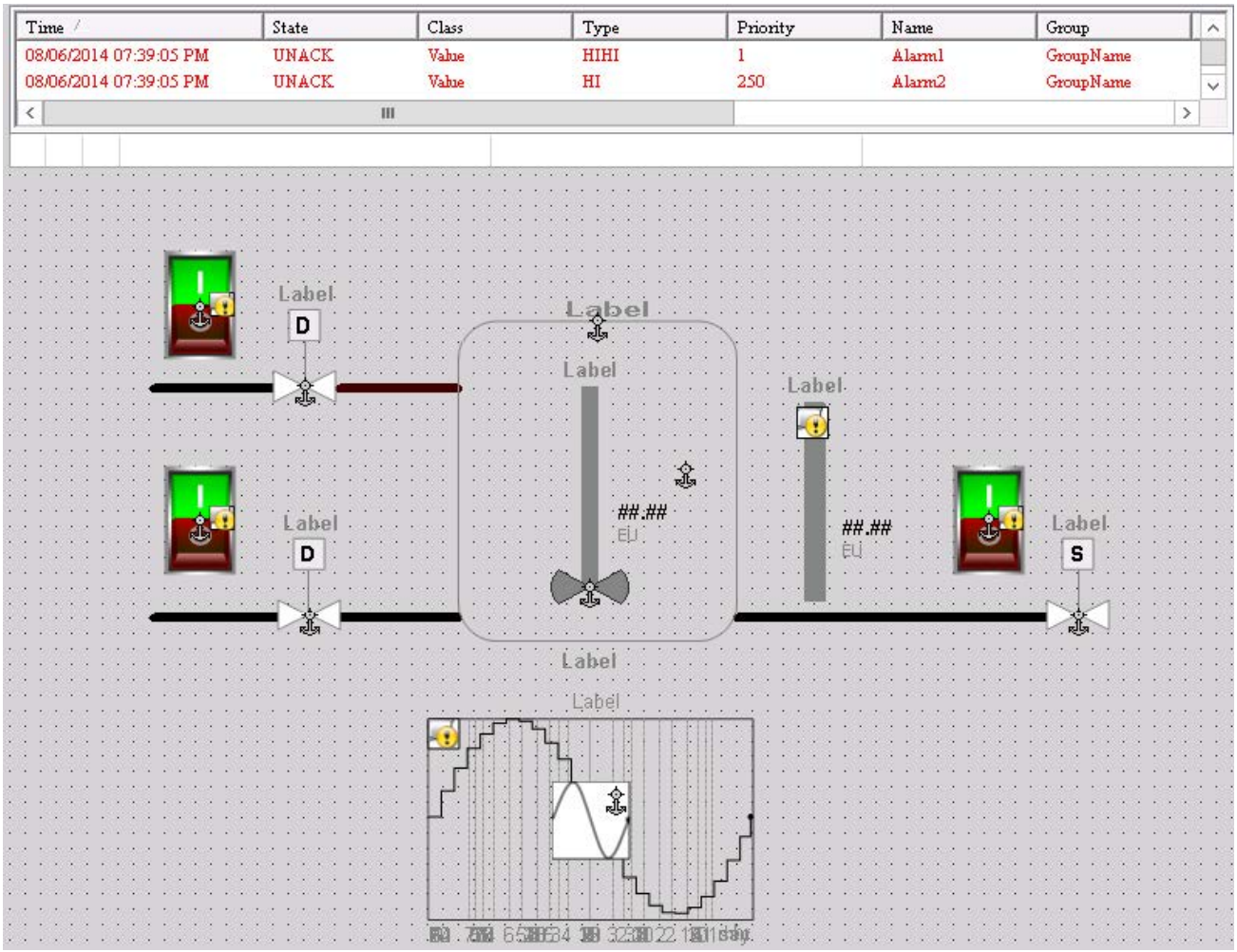
    - 1 agitator (Equipment folder, SA_Agitator_Settler)
- Industrial Graphic Library
    - 3 rocker switches (Switch folder, RockerSwitch)
- InTouch Wizards
    - 1 alarm viewer control (AlarmViewerCtrl)
- InTouch Graphic Toolbar
    - 3 lines that represent pipes

Add graphics to an application

1. If necessary, open the Mixing Station window you created in WindowMaker.
2. Expand the Situational Awareness Library folder in the **Industrial Graphic Toolbox** to show the list of folders.
3. Open the **Valves** folder and select the SA_ISA_2WValve symbol.
4. Keeping your left mouse key pressed, drag the symbol to the open window and release the key at the approximate location where you want to place the symbol.
5. Select the graphic symbol and place it precisely where you want it to appear in the window.
6. Repeat steps 3-5 and add the remaining Industrial Graphics and Situational Awareness Library symbols listed on the previous page.

   The list of graphics on the previous page includes the folders in the Industrial Graphic Toolbox where the symbols are located.
7. On the **Draw** menu, in the Insert group, select **Wizards**.

   The **Wizard Selection** dialog box appears.
8. Select **AlarmViewerCtrl** from the **ActiveX Controls** group.
9. Select **OK** and place the Alarm Viewer control near the top of the window.
10. On the **Draw** menu, in the Shapes group, select **Line**.
11. Draw three lines that represent the two input pipes and the output pipe.
12. Select **Line** from the menu bar and choose a thicker line type to make your lines look more like pipes.
13. Arrange the graphic elements on your window to look like the following example of a chocolate milk mixing station.



Create InTouch HMI tags

InTouch HMI applications represent an industrial process using data associated with InTouch HMI tags. It also supports User Defined Types (UDTs). You can perform different actions like add, delete, view, import, export and so on.

In this simple application, tag data will be shown or used to set the state of the symbols that represent the equipment of a mixing station.

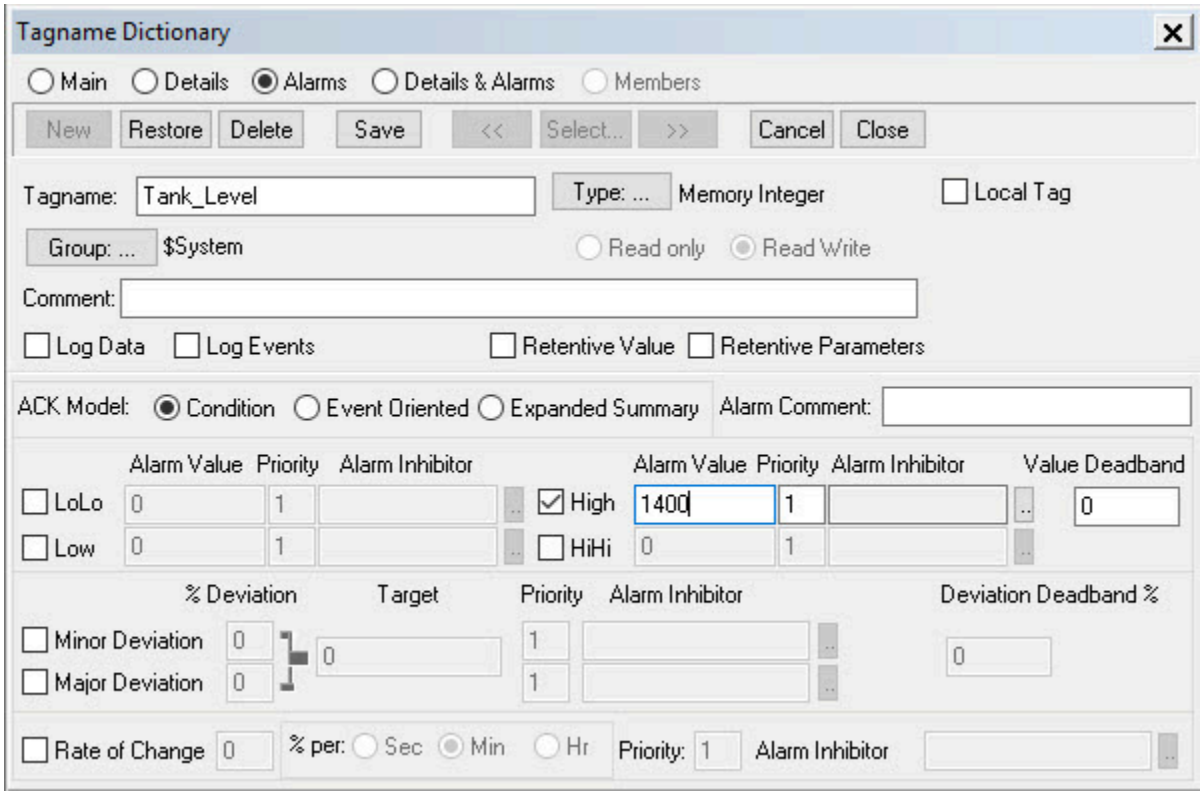
This procedure explains how to create the following tags for a mixing station Modern application:

Tag	Tag Type	Symbol Association
Tank_Level	Memory Integer	Mixing Tank
Valve_Chocolate	Memory Discrete	Chocolate Valve
Valve_Milk	Memory Discrete	Milk Valve
Valve_Outlet	Memory Discrete	Outlet Valve

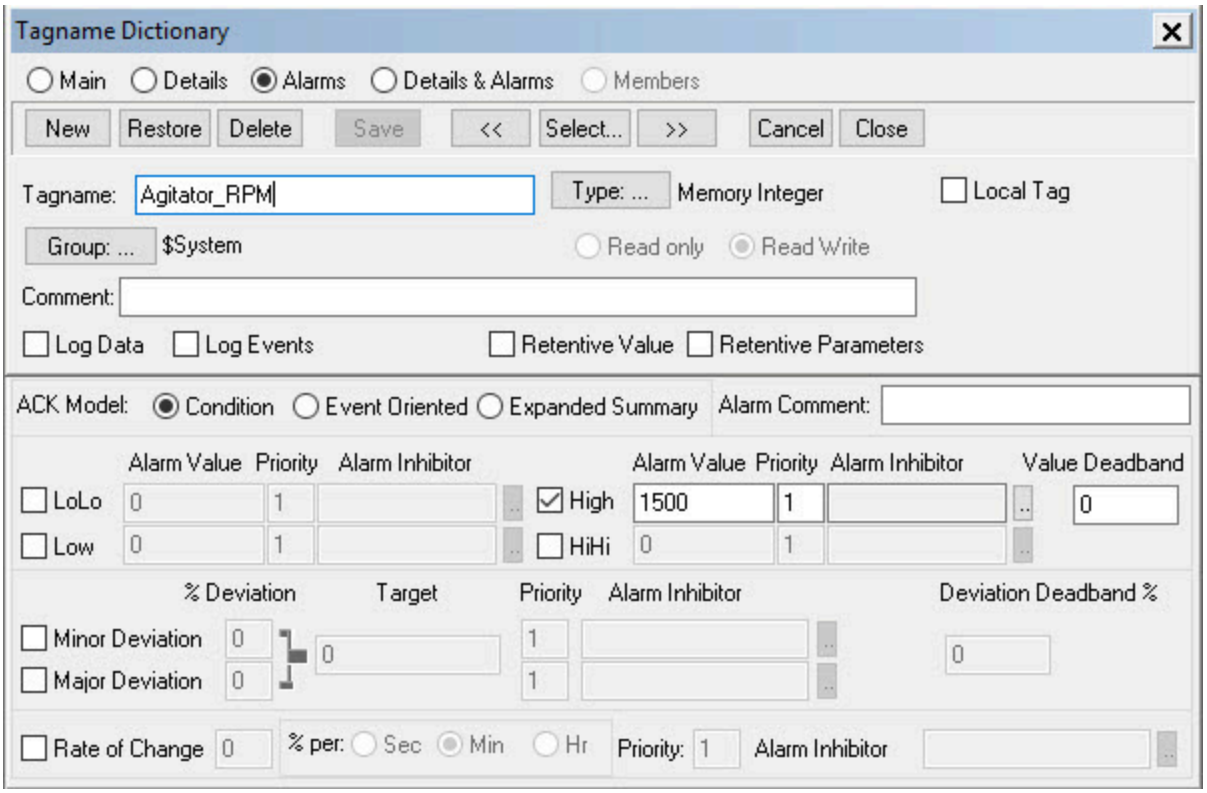
Agitator_RPM	Memory Integer	Tank Agitator

To create InTouch HMI tags

1. On the **Home** menu, in the **Tags** group, select **Tag Dictionary.**
2. The **Tagname Dictionary** dialog appears.
3. Select **New**. The **Tagname** field clears.
4. Type `Tank_Level` in the **Tagname** field.
5. Select **Type** to show the various types of InTouch HMI tags.
6. Select **Memory Integer** as the type of tag.
7. Select **Alarms** near the top of the Tagname Dictionary to expand the dialog box to show fields to set alarm conditions.



8. Select **High** and set 1400 in the **Alarm Value** field.
9. Select **Save**.
10. Repeat steps 2-4 to create the three valve tags.
    a. Enter the name of the valve tag in the **Tagname** field.
    b. Set the tag type to Memory Discrete for all three valve tags.
    c. Select **Save** to save each valve tag.
11. Create the Agitator_RPM tag using the same steps (2-8) used to create the Tank_Level tag.
    a. Enter Agitator_RPM as the name of the tag.
    b. Set the tag type to Memory Integer.
    c. Select **High** and set the **Alarm Value** field to 1500.



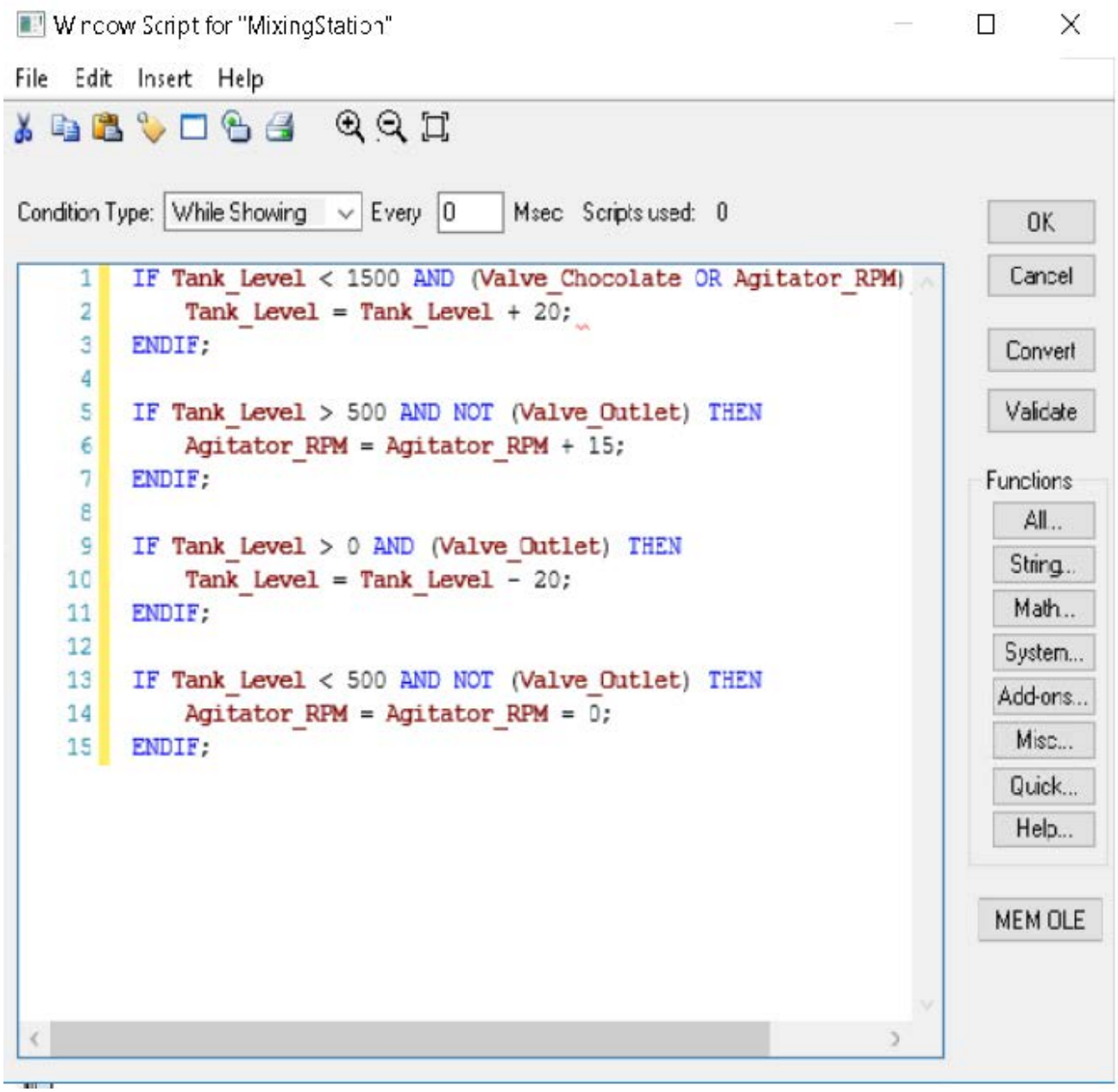
Create a window script

A window script sets the operating conditions of the Chocolate Milk application while it is running in WindowViewer:

- When the Chocolate or Milk valves are open and the Outlet valve is closed, the tank fills with ingredients.
- When the Chocolate or Milk valves are open and the Outlet valve is open, the tank volume remains constant.
- When the Chocolate and Milk valves are closed and the Outlet valve is open, the chocolate milk empties from the tank.
- When the mixing tank level is less than 1500 liters and the chocolate or milk values are open, the tank begins to fill with ingredients.
- When is the tank level is greater than 500 liters and the outlet valve is closed, the agitator begins to rotate.
- When the tank level falls to less than 500 liters and the outlet valve is open, the agitator stops.

Create a window script

1. Right-click on a blank area of the Mixing Station window to show a shortcut menu.
2. Select **Window Scripts** from the shortcut menu.
3. Type or copy the following windows script into the Scripts dialog box.



4. Set the **Condition Type** field to **While Showing**.
5. Set the **Every** field to a value between 500-700 milliseconds.
   The window script will run periodically at the interval you set in the **Every** field.
6. Select **Validate** to see if there are any errors in the script.
7. Correct any script errors and select **OK**.

Configure graphics

Industrial Graphics contain custom properties that extend the standard properties of a symbol. In this sample application, you must assign tags to custom properties to show the current value of a tag or set the states when the equipment represented by a symbol is active or inactive.

Most Situational Awareness Library symbols are also Symbol Wizards. In addition to custom properties, Symbol Wizards contain Wizard Options to configure their appearance and functionality.

This procedure explains how to assign values to the custom properties and Wizard Options listed in the following tables.

Graphic Custom Properties

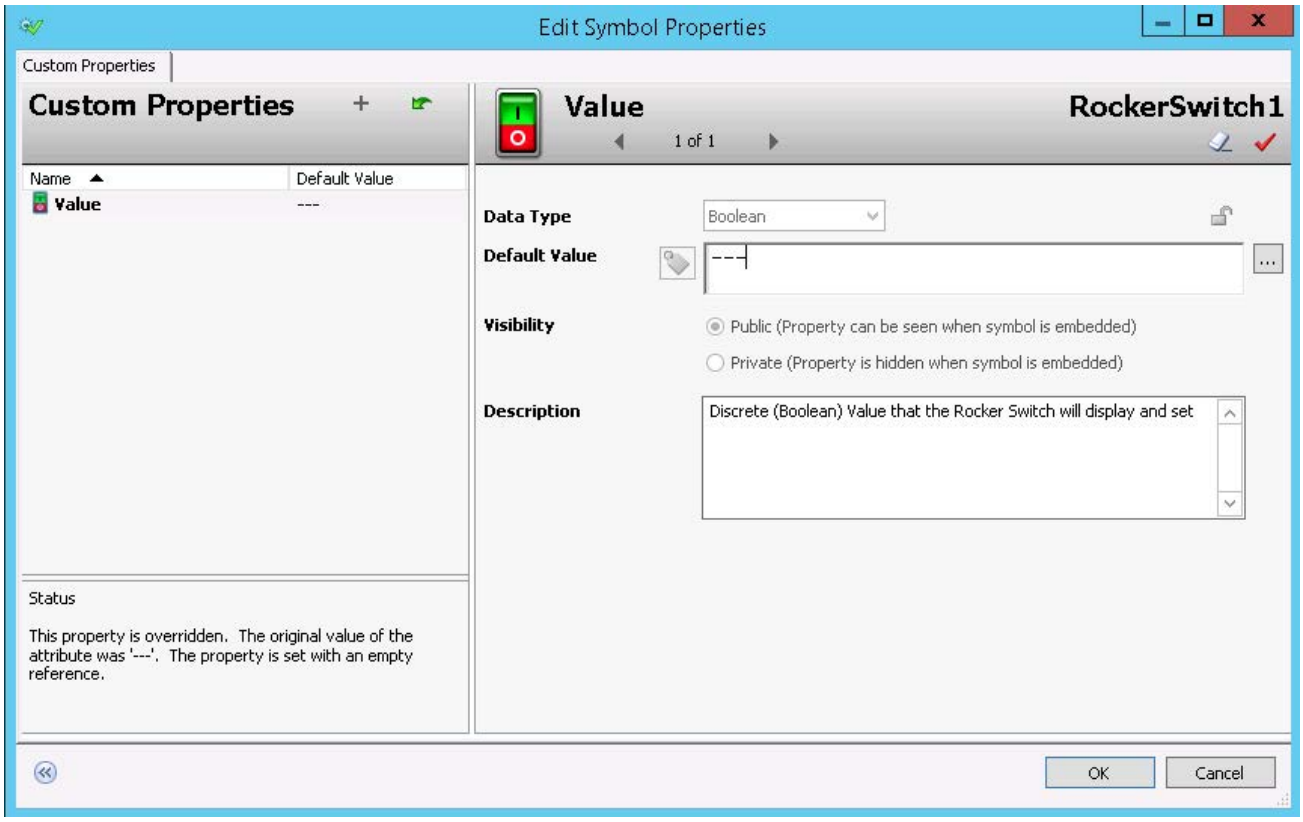
Graphic	Custom Property	Assigned Values
Chocolate Rocker Switch	Value	Valve_Chocolate
Milk Rocker Switch	Value	Valve_Milk
Outlet Rocker Switch	Value	Valve_Outlet
Chocolate Valve	EquipmentStateActive	Valve_Chocolate
Milk Valve	EquipmentStateActive	Valve_Milk
Outlet Valve	EquipmentStateActive	Valve_Outlet
Mixing Tank	LabelVisible	True
Tank Agitator	PV	Agitator_RPM
	PVRangeFullScaleMax	3000
	PVRangeFullScaleMin	0
Tank Volume Meter	PV	Tank_Level
	PVRangeFullScaleMax	1500
	PVRangeFullScaleMin	0
Tank Volume Trend	Pen	Tank_Level
	Pen_RangeFullScaleMax	1500
	Pen_RangeFullScaleMin	0
Alarm Client	None	N/A

Symbol Wizards

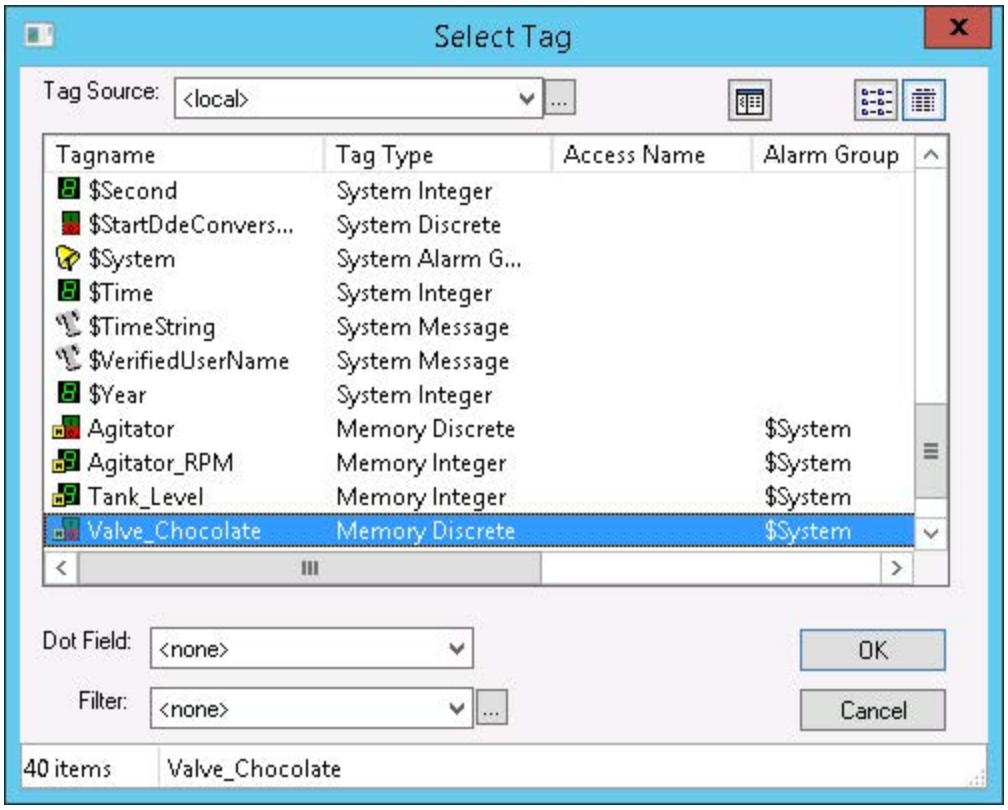
Symbol Wizards	Wizard Options	Assigned Values
Chocolate Rocker Switch	None-Industrial Graphic	N/A
Milk Rocker Switch	None-Industrial Graphic	N/A
Outlet Rocker Switch	None-Industrial Graphic	N/A
Chocolate Valve	ActuatorType	Digital
Milk Valve	ActuatorType	Digital
Outlet Valve	ActuatorType	Digital
Mixing Tank	QualityStatusIndicator	False
Tank Agitator	PVNumericDisplay	True
	EngUnits	True
	EngUnitsType	StaticText
	QualityStatusIndicator	False
Tank Volume Meter	Type	Level
Tank Volume Trend	SymbolMode	Advanced
	YAxisRangeType	ClipOutOf RangeValues
	PlotType	Line
	Label	True
	LabelType	StaticText
	GridVerticalTimePeriod Scale	True
	GridVerticalTimePeriodScaleUnits	True
	TimePeriod	True
Alarm Client	None	N/A

To configure custom properties and Wizard Options

1. Double-click on the Chocolate rocker switch to show the **Edit Symbol Properties** dialog box.



2. Select **Value** from the **Name** field.
3. Select the **Browse** button at the right of the **Default Value** field to show the **Select Tag** dialog box.



4. Select the **Valve_Chocolate** tag and select **OK**.

   The current value of the Valve_Chocolate tag is associated to the Chocolate rocker switch's Value custom property.
5. Repeat steps 1-4 for the other two rocker switch symbols and assign the tag shown in the Symbol Custom Properties table to each symbol's Value custom property.
6. Double-click on the Chocolate valve to show the **Edit Symbol Properties** dialog box.
7. Using steps 1-4, assign the Valve_Chocolate tag to the symbol's EquipStateActive custom property.
8. Select the **Wizard Options** tab.
9. Select **ActuatorType** from the **Name** field.
10. Set the **Value** field to Digital and select **OK**.
11. Repeat steps 1-10 and assign values to the custom properties and Wizard Options of the remaining symbols of the Chocolate Milk application.

Change symbol labels

The Symbol Wizards used in the Chocolate Milk application have their Label Wizard Option set to True and the LabelType Wizard Option set to StaticText. With this symbol configuration, you can use the InTouch Substitute Strings function to assign a visible static label for the symbols in the Mixing Station window.

Symbol	Current String	Application Label
Chocolate Valve	Label	Chocolate

Milk Valve	Label	Milk
Outlet Valve	Label	Outlet
Mixing Tank	Label	Chocolate Milk
Agitator	Label	Agitator
	EU	RPM
Tank Meter	Label	Tank Volume
	EU	Liters
Trend	Label	Tank Volume

Change symbol labels

1. Right-click on the Chocolate valve symbol to show the shortcut menu.
2. Select **Substitute** and **Substitute Strings** from the shortcut menu.

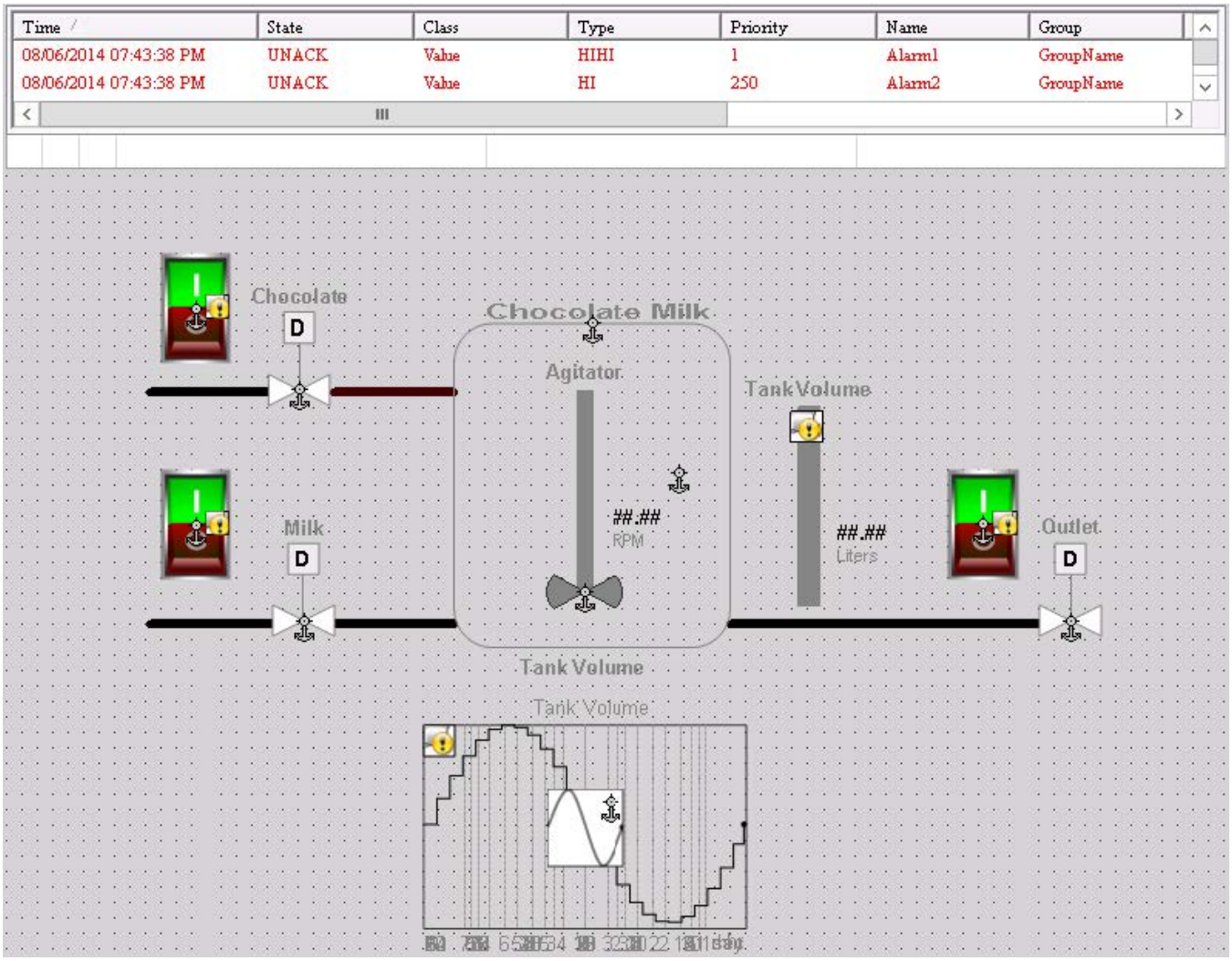
   The Substitute Strings dialog box appears with fields to substitute the current strings of the symbol.

3. Type Chocolate in the **Label** field and select **OK**.

   The Chocolate label appears above the valve symbol.
4. Repeat steps 1-3, assign labels to the remaining symbols you added to the Mixing Station window.

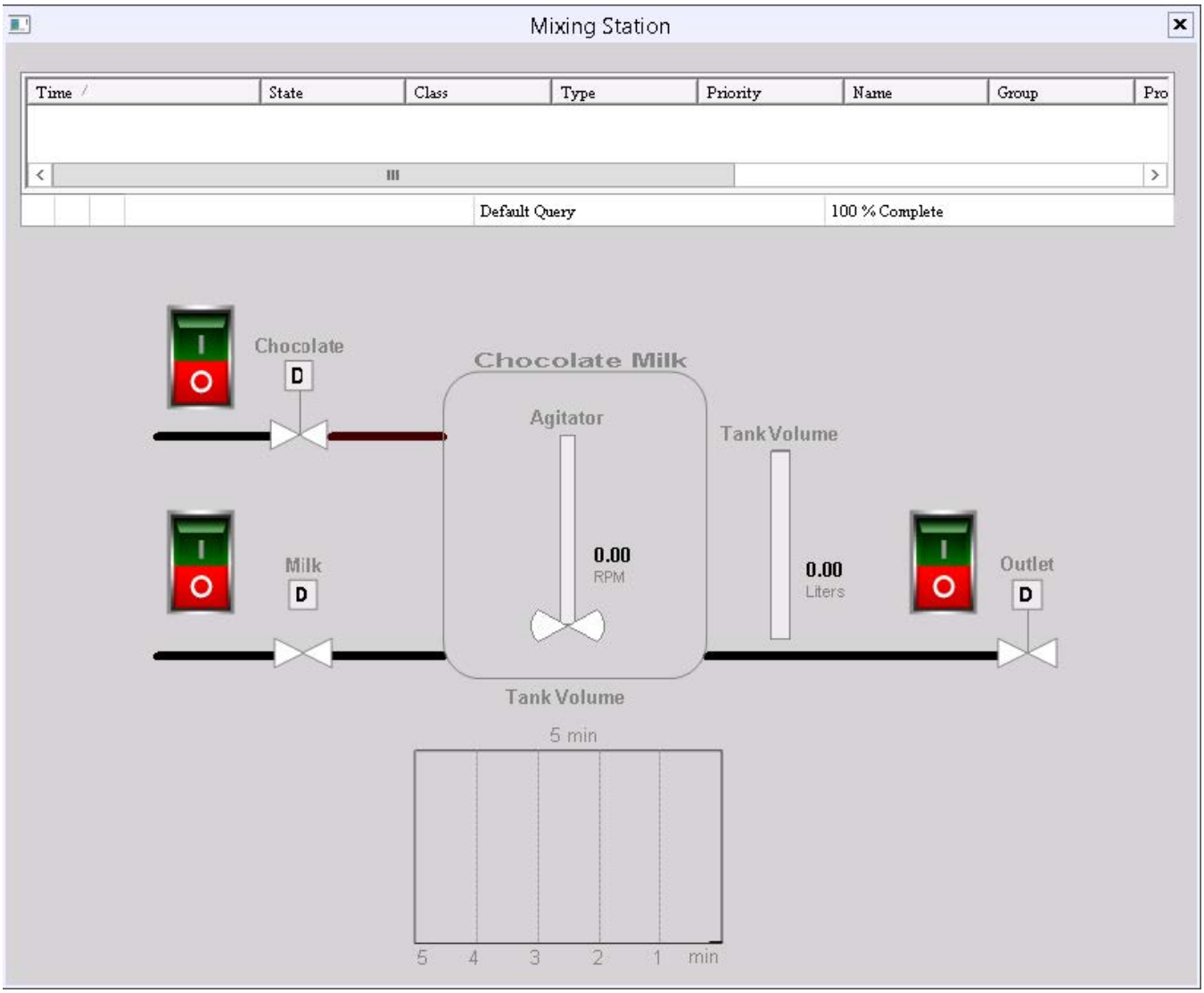
Run a standalone application

You view a running application from WindowViewer. After you have finished, your Mixing Station window should look like the following example.

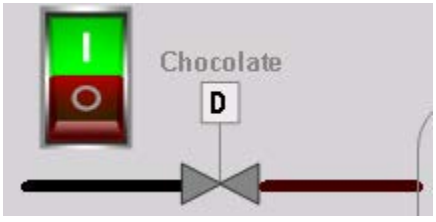


In this example, a window script begins running when viewing the application. The script assigns states and values to the assigned InTouch HMI tags associated with the symbols shown in the application's window.

The following example shows the Chocolate Milk application immediately after starting it in WindowViewer. All of the valves are closed and the mixing tank is empty.



Typically, Situational Awareness Library symbols use fill shading to indicate their current state. Open the Chocolate or Milk valve by selecting a rocker switch. Notice the change in the fill color that indicates the valve is open. Also, the fill color of the agitator and meter symbols change to indicate they are in an active state or showing a value.



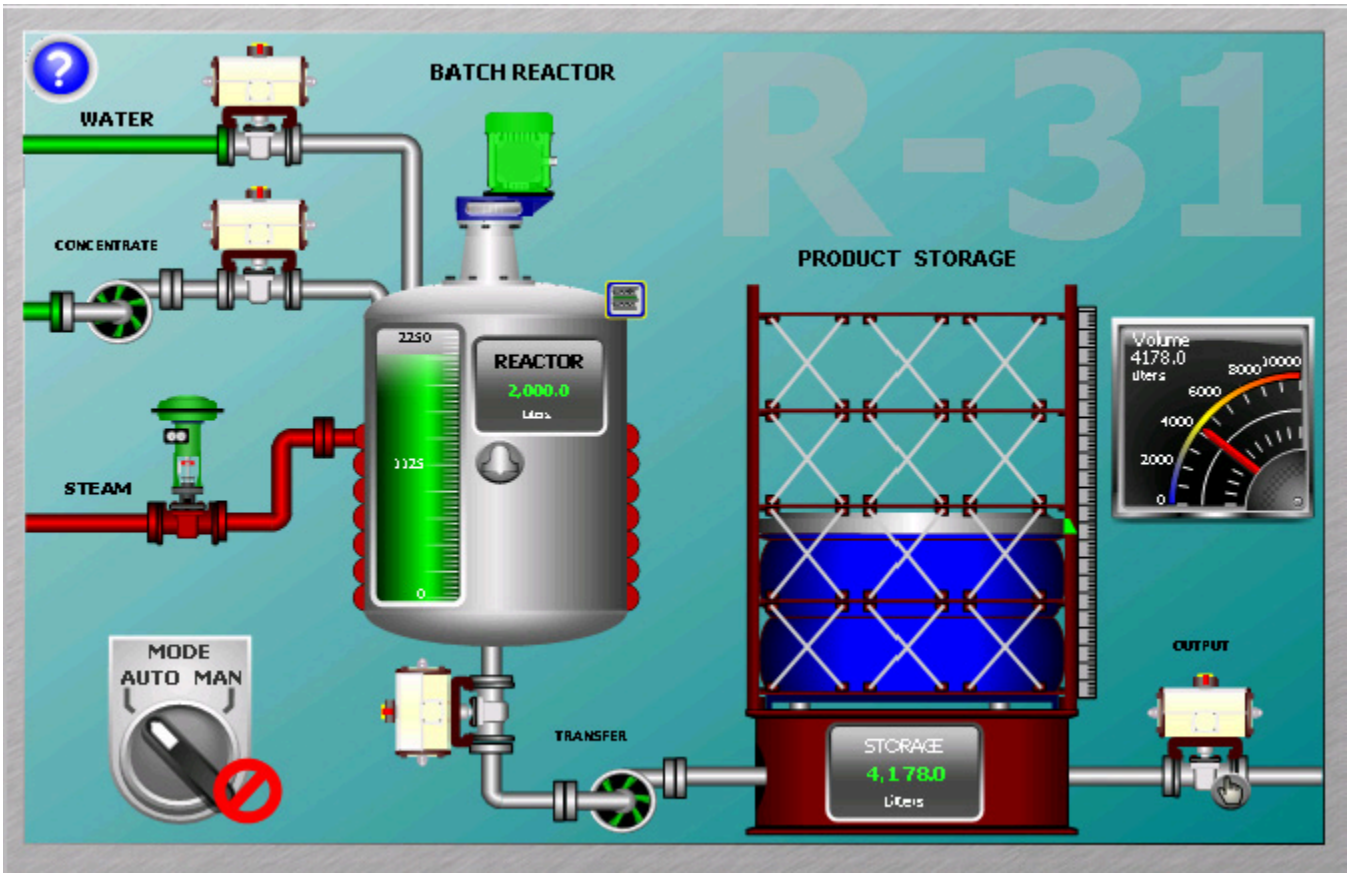
When the tank volume reaches 500 liters, the agitator starts, and its current RPM appears next to the agitator. Alarms occur if the tank volume exceeds 1400 liters or the agitator exceeds 1500 RPM. You manage alarms by selecting options from the shortcut menu of the Alarm Viewer control.

Create managed applications

Managed applications are built using the IDE and automation objects, in addition to the components of a Standalone application. Each Managed application is associated with a InTouchView App object.

When you install InTouch HMI, you can install several sample applications. You can examine these sample applications to understand how scripts, animations, and graphics work together to provide a visual interface for your production environment.

The following figure shows a portion of a window from the InTouch reactor demonstration application.



The reactor application demonstrates how you manage an application with the IDE and includes Industrial Graphics and objects.

The analog meter next to the product storage tank shows the current volume of liquid stored in the tank. The meter is not part of the standard reactor application.

This section describes the essential tasks to create a managed InTouch application by showing how to embed this meter into the Reactor application.

Start the IDE

You can start the IDE from the Windows **Start** menu.

Starting the IDE from the Start Menu

The following procedure shows the steps to start the IDE from a computer running Microsoft Windows 7 and

later versions of Windows.

Start the IDE from the Windows Start Menu

- From the **Start** menu, go to the AVEVA System Platform folder, and then select **System Platform IDE**.

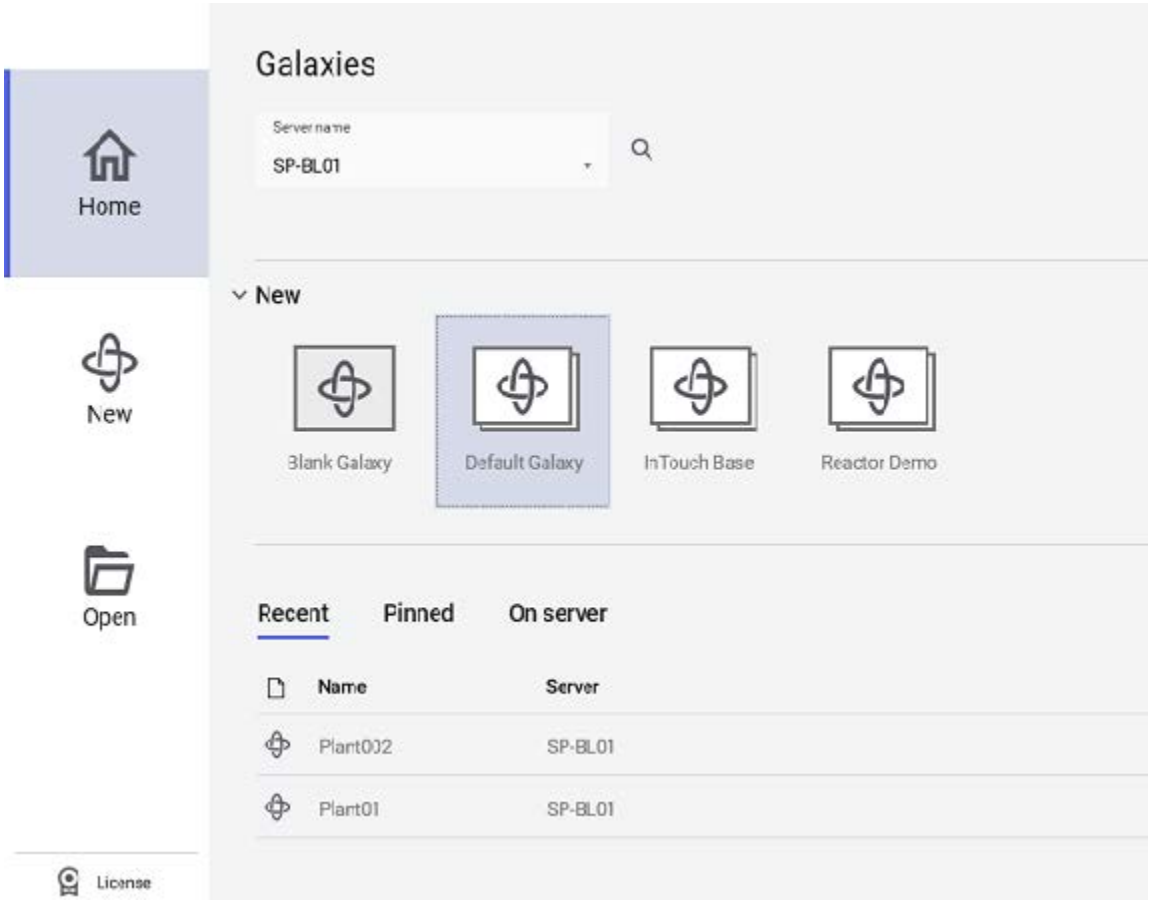
  The **Galaxies** screen appears.

Create a Galaxy

After you start the IDE, the Galaxies screen appears. The first time you start the IDE you need to create a Galaxy, which is an project database. After that, you can select the Galaxy in which you are developing managed applications each time you start the IDE.

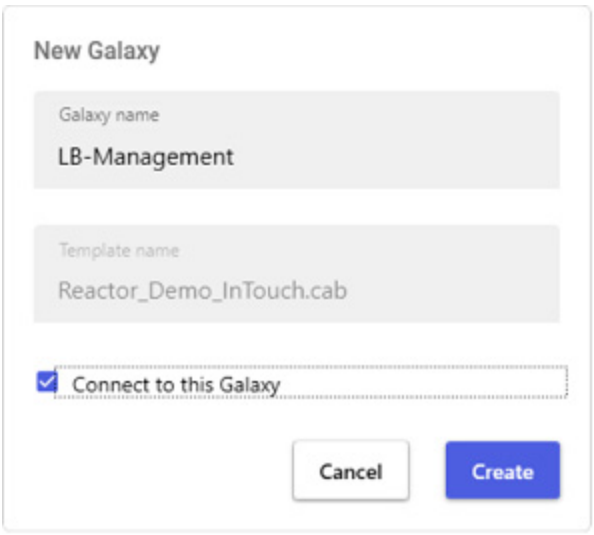
Create a Galaxy

1. Start the System Platform IDE. The **Galaxies** screen appears.



2. Under **Galaxy**, select a template.

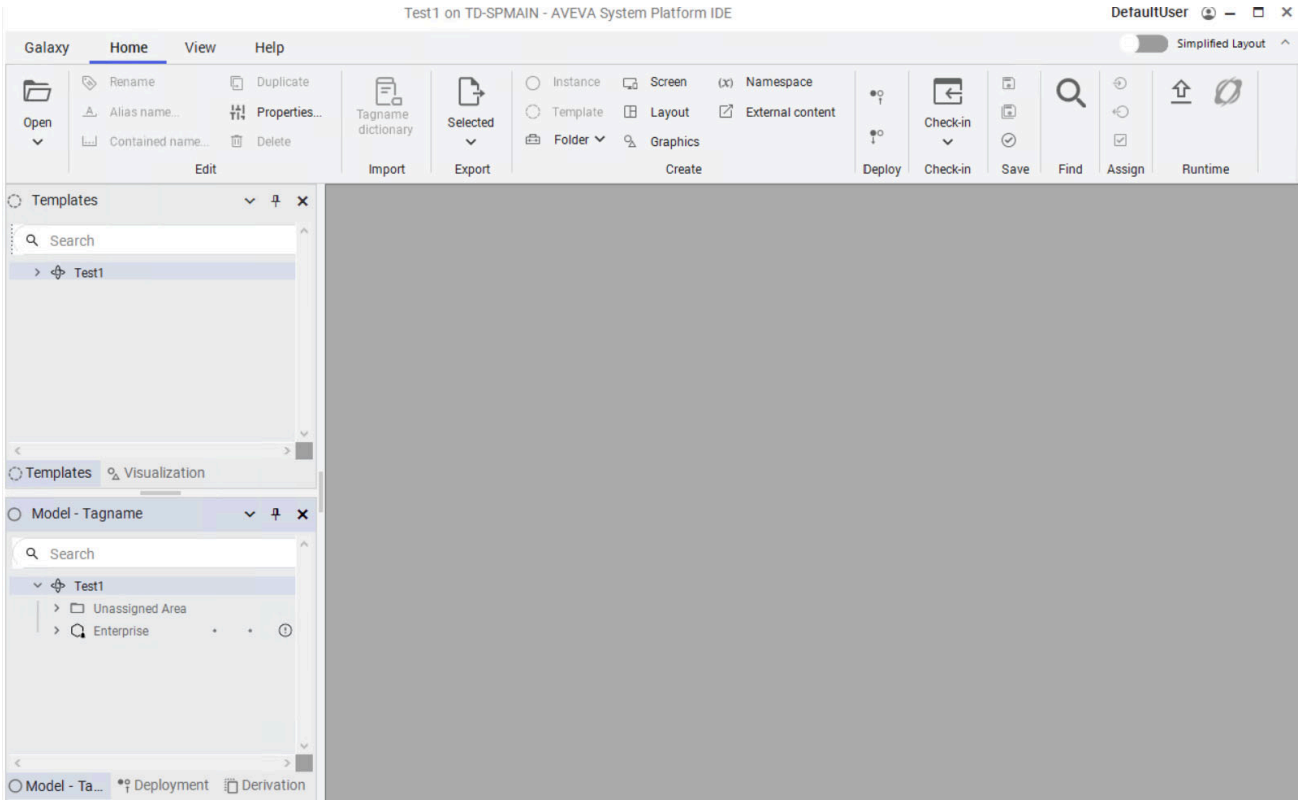
   The **New Galaxy** dialog box appears.



3. Complete the fields of the **New Galaxy** dialog box by doing the following:
   a. In the **Galaxy name** box, type the name of the Galaxy that you are creating.
   b. The T**emplate name** box displays **Reactor_Demo_InTouch. cab** as the type of Galaxy.
   c. Select the checkbox **Connect to this Galaxy**.

   This is a custom Galaxy that includes the InTouch Reactor Demo application.
4. Select **Create**. The **New Galaxy** dialog box shows the progress of creating a new Galaxy.
5. Select **Close** after the new Galaxy is created.
6. Select **Connect** to connect to the Galaxy you created.

   The **AVEVA System Platform IDE** dialog box appears.



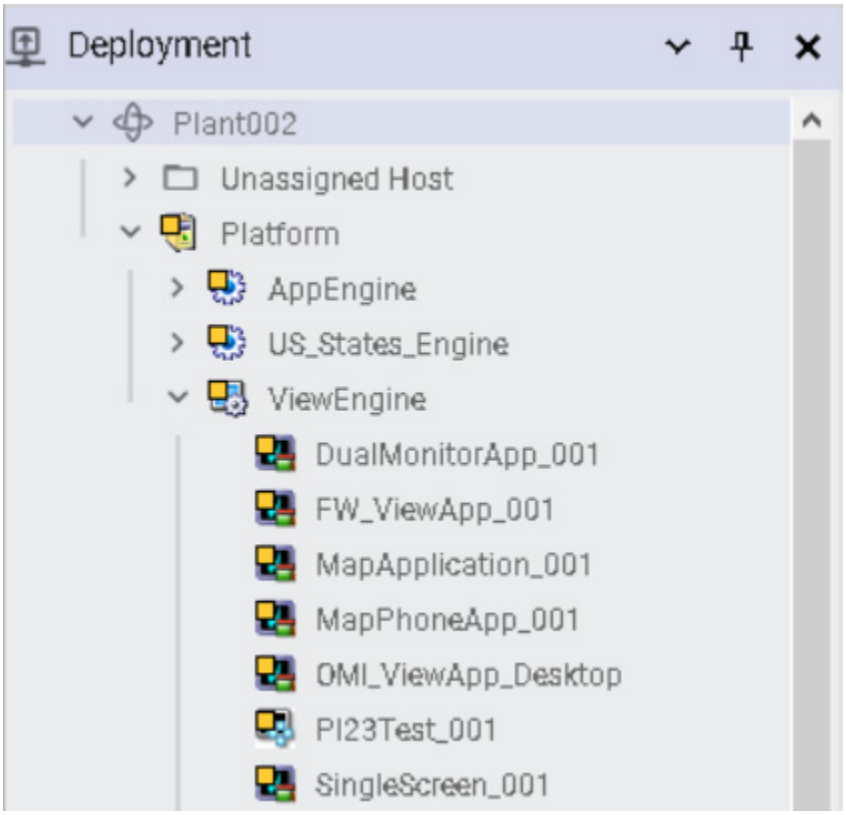
7. Select the **Template** tab.
8. If necessary, select the **InTouch Reactor** folder to show the objects within it.

The **Template Toolbox** shows the $ReactorDemo derived template. You can use this derived template to create a managed application or edit the application within WindowMaker.

Deploy your application objects

Deploying your Galaxy copies the objects from your development environment to a run-time environment. Until you deploy your application's objects from the IDE, you cannot run your managed applications.

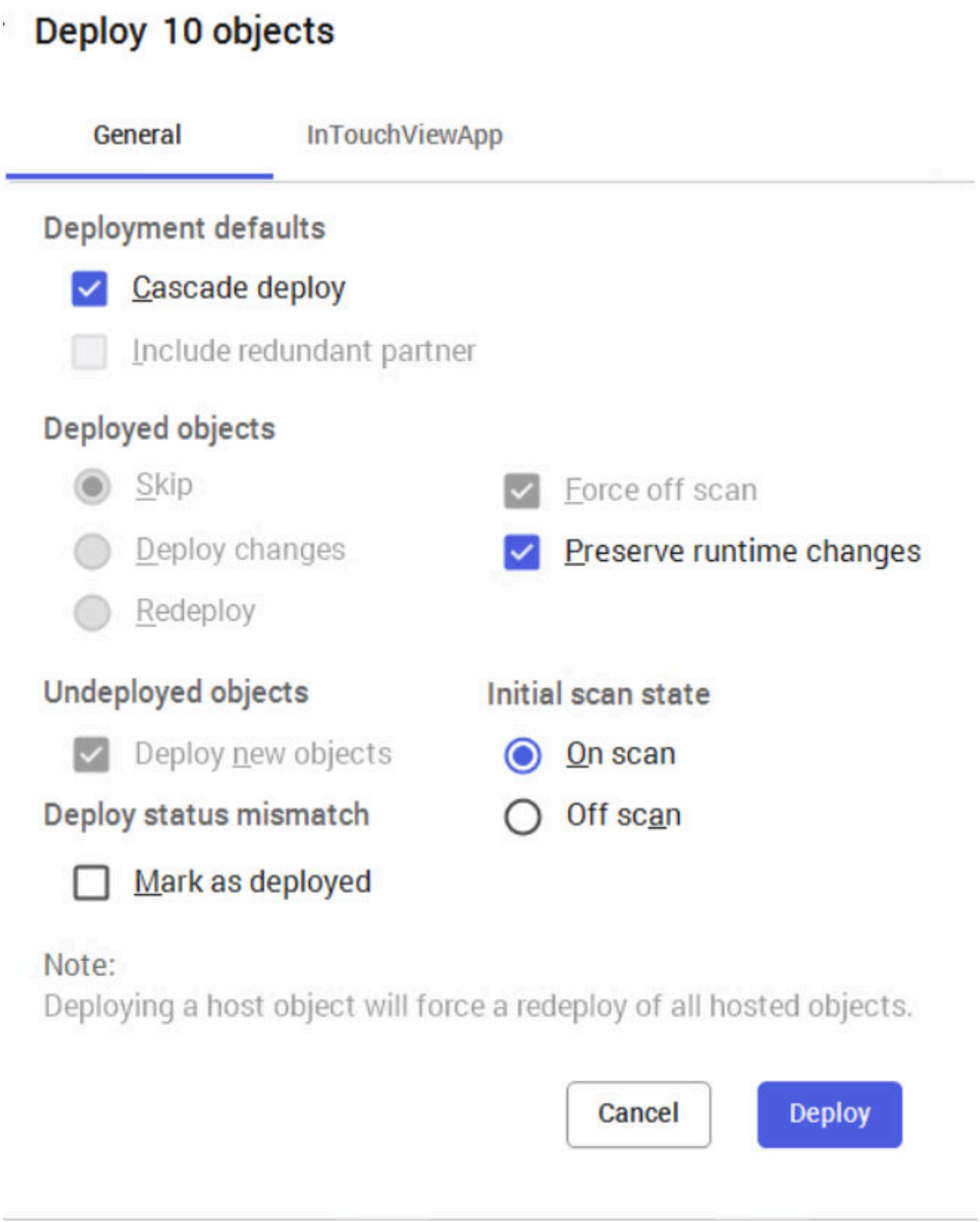
The pane at the bottom left of IDE includes three tabs to show different views of your Galaxy objects. Selecting the **Deployment** tab shows the current status of all objects that are part of your application. An orange square next to an object indicates the object is undeployed.



Deploy your application's objects

1. In the **Deployment** pane, right-click on the name of the Galaxy at the top of the list of application objects.
2. Select **Deploy** from the shortcut menu.

   The **Deploy** dialog box appears with options.



3. Accept the default options and select **Deploy**.
4. Select **Close** after all of your objects have been deployed.

The orange square next to each object is gone, indicating that your application's objects are deployed.

Edit a managed application

You edit a managed InTouch application by starting WindowMaker from the IDE.

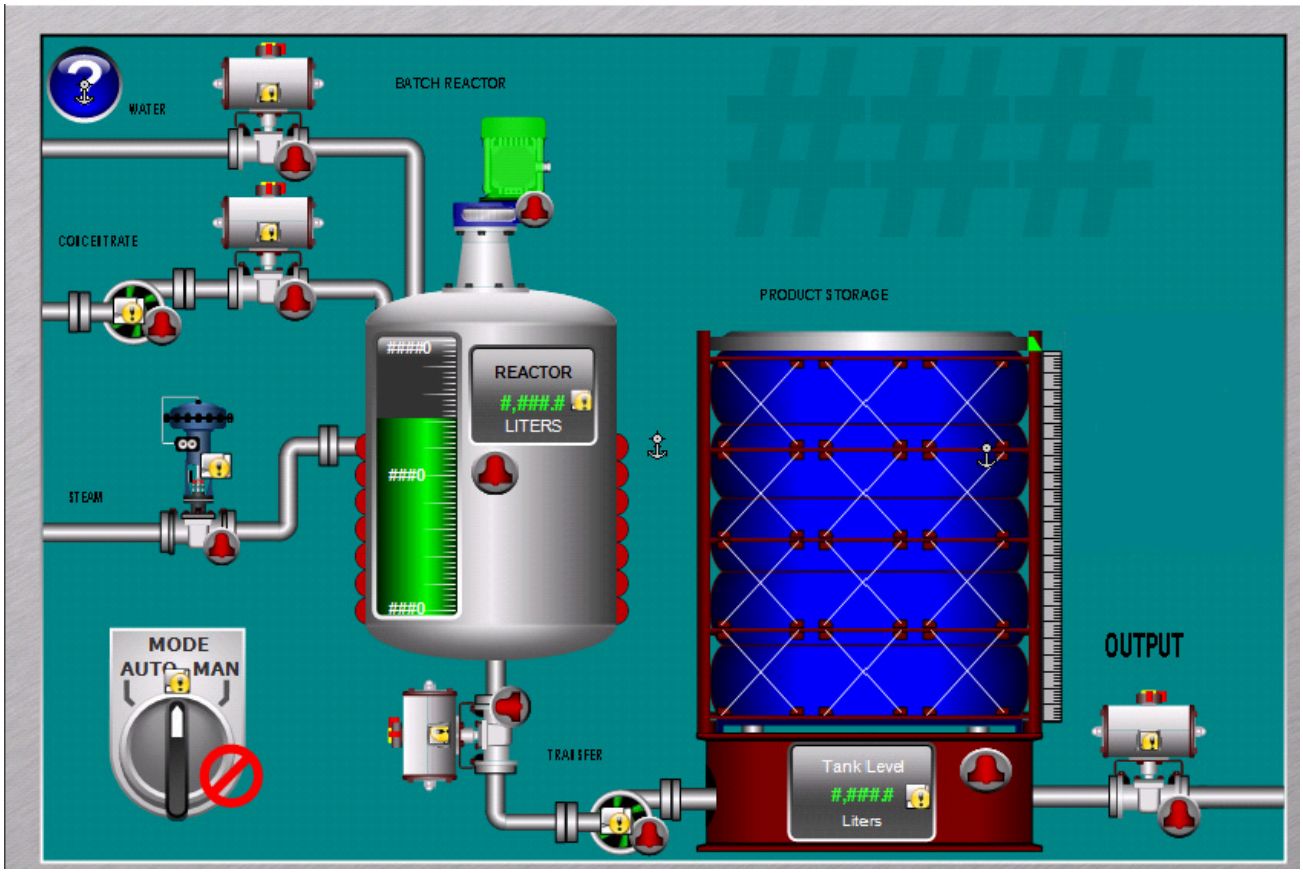
Start WindowMaker from the IDE

1. Open the IDE.
2. Select the **Templates** tab to show the set of folders containing automation objects.
3. If necessary, select on the **InTouch Reactor** folder to expand the list of automation objects within it. You see the $ReactorDemo template.
4. Double-click on the $ReactorDemo template.

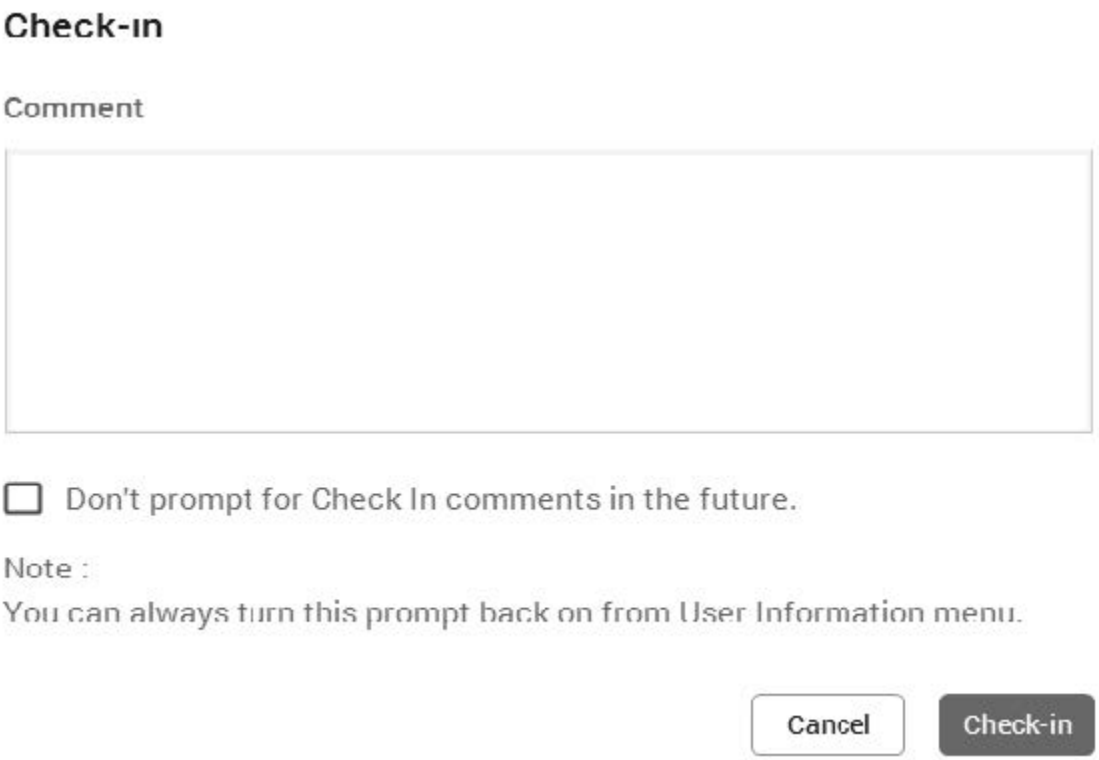
   The Windows panel lists the windows that are part of the Reactor demonstration application.
5. Select **File** menu, then select **Open**.

   The **Windows to Open** dialog box appears.
6. Select **File**, and then **Open Window** to show the **Windows to Open** dialog box.
7. Select the **Main**, **Menu**, and **Reactor Display** windows from the list and select **OK**. Together, these windows make up the main Reactor demonstration screen.

   The following figure shows the portion of the Reactor Demo window containing graphics that represent the components of a reactor.



8. Increase the text size of the OUTPUT caption to the right of the tank by doing the following:
   a. Select the OUTPUT caption, which shows the text box sizing handles.
   b. Using your mouse, click on a sizing handle and keep the mouse button pressed.
   c. Move the mouse to increase the size of the text box.
   d. Release the mouse button when the text is the size you want.
9. Save your changes to the window.
10. Select **File**, and then **Exit**. WindowMaker closes and you return to the IDE. The **Check In** dialog box appears.



11. Type a comment if you want, and then select **Check-in**.
12. Select **Close** after the application is checked in.

You have made a round trip from the IDE to WindowMaker and back to the IDE again. Now that you understand the steps to edit a managed application from WindowMaker, the next section explains how to embed an Industrial Graphic in an application window.

Embed Industrial Graphics into an InTouch managed application window

You can embed an Industrial Graphic into the windows of your managed InTouch application. You cannot embed Industrial Graphics into windows of stand-alone applications.

The embedded symbol appears with its original name appended by a number. The number increments each time you embed the same symbol again.

Embed an Industrial Graphic from the Graphic Toolbox

1. Open the IDE.
2. Double-click on the $ReactorDemo derived template to open it in WindowMaker.
3. Show the main Reactor Display window.
4. Right-click a graphic and select **Embed Industrial Graphic**.

   The Galaxy Browser appears.

5. Select the Graphic Toolbox icon. The symbols that belong to the **Graphic Toolbox** are listed in the left pane.

6. Expand the list of the **Industrial Graphic Library** folder.
7. Expand the **Analog Meters** folder. The meter symbols within the folder appear in the right pane of the Galaxy Browser.
8. Select the AnalogMeter90Degree symbol and select **OK**. WindowMaker reappears.
9. Click to the right of the tank in the Reactor window to embed the meter symbol. The meter symbol appears at the window location you selected.
10. Select the meter. Sizing handles appear around the border of the symbol.
11. Using your mouse, move the sizing handles to change the size of the meter.
12. Reduce the size of the meter and position it near the top of the tank.

    Tip: Press the SHIFT key to maintain the vertical and horizontal perspective when you resize the meter.
13. Select the **Runtime** icon to run the application in WindowViewer.
14. Verify the size of the meter and that it is aligned with the top of the tank.
15. Select the **Development** icon to return to WindowMaker.

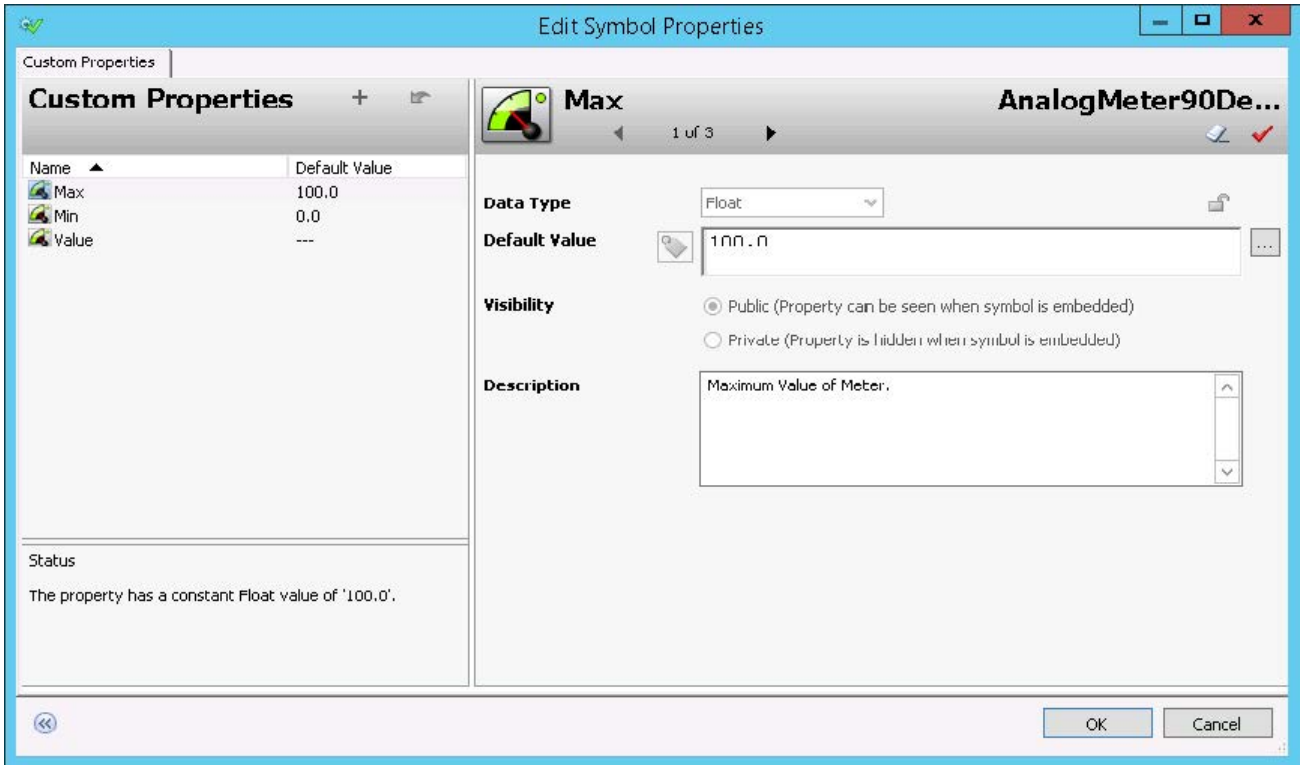
Connect attributes to an Industrial Graphic

You can connect Industrial Graphic attributes to InTouch HMI tags by overriding the custom properties of an embedded Industrial Graphic.

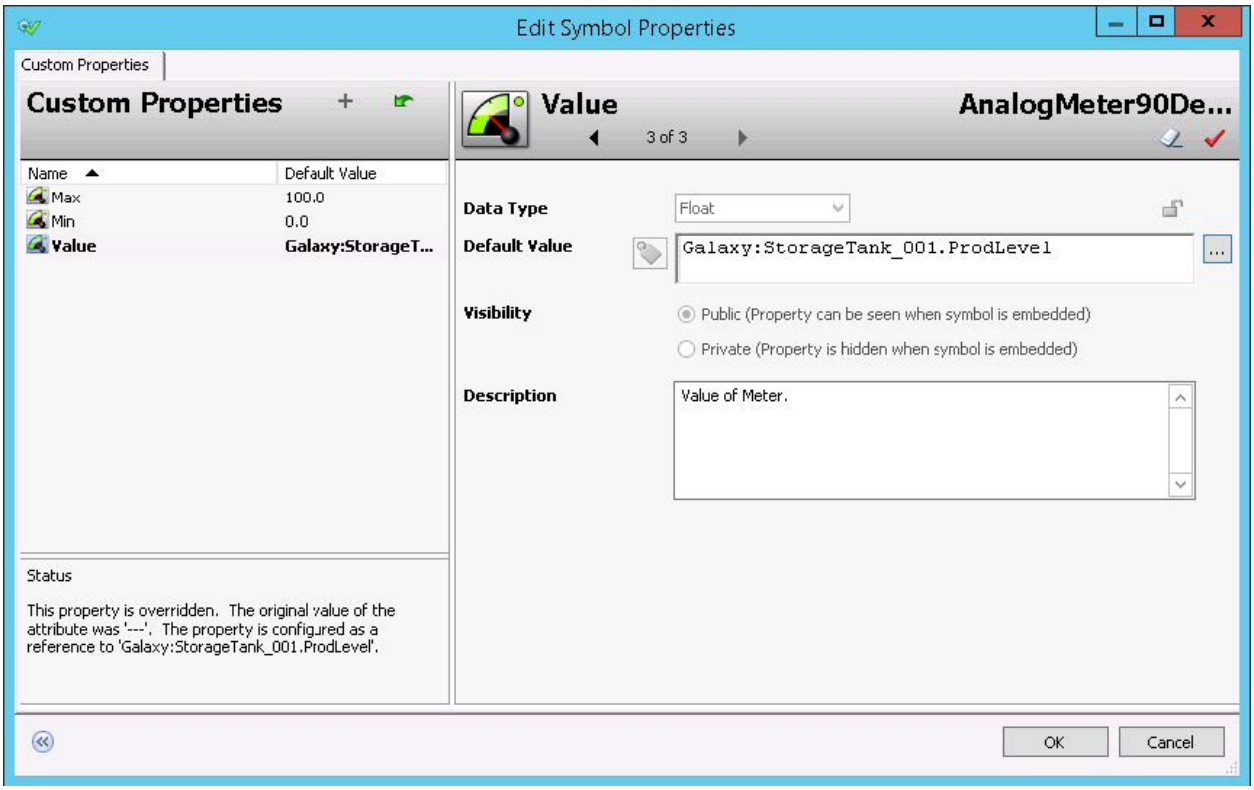
Custom properties expose the attributes of an Industrial Graphic to InTouch. The custom properties may or may not be used internally by the animations of the Industrial Graphic.

Connect an Industrial Graphic to an attribute

1. Double-click on the embedded meter. The **Edit Symbol Properties** dialog box appears.



2. Select the **Value** property. Do the following to associate an attribute to the meter:
   a. Remove the three hyphens from the **Default Value** field.
   b. Select the browse button of the **Default Value** box. The **Select Tag** dialog box appears.
   c. In the **Tag Source** field, select **Default Galaxy**. The **Galaxy Browser** dialog box appears with StorageTank_001 listed in the **Instances** column.
   d. Select StorageTank_001 to show a list of attributes.
   e. Select the ProdLevel attribute from the list and select **OK**. The **Edit Symbol Properties** dialog box shows the StorageTank_001 ProdLevel attribute assigned to the **Value** property of the meter symbol.



f. Select the **Max** property from the **Edit Custom Properties** dialog box.

g. Type 10000 in the **Default Value** box.

h. Select **OK** to close the **Edit Custom Properties** dialog box.

Any animation in the Industrial Graphic configured with the selected custom property now uses the object attribute value during processing.

3. Select the meter symbol you embedded.
4. Select **Special**, and then **Substitute Strings** to show the **Substitute Strings** dialog box. Do the following to change the labels that appear on the meter:

a. Type the word Volume in the **Label** box.

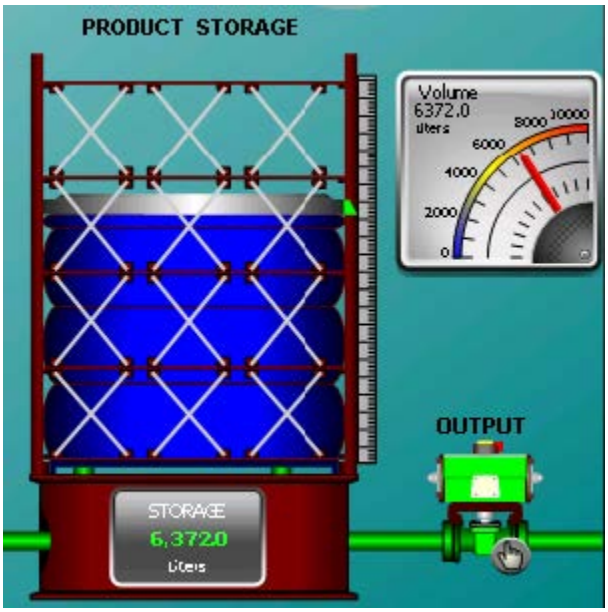
b. Type Liters in the **Units** box, and then select **OK**.

The face of the meter shows the labels you changed.

5. Save your work in WindowMaker.

6. Test your managed InTouch HMI application by switching to WindowViewer and running the Reactor application.

The analog meter symbol you embedded in the Reactor window shows the same tank volume value as the digital read-out at the bottom of the tank.



During testing, changes can be made to a symbol even without checking in the symbol. These changes are propagated to all places the symbol is used and that switching to WindowMaker and then back to WindowViewer accepts these changes. This is the quickest way to develop a managed InTouch application with Industrial Graphics. After you place your application into production, you can designate specific computers just to run your applications in WindowViewer.

These procedures explained some of the essential steps to create a managed application with the IDE. The next section includes a set of tables that list other important tasks and the books within the InTouch library that describes how to complete these tasks.

View InTouch applications remotely

InTouch HMI offers two methods to view InTouch applications or graphics remotely.

- InTouch Access Anywhere
- InTouch Web Client

InTouch Access Anywhere

InTouch Access Anywhere provides you with remote access to InTouch applications from almost all devices that support an HTML5 compatible web browser. Using InTouch Access Anywhere, you can show WindowViewer running your applications within a web browser on mobile phones, tablets, laptops or desktop computers.

InTouch Access Anywhere is available as an option during installation. For more information on how to install, configure and use InTouch Access Anywhere, refer to the *InTouch Access Anywhere Server Administrator Guide*.

InTouch Web Client

The InTouch Web Client feature allows you to view selected Industrial Graphics used within an InTouch HMI application on any HTML5 compatible web browser. A built-in Web Server provides web browsers access to application graphics, from any Microsoft Windows client or server operating system without the use of Remote Desktop Protocol (RDP) or Internet Information Services (IIS) for Microsoft Windows Server.

You can view application graphics in a web browser for both modern and managed applications. Using the InTouch Web Client you can:

- Toggle between WindowMaker and the Web Client easily with the Web Client Fast Switch
- View graphics on multiple devices and multiple screen sizes
- Pan and zoom application graphics
- Host InTouch application graphics on external websites

The InTouch Web Client is licensed separately but installed as part of the InTouch HMI installation. For more information on how to configure and use the InTouch Web Client, refer to the *AVEVA InTouch HMI help*.

Design Standards

Integrated development environment

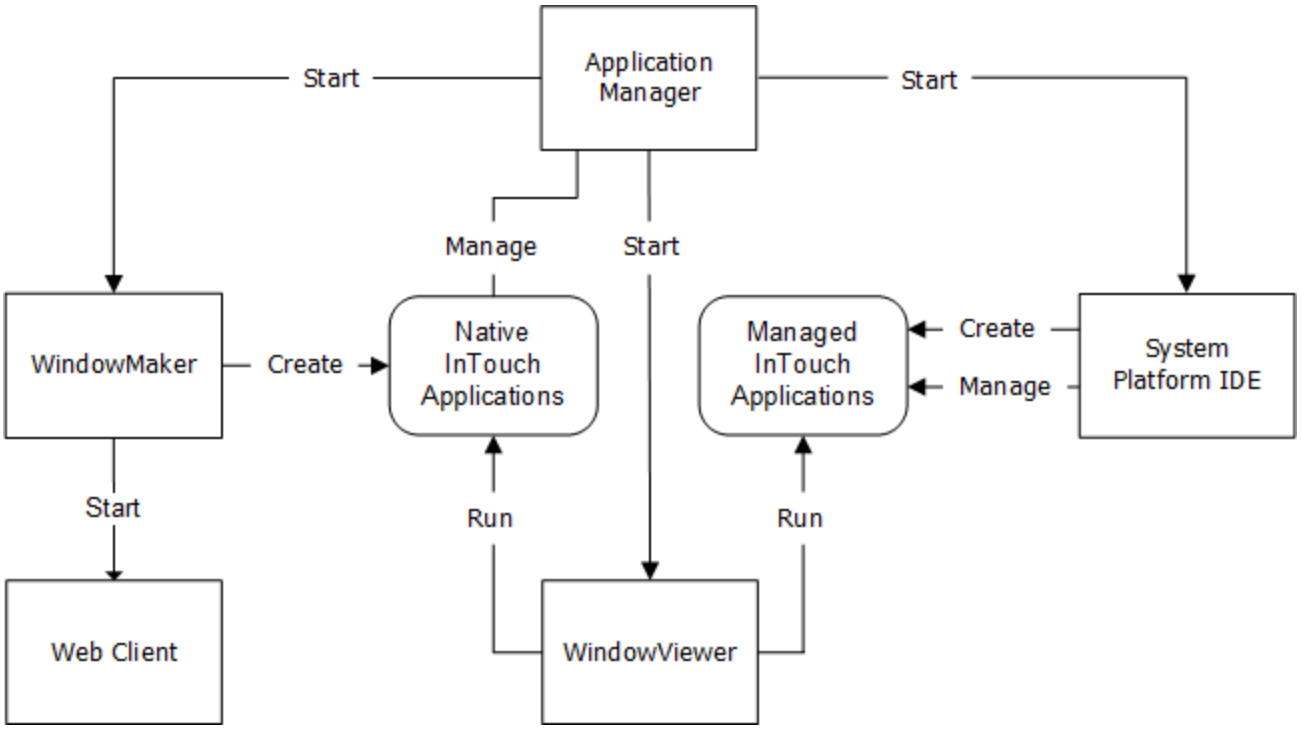
WindowMaker is the development environment you use to create InTouch HMI applications. You use WindowMaker to create the visual interface used by operators to view and manage your manufacturing processes. An InTouch HMI interface shows data from and writes data back to the production environment.

You can configure the following visual interface elements of your InTouch applications with WindowMaker:

- Windows contain interactive visual elements for plant operators to manage a production process.
- Basic objects are simple graphical elements, such as rectangles, circles, lines, and text.
- User-defined complex objects consist of one or more basic objects that represent elements in your production environment, such as valves and tanks.
- Pre-defined complex objects perform specific functions, such as alarming, historical trending, or Symbol Wizards.
- Animation links are properties of simple and complex objects to animate their appearance and transmit user input to tasks and production data changes.
- Wizards are pre-defined complex objects that perform specific functions or have specific appearance, such as sliders and meters.
- ActiveX controls are controls that you can place in InTouch windows to perform specific functions such as to show current alarms.

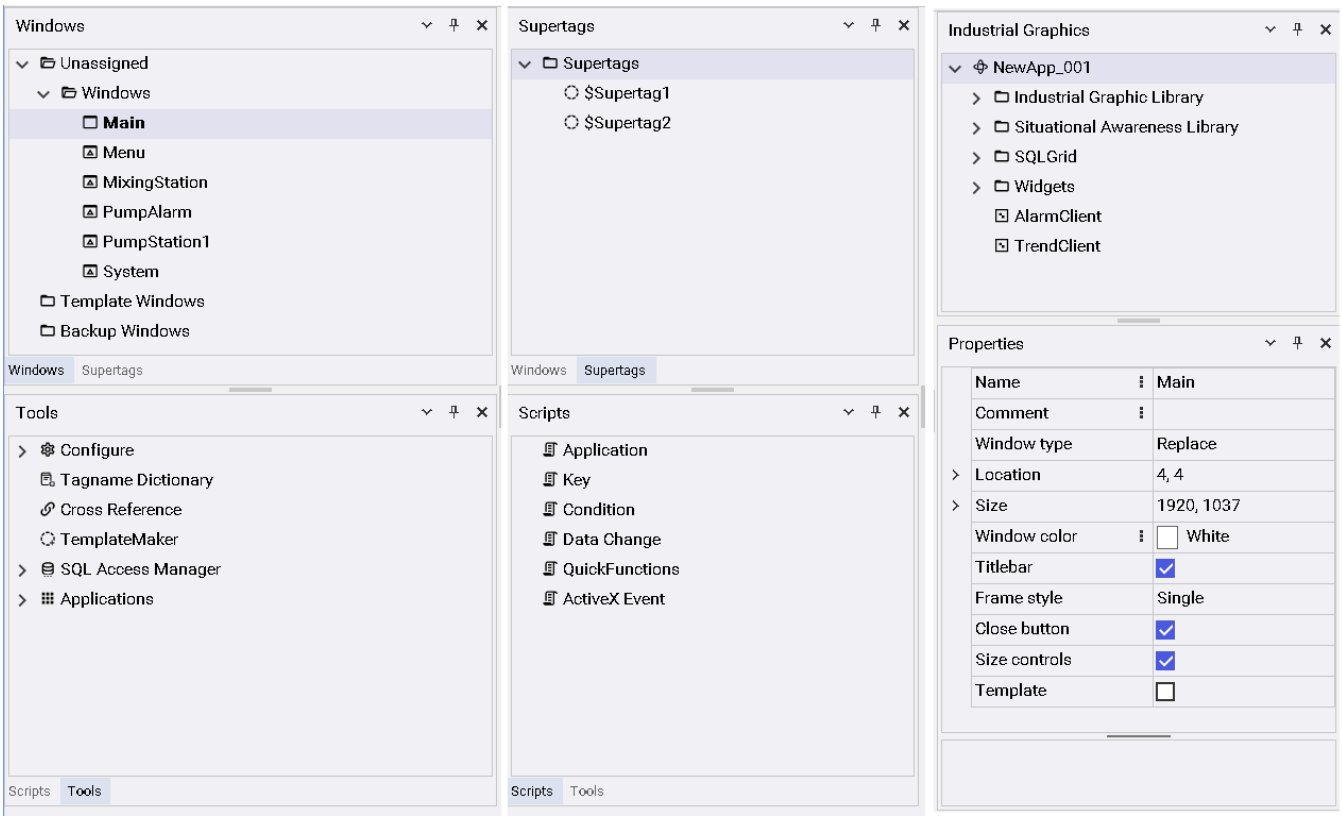
About the InTouch Application Manager

You use the InTouch Application Manager to manage most global tasks such as creating, deleting, and modifying your InTouch applications. Application Manager shows a list of your current InTouch applications. You select an application from the list to open in WindowMaker or WindowViewer.



Use the Application Explorer

The Application Explorer has tabbed sections for Windows/ Supertags, Scripts/ Tools, and Industrial Graphic Toolbox. Like other toolbars, they can be opened or closed, pinned or docked. By default, the panes appear in the positions shows in the illustration below.



These views give you access to all application windows, scripts, configuration menus, the Tagname Dictionary, and wizards.

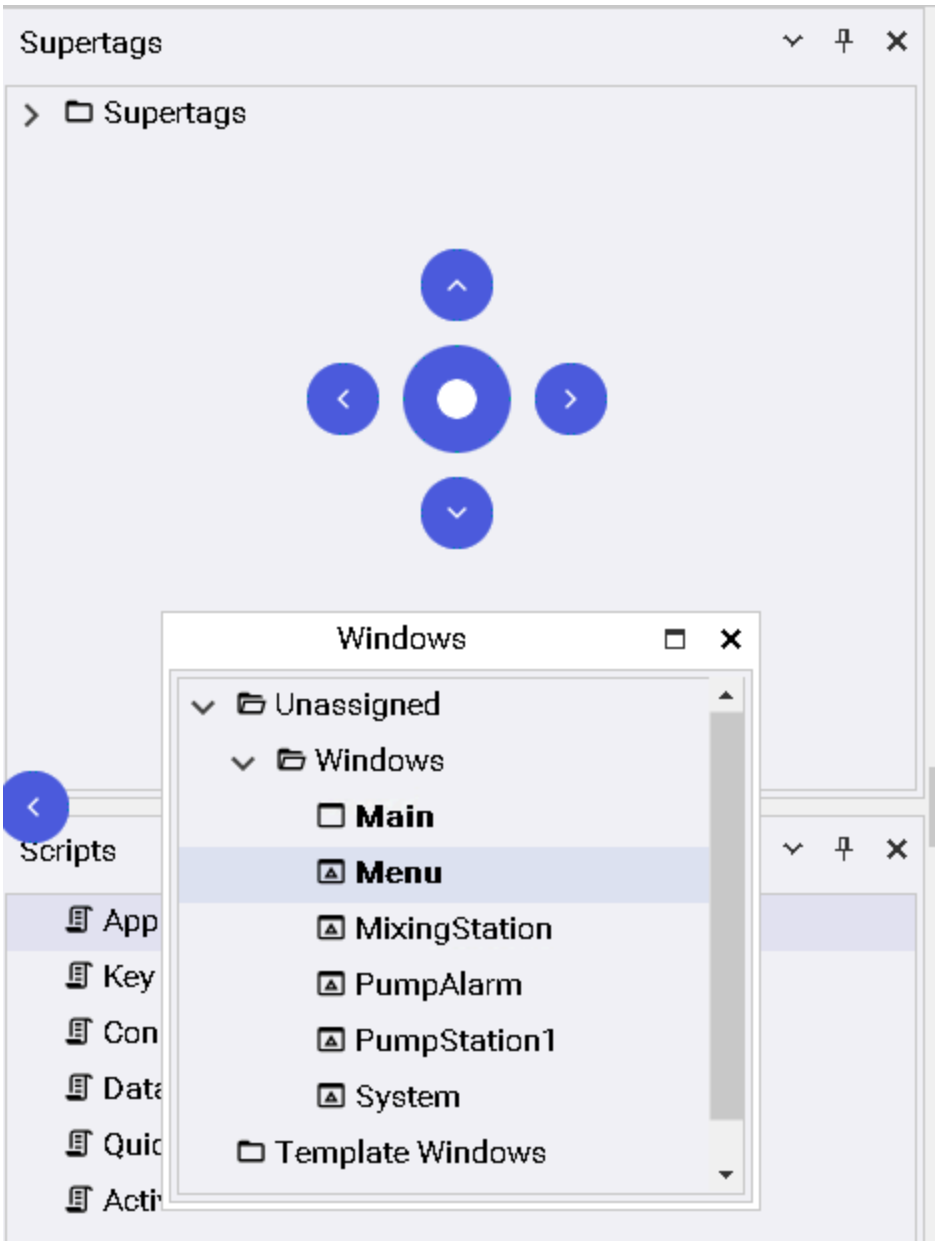
Dock the Panes

Each of these panes can be moved around and docked in the left, right, top, and bottom positions of the the screen. To dock a pane, select the header of the pane. The cursor turns into a two-sided arrow. Drag and dock the pane anywhere on the screen.

You can also use the navigation aid to dock the floating panes. In a particular position on the screen:

- Select the upward arrow to dock the pane towards the top of the screen.
- Select the downward arrow to dock the pane towards the bottom of the screen.
- Select the right arrow to dock the pane towards the right-side of the screen.
- Select the left arrow to dock the pane towards the left-side of the screen.

The panes can be docked vertically as well as horizontally.



When the position of a pane is changed, the customization is retained even after the closing and reopening the WindowMaker.

The changes to the position and docking of the panes is stored in the DockSettings.xml. If you face any issues with the navigation after changing the dock settings, do the following:

1. Close WindowMaker.
2. Navigate to **C:\Users\<username>\AppData\Local\Wonderware**.
3. Locate the file DockSettings.xml and delete it.
4. Open WindowMaker again.

Navigate in the Application Explorer

You can expand or collapse the folders in either of the Application Explorer toolbars.

The **Applications** view shows other installed applications.

Expand or collapse the Application Explorer folders

1. Double-select a folder or icon to expand and show the group members.
2. Double-click on a member to open that member.

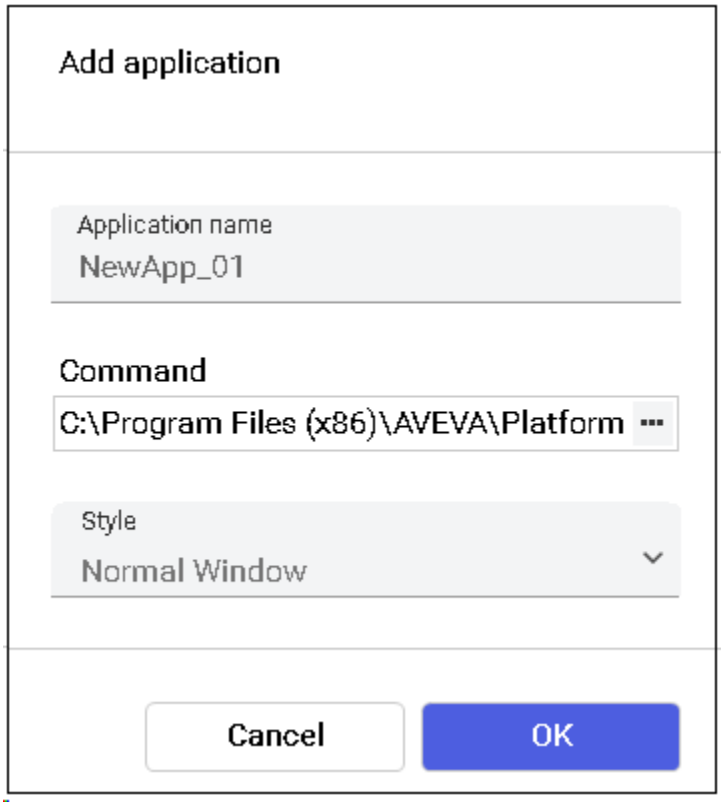
Add applications to the Application Explorer

The WindowMaker Application Explorer can start other applications from within WindowMaker. For example, you can run an I/O Server and configure it at the same time that you are developing your application. You can start third-party programs such as Windows Notepad, Wordpad, Microsoft Excel, Microsoft Word, Microsoft Paint and so on.

You can also configure the Application Explorer to open a specific file such as a document or spreadsheet.

Add a new application to Application Explorer

1. In the **Tools** pane, right-click **Applications**, and then select **New**.
   The **Add application** window appears.



2. In the **Application name** box, type the name of the application.
3. In the **Command** box, enter the full path for the application. Select the ellipsis button to browse for the application.

You can add command line parameters for the application in the **Command Line** box.

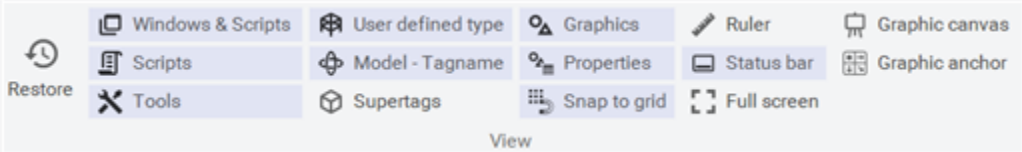
4. From the **Style** list, select how you want the application to appear when it starts up. The available options are Normal Window, Minimized, and Maximized.
5. Select **OK**.

   The application is added to the Application Explorer under Applications. You can now run the application at any time from WindowMaker.

Note: Do not add WindowViewer (view.exe) to the **Application Explorer**. The proper way to start WindowViewer is by selecting **WindowViewer** on the **File** menu, or by selecting the **Runtime** fast switch in the toolbar.

Manage toolbars

The toolbars in WindowMaker allow quick access to commands that are commonly used. When you start WindowMaker, by default, all toolbars are shown. Use the View menu to manage toolbars and to show or hide any of the toolbars.



You can float or dock a toolbar by dragging it. When you show a docked toolbar that was hidden, it reappears in its last docked location in the window. You can move any toolbar from its default docked position to any other location within the development window. Floating toolbars have title bars and they allow you to change their size.

Show or hide a toolbar

- On the **View** menu, select the toolbar icon.

Change the size of a floating toolbar

1. Move the cursor over any edge of the toolbar. The cursor changes to a double-ended arrow.
2. Drag the edge of the toolbar to move and resize the toolbar.

   As you move the cursor, a box appears to indicate the size the toolbar when you release the mouse button.

Set your WindowMaker preferences

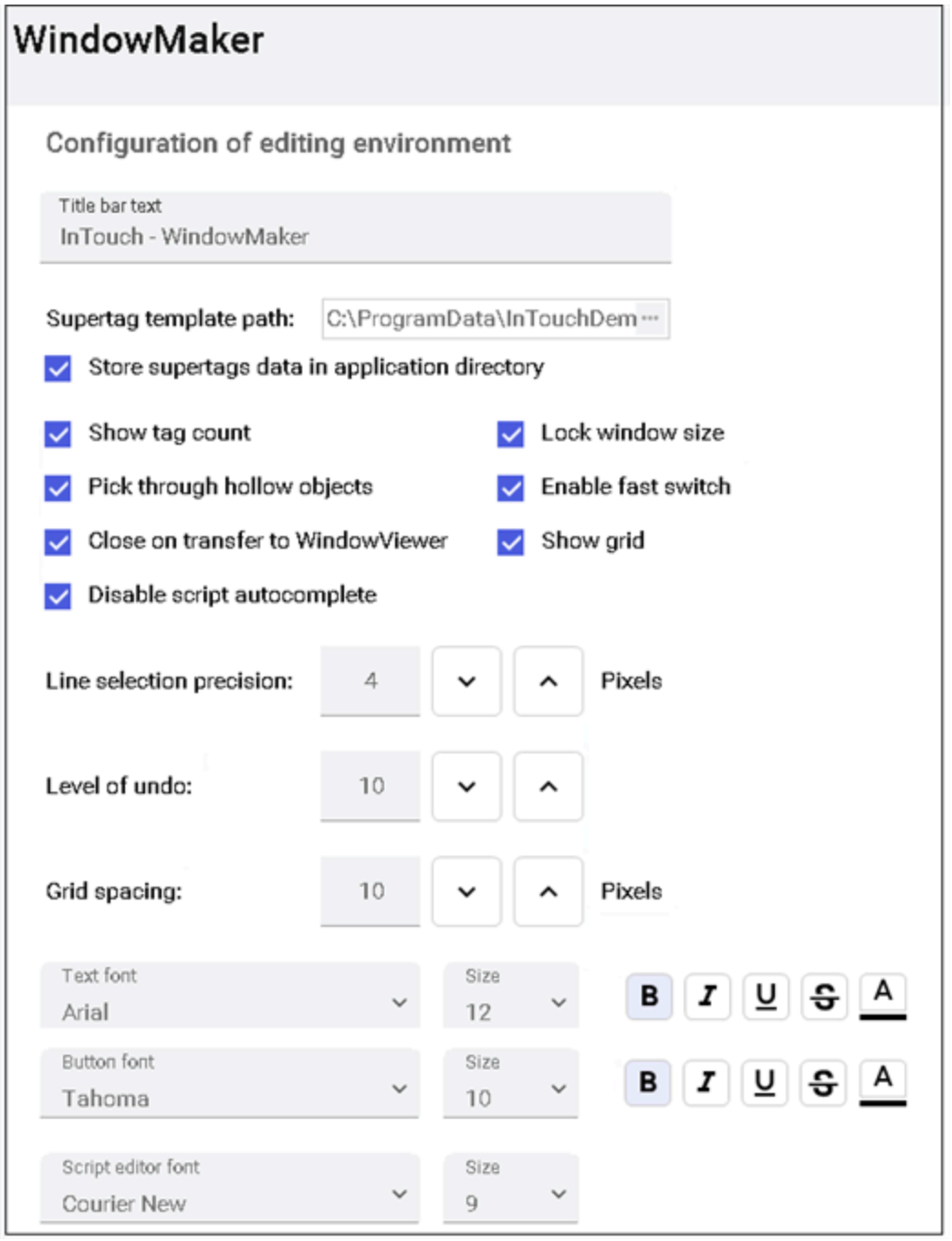
Using the WindowMaker configuration screen, you can configure preferences and options affecting the behavior of WindowMaker. You can:

- Change the title bar text.
- Display the grid or turn off the grid.
- Change the spacing between the pixels on the grid.
- Show the tag count.
- Change the default fonts for text and buttons.

- Set the precision for line selection.
- Set an option to close WindowMaker when switching to WindowViewer.
- Set an option to pick through hollow objects.
- Enable fast switch from WindowMaker to WindowViewer.
- Set the number of undo levels.

Set the properties for WindowMaker

1. On the **File** menu, select **Configure**, and then select **WindowMaker**.
   The WindowMaker configuration screen appears.



2. In the WindowMaker configuration of editing environment area, configure the appearance of the title bar. Do any of the following:
   - In the **Title bar text** box, type the text to appear in the title bar during design time.
   - In the **Supertag template path**, specify the path to store the Supertag template.
3. Select the **Store Supertags data in application directory** check box to store the Supertags data in the application directory. This will override the path specified in the **Supertag template path** field.
4. Configure miscellaneous window properties. Do any of the following:
   - Select the **Show tag count** check box to show the number of tagnames in your Tagname Dictionary in the menu bar. If you have a lot of tags, showing the tag count can impact the Tagname Dictionary

performance.

This is useful if you are creating an application with a limited Tagname Dictionary size. The tagname count does not include remote tagname references or system tags. Select **Update Use Counts** on the **Home** tab to find out your remote tagname reference usage.

- Select the **Lock window size** check box to lock the size of the InTouch application window.

  This allows you to convert the application to a different resolution without scaling the window and graphics.
- Select the **Pick through hollow objects** check box to select objects that are behind hollow objects.

  This enables you to select an object within a frame without having to send the frame to the background.
- Select the **Enable fast switch** check box to use the "fast switch" to toggle between WindowMaker and WindowViewer.

  The fast switch is the word **Runtime** that appears in the upper right corner of WindowMaker. In WindowViewer, it is the word **Development**.

  When enable the fast switch, WindowMaker automatically saves all changes made to all open windows before switching to WindowViewer.
- Select the **Close on transfer to WindowViewer** check box to close WindowMaker automatically when you start WindowViewer.

  The purpose of this option is to conserve limited memory. If memory is not an issue and you are moving often between WindowViewer and WindowMaker, do not select this option.

  When you select **Close on transfer to WindowViewer**, the reciprocal command, **Close WindowViewer**, on the **General Properties** tab in the **WindowViewer Properties** screen is also selected.
- Select the **Show grid** check box to show the grid.
- Select **Disable script autocomplete** to disable the display of autocomplete suggestions list box.
- In the **Line selection precision** box, type the number of pixels your cursor can be away from a line and still be able to select it.

  In most cases, the default setting of 4 works well.
- In the **Levels of undo** box, type the number of **Undo** and **Redo** levels to maintain.

  You can have up to 25 levels. If you type zero, the undo/redo function is turned off.

  One level represents one action. The **Undo** and **Redo** stacks are empty when you create a new window or open an existing window. Both stacks are emptied when you close a window.
- In the **Grid spacing** box, type the number of pixels between the grid coordinates.

5. You can also configure the font properties for the text, button, and script editor using the **Text font, Button font,** and **Script Editor font** fields respectively.
6. You can override these defaults in any window by using the **Font** settings on the respective screens.
7. Select **OK**.
8. Restart WindowMaker to apply any changes you made.

Mouse short cuts

Use the following short cuts to open dialog boxes and do other common tasks.

Access menu commands for items in WindowMaker

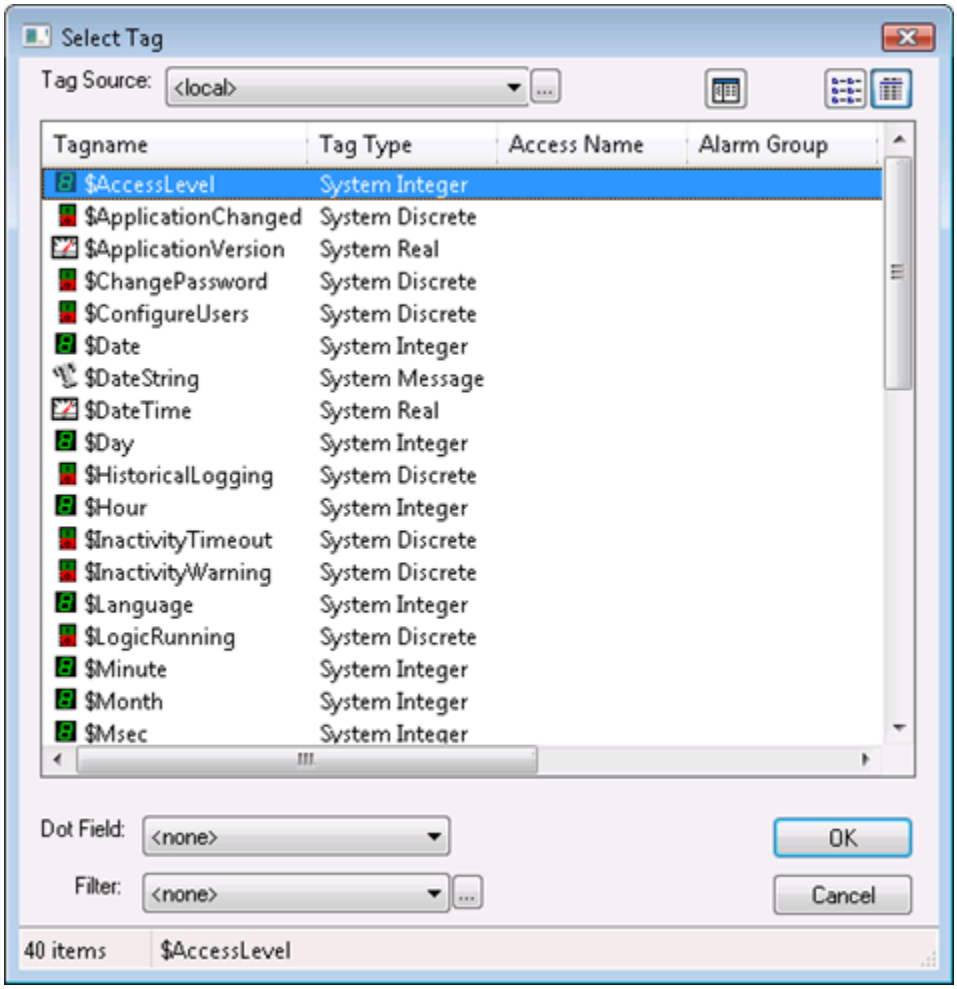
- Right-click on the item. Items include graphic objects, folder names in the views, and so on.

Open the Animation Links dialog box

- Double-click an object or symbol.

Open the Select Tag dialog box (Tag Browser)

- Double-click a blank expression input field within a link definition dialog box. The **Select Tag** dialog box appears.



Access the tag dotfields

- In any Tagname or Expression input box, type a tagname plus a period, then double-click to the right of the period. You can also type just a period and double-click to the right of it. The **Choose field name** dialog box appears showing all tagname dotfields.

Open a tagname definition in the Tagname Dictionary

- Double-click the tagname.

Use full screen mode

Full screen mode hides all program elements except open windows and floating toolbars.

Toggle full screen mode on or off

- On the **View** toolbar, select **Full Screen** to switch from normal to full screen mode.

  The **View** toolbar changes to the **Restore** toolbar automatically, and floats on top.

Manage windows from File menu

When you are opening, saving, closing, or deleting windows using the **File** menu, the names of all the windows that are valid for the selected command appear in a list.

View Windows List

You can view the windows in Icon view, List view, or Details view.

- Icon view: Select **Icons** to view the list of windows arranged in a grid.
- List view: Select **List** to view the windows arranged in a multi-column list.
- Detail view: Select **Detail** to view the list of windows alongside the description of the windows. The desciption is populated from the Comments field of the windows properties panel.

Select Windows

- To select a window, select the window icon or entry. The window gets highlighted.
- To select multiple windows, select the window icons or entries. The windows get highlighted.
- To select all windows, select the **Select all** check box.

Deselect Windows

- To deselect a selected window, select the window icon or entry again.
- To clear the selection of all windows, select to clear the **Select all** check box.

Search Windows

- To search for a window from the list, use the search bar.

Set font defaults

You can set font defaults for text objects and button objects that have text. You can override these defaults by using the fonts section to customize window or button text.

Set the font defaults

1. On the **File** menu, select **Configure**, and then select **WindowMaker**.

   The WindowMaker configuration screen appears.
2. You can configure the font properties for the text, button, and script editor using the **Text font, Button font,** and **Script Editor font** fields respectively.

   You can override these default settings by using the **Fonts** section.
3. Select **OK**.

Move objects with the arrow keys

In WindowMaker, you can use the **arrow keys** to move a selected object or group of objects.

When moving objects with the arrow keys, how far an object moves depends upon whether or not the grid is showing.

When the grid is showing, how many pixels an object moves depends upon the grid spacing, which is set on the WindowMaker configuration screen. The default setting is ten pixels between grid points.

When the grid is showing:

- Pressing an arrow key moves the object one grid point.
- Pressing SHIFT + an arrow key moves the object two grid points.
- Pressing CTRL + an arrow key moves the object four grid points.

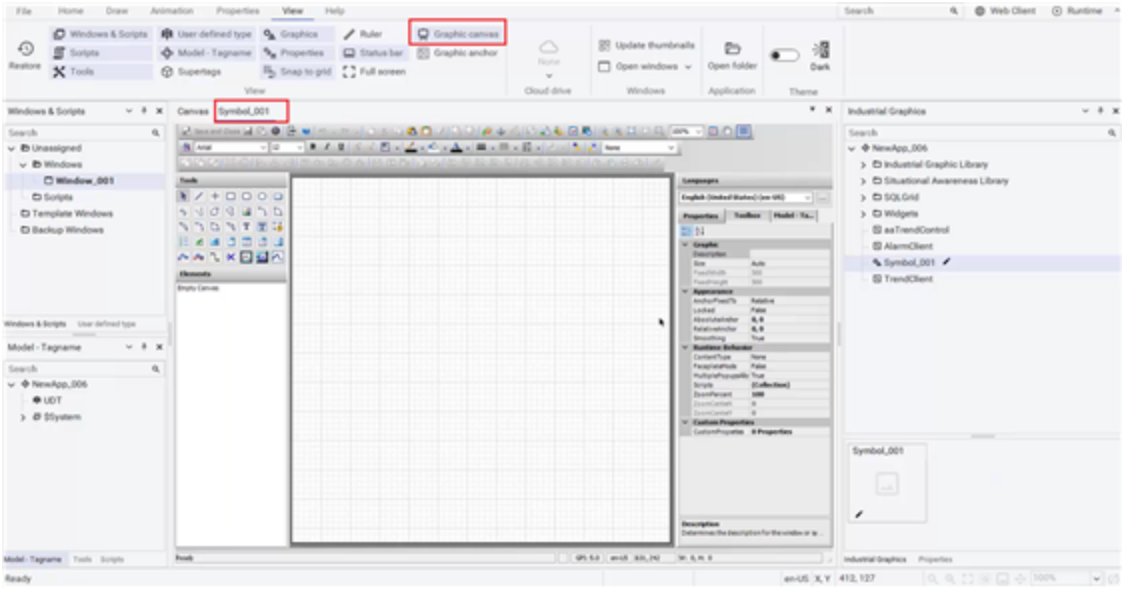
When the grid is not showing:

- Pressing an arrow key moves the object one pixel.
- Pressing SHIFT + an arrow key moves the object ten pixels.
- Pressing CTRL + an arrow key moves the object 50 pixels.

Use Graphic Canvas and Graphic Anchor

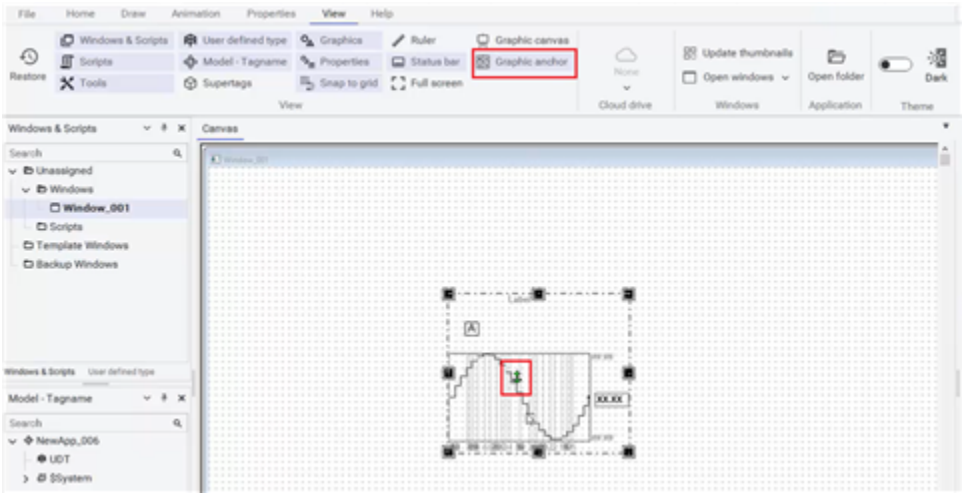
Graphic Canvas

In WindowMaker, under **View** menu, use the **Graphic canvas** option to view the Industrial Graphic Editor window as a tab in WindowMaker. That is, when the Graphic Canvas option is selected, if you open a graphic, it will open in Industrial Graphic Editor in a tab inside WindowMaker.



Graphic Anchor

In WindowMaker, under **View** menu, use the **Graphic anchor** option to view the anchor point on a graphic. To know more about anchor point refer to the Size Propagation and Anchor Points section of the AVEVA Industrial Graphic Editor help.



Use color palettes

You can use color palettes to apply color to static and dynamic properties of lines, rectangles, round rectangles, ellipses, polylines, polygons and text. You can select the background color for your windows and the transparent color for bitmaps that enable you to view objects behind bitmaps.

The palette offers you a wide range of color selections, up to 16.7 million colors. The available colors may be limited by your video card capability.

You can also define and add custom colors.

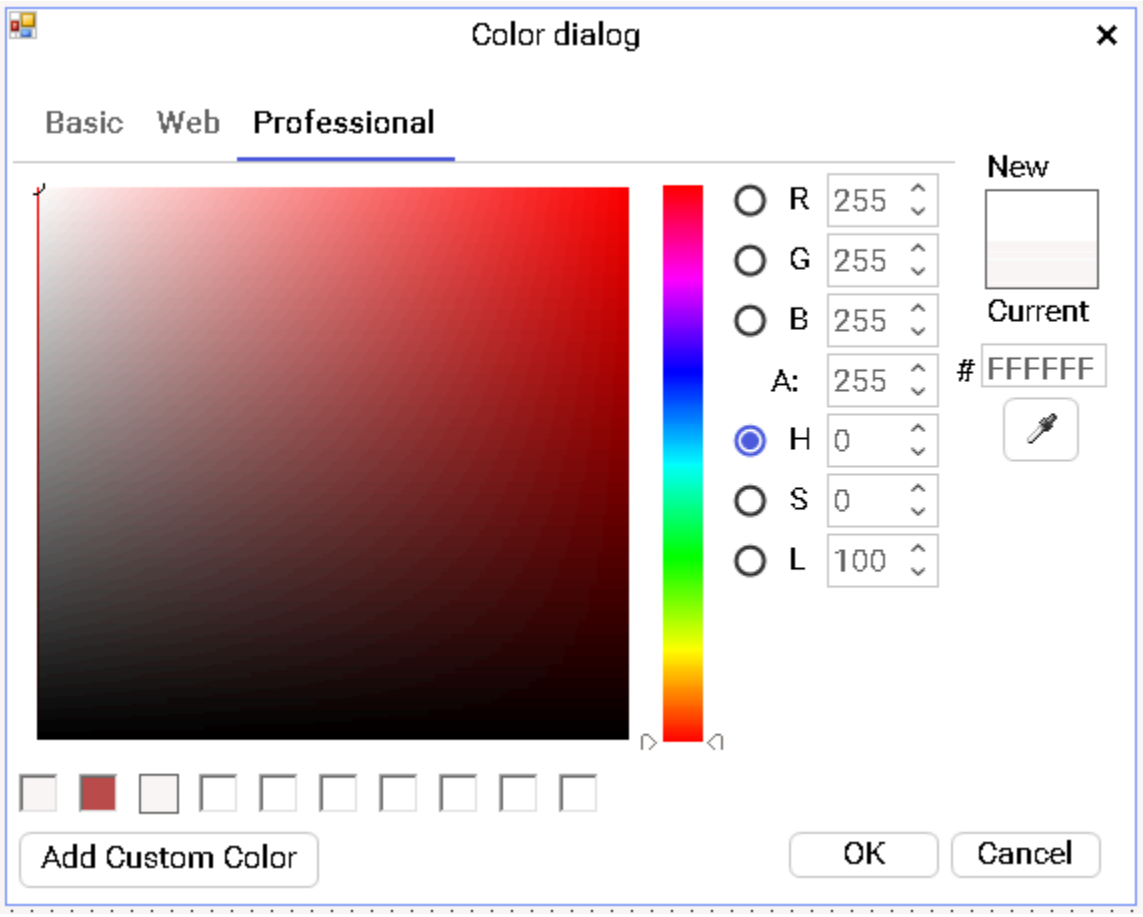
Open the color palette

The color palette appears when you select a colored square in the Properties panel or you select one of the color tools to apply line, fill, or text color to a selected object.

Open the color palette

1. In a configuration screen, select the colored square next to the Color field.

   The **Color dialog** appears.



2. Select the **Basic** and **Web** tabs to access other classic color palette.
3. Do any of the following:
   - Select anywhere in the color display and use the slider to adjust the color. To specify that you want 100% of the color with no white or black, press ALT + O.
   - Type values in the **Red, Green** and **Blue** boxes to define a color. You can experiment with these values by observing the color matrix. Notice the values for hue, saturation and luminosity also change.
   - Type values in the **Hue**, **Sat**,and **Lum** boxes to define a color. As you change any of these values, the red, green, and blue scales change to match.

     Hue is a discrete color value, where 0 is red, 60 is yellow, 120 is green, 180 is cyan, 200 is magenta, and 240 is blue.

     Saturation is the amount of color in a specified hue, up to a maximum of 240.

     Luminosity is the brightness of a color.

4. Select **OK**.

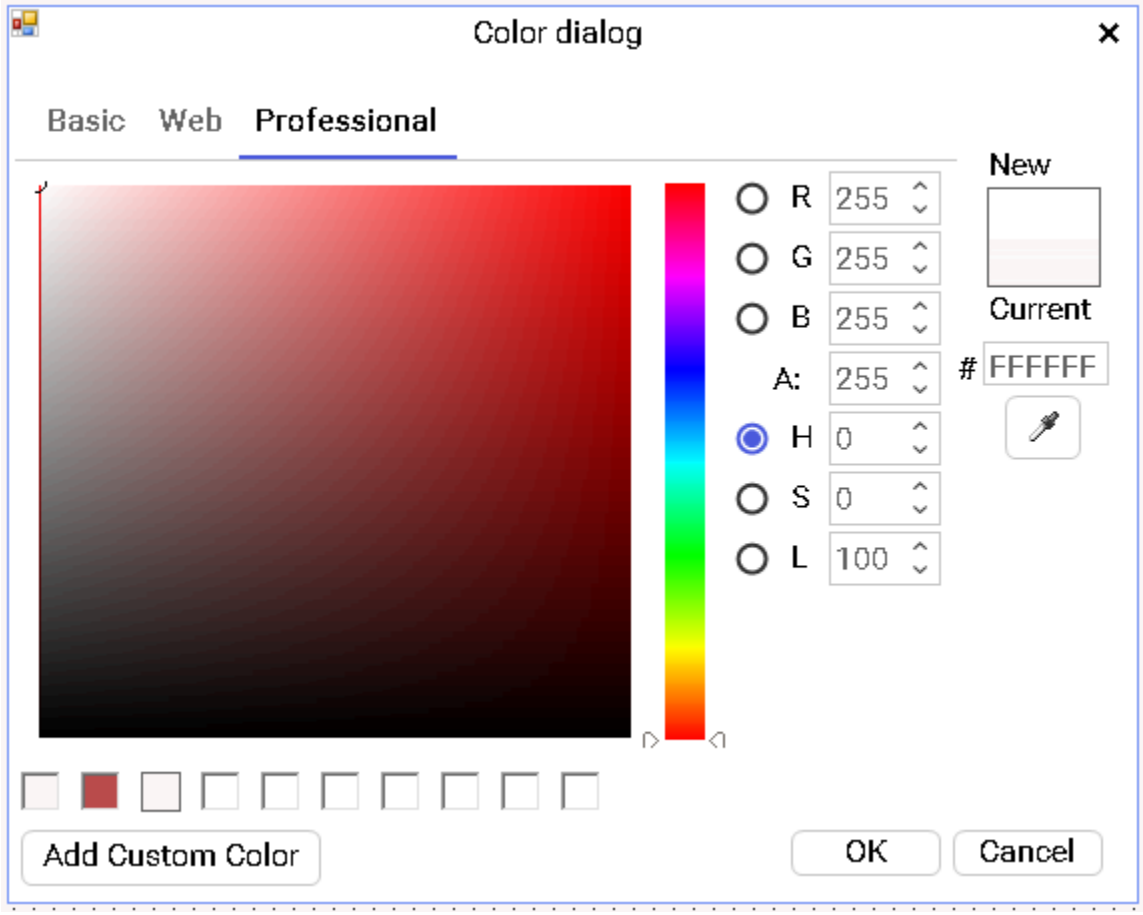
   The color dialog closes and the color you selected is applied.

Create custom colors

You can create a palette of custom colors.

Create a custom color

1. Open the color palette.
2. In the **Custom Palette** area, select a color and select **Add Custom Color**.



3. View the resulting color in the **Color|Solid** box.

   If your monitor is set to display 256 colors, the **Color|Solid** box might show two colors. The right side shows how the selected color appears as a solid color. The left side shows the dithered color, or the approximation of the specific color using two of the 256 colors.

4. Select **OK**.

You can also create custom colors using the eye dropper tool.

Note: Use this feature for creating transparent bitmaps.

Select a custom color with the eye dropper tool

1. Open the **Color Palette**.
2. Select the eye dropper tool and then select the color that you want to add.

Pan and zoom

You can zoom in and out get a better look at the elements that you are editing to ensure objects line up exactly or are positioned correctly.

The Pan and Zoom toolbar appears by default at the bottom right of the screen. It can be floated or docked in other locations, like the other toolbars.

If you have specified a target resolution that is different than your screen resolution, the boundary canvas will adjust accordingly upon panning or zooming.

You can:

- Zoom in and out from 100% to 500%.
- Zoom to a specific area with the rubber band tool.
- Zoom the window to a specific percentage.
- Select and drag to pan the window.
- Return to the normal default view.

Show or hide the Pan and Zoom toolbar

- On the **View** menu, select **Status Bar**.

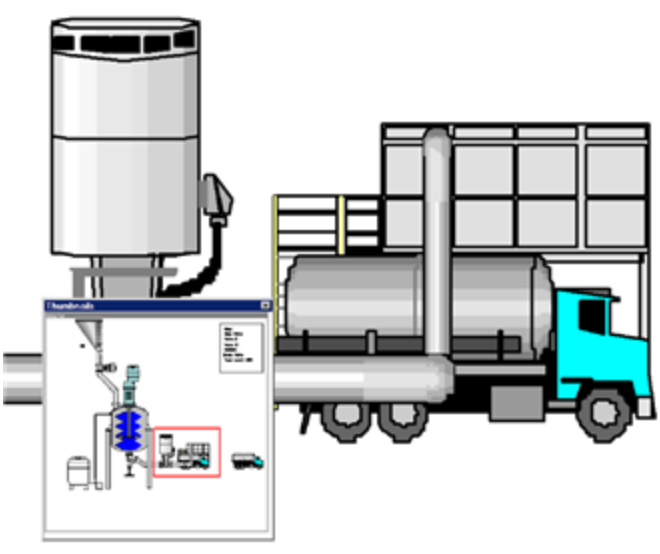
Use the Thumbnail window to pan and zoom

When you zoom in to a detail of your application window, the thumbnails window shows you the relationship of the detail to the whole application.

The **Thumbnails** window gives you both an overview and a detail of the development area.

Show or hide the **Thumbnails** window by selecting the **Thumbnails** button.

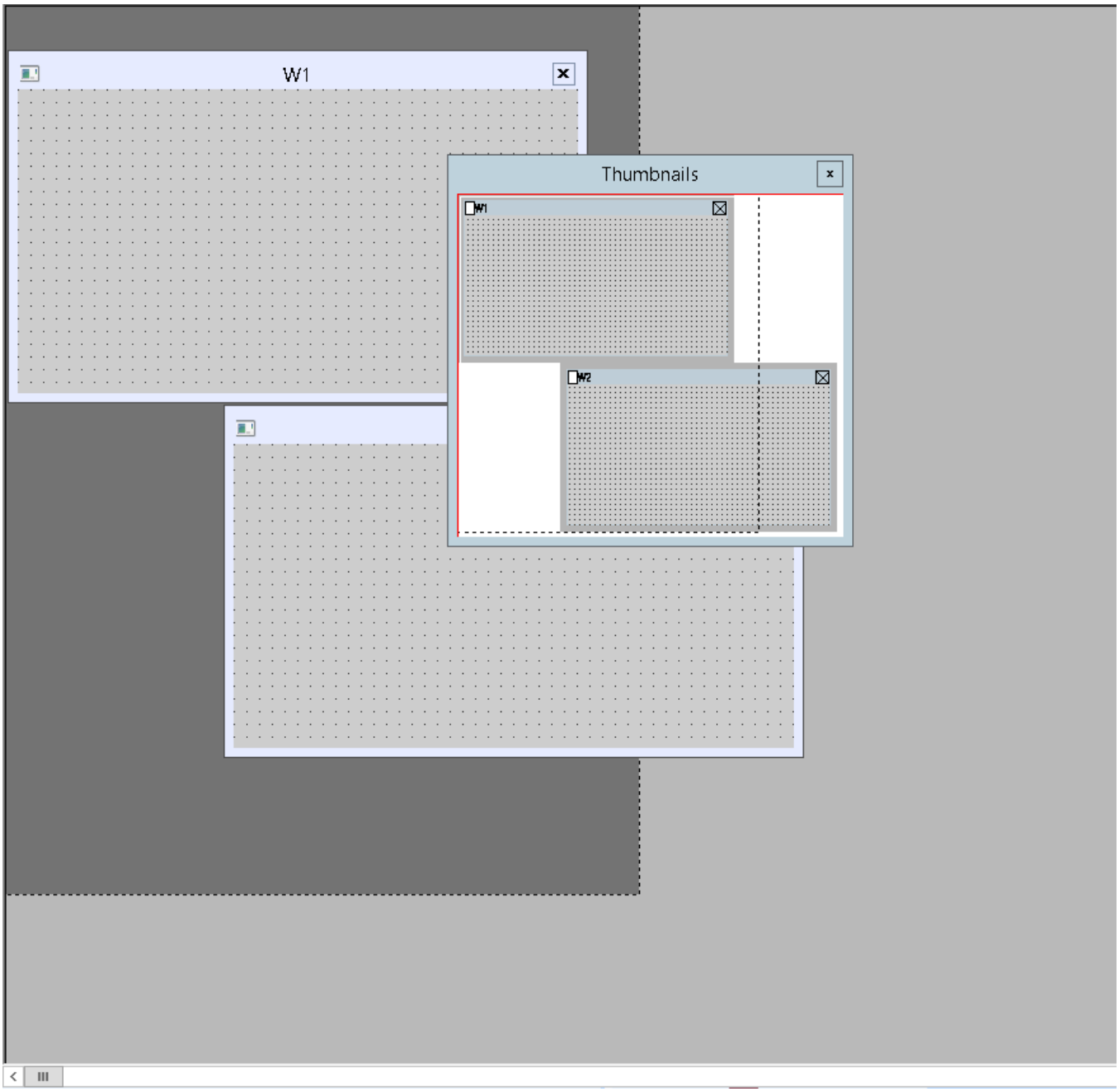
A red rectangle is the boundary of the zoomed area within a window.



- Drag the red rectangle to show a different part of the window.
- Select on a different area of the window to move the rectangle to that area.
- Resize the rectangle to change the zoom level of the display area.

The thumbnail view shows a white rectangle for objects that you cannot zoom in on, such as an ActiveX control.

If you have specified an application target resolution that is different from your screen resolution, the boundary canvas outlining your specified target resolution dimensions will also appear in the thumbnail view.



Note: Application windows exceeding the dimensions of the target resolution boundary will still appear in the thumbnail view as seen in the example above.

Use the mouse wheel to zoom and pan

If your mouse has a wheel, you can press the CTRL key and roll the wheel to change the zoom level of your image.

- As you roll the mouse wheel, each select changes the zoom level by 20%.
- You can also place the cursor in the InTouch window and press the mouse wheel to navigate within the window. When you press the mouse wheel, an icon with four arrows appears. Move the mouse to navigate

in the window.

Pan and Zoom Limitations

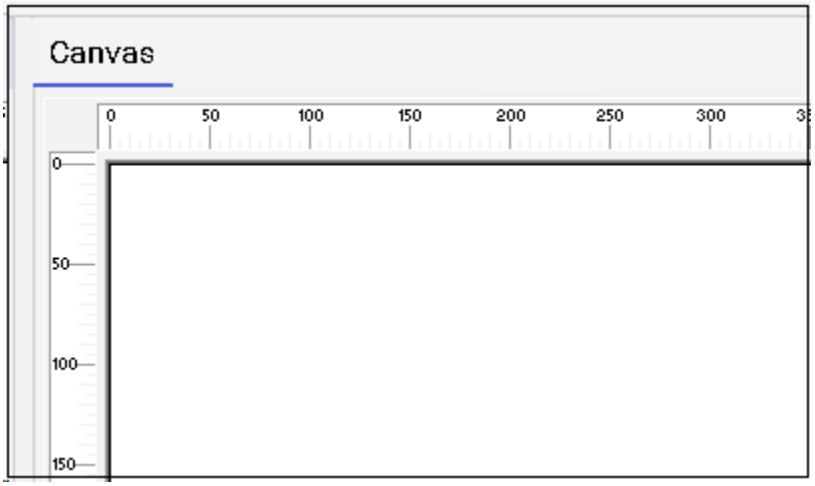
Panning and zooming do not apply to the following controls:

- ActiveX controls
- The Distributed Alarm Object
- The 16-Pen Trend
- Text boxes
- Check boxes
- List boxes
- Combo boxes
- Radio Group objects

If one of these controls is in the visible area when the view is zoomed more than 100%, a rectangle with the name of the control appears in the area occupied by the control.

Use the screen grid and ruler

You can show a grid and ruler in the Development area to help you arrange and align objects.



Snap objects to the grid

You can make your objects snap to predefined grid points by selecting **Snap to Grid**.

By default, the grid is set to 10 pixels and to be visible when you start WindowMaker. You configure the pixel interval for the grid in the WindowMaker configuration screen.

To see the grid, you must select **Show Grid** on the WindowMaker configuration screen and select **Snap to Grid** on the **View** menu.

Configure the grid

1. On the **File** menu, select **Configure**, and then select **WindowMaker**. The WindowMaker configuration screen appears.
2. In the **Grid spacing** box, type the number of pixels to space between coordinates.
3. Select the **Show grid** check box if you want to see the grid when **Snap to Grid** is selected.

   If you do not select **Show Grid**, no grid is visible in your windows when you select **Snap to Grid**.

Use the ruler

Use the rulers for precision alignment of the objects in your windows. The rulers appear across the top and along one side of your development environment window.

Show or hide the rulers

1. On the **View** menu, in the **View** group, select **Ruler**.
2. Repeat the step to hide the ruler.

About WindowViewer

WindowViewer provides the run-time environment for InTouch applications. Based upon your application's operational requirements, you can configure how WindowViewer supports an application. For example, depending on your application's security requirements, you can configure the menus and commands available to operators from WindowViewer.

Customize your run time environment

You can set properties to customize your run time WindowViewer environment.

- General properties determine the environmental conditions to run an InTouch application.
- Window configuration properties determine the menus, commands, and window components accessible to users as they interact with an InTouch application running in WindowViewer.

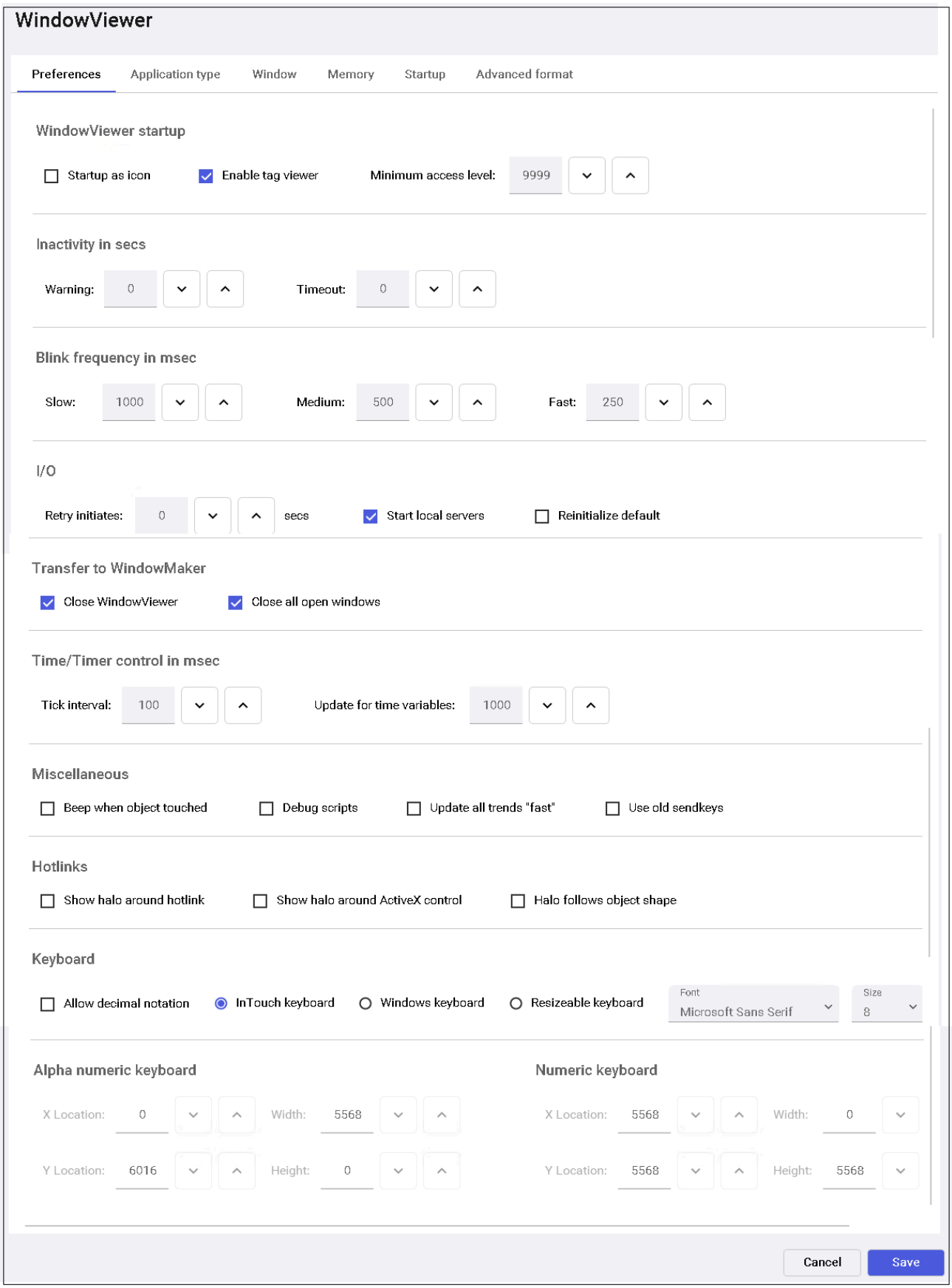
Configure general WindowViewer properties

You can configure a set of general properties that determine the characteristics of WindowViewer as it runs an InTouch application. After you modify these general properties, you must restart WindowViewer for your changes to become effective.

Note: Close WindowViewer before changing the Regional Formats of the Operating System.

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.



3. On the **Preferences** tab, in the **WindowViewer Startup** section, do the following:
   - Select the **Start up as Icon** check box if you want WindowViewer to start up minimized.
   - Select the **Enable tag viewer** check box to allow the Tag Viewer application to be started at run time. The Tag Viewer allows you to watch and monitor tags and modify tag values at run time. For details, see Enable Tag Viewer in AVEVA™ InTouch HMI Application Run Time.
   - In the **Minimum access level** box, type the minimum security access level required to run the Tag Viewer application. The value must be between 0 and 9999. For details, see Enable Tag Viewer in AVEVA™ InTouch HMI Application Run Time.
4. In the **Inactivity in secs** section, set warning and time-out periods for operator inactivity.

   For more information about setting warning and time-out periods, see Configure an inactivity time-out in AVEVA™ InTouch HMI Management.
5. In the **Blink frequency in msec** section, type the interval length in milliseconds for your **Slow, Medium**, and **Fast** blink animation links.
6. In the **I/O** area, do the following:
   - In the **Retry Initiates** box, type the number of seconds to wait before the InTouch application retries connecting to an I/O Server after a failed connection attempt. The **I/O Retry Initiates** box has no effect when InTouch can successfully connect to the I/O server the first time.
   - Select the **Start local servers** check box if you want a dialog box to appear when you start WindowViewer and the I/O Server you are trying to communicate with is not running.
   - Select the **Reinitialize default** check box if you want to reinitialize Access Names with their default settings. Current values assigned to Access Names are ignored and they are reinitialized with their original settings.
7. In the **Transfer to WindowMaker** area, do the following:
   - Select the **Close window viewer** check box if you want WindowViewer to automatically close when you start WindowMaker.

     When you select this option, the Close on Transfer to WindowViewer option located on the WindowMaker configuration screen is automatically selected too. If memory is not an issue, and you are using the fast switch to move between WindowViewer and WindowMaker, this option should be cleared.
   - Select the **Close all open windows** check box if you want all open windows to close automatically when you transfer from WindowViewer to WindowMaker.
8. In the **Time/Timer Control in msec** area, do the following:
   - In the **Tick interval** box, type the interval that the InTouch HMI uses to check its internal timers.

     This interval determines when Application While Running, Window While Showing, Condition While On True/On False, Key and Touch Pushbutton Action While Down QuickScripts can be started.

     This option sets the value for TimerTickInterval parameter in the INTOUCH.ini file. You should set the Tick Interval no longer than 50 msec for a script scheduled to run every 100 msec. On computers running Windows XP or Windows 2003, the lower tick interval is 10 msec.
   - In the **Update for time variables** box, type the interval in milliseconds that time is updated for system tags like $Msec, $Second, or $Minute.

     We recommend that you use the default setting of 1000 milliseconds. Setting this option to zero prevents time variables from being updated.
9. In the **Miscellaneous** area, do the following:
   - Select **Beep when objects touched** if you want the InTouch application to emit a beep sound when

operators select touch-sensitive objects on a window.

- Select **Debug scripts** if you want a message to be written to the Logger each time a QuickScript runs.

  If you select **Debug** from the **Window Configuration** property sheet, you can switch QuickScript logging on and off from WindowViewer's **Special** menu.
- Select **Update all trends "fast"** if you want your trend objects to be updated more quickly.

  Select this option only if no objects overlap run-time trends on the application window. If you select this option and any object overlaps a trend, the object can be drawn incorrectly.
- Select the **Use old sendkeys** check box if you are using an international application developed with InTouch version 3.26 or earlier.

10. In the **Hotlinks** area, do the following:
    - Select the **Show halo around Hotlink** check box if you want an object on the run time screen to be highlighted when the user moves the cursor over it.
    - If you want a halo around Active X controls, select the **Show halo around ActiveX control** check box.
    - Select the **Halo follows object shape** check box if you want a highlight halo around the contours of an object when the user moves the cursor over it.
11. In the **Keyboard** area, select the type of keyboard you want to use, if any.

    For more information about setting keyboard options in WindowViewer, See Animate objects in AVEVA™ InTouch HMI Application Development.
12. Select the **Allow decimal notation** checkbox, to allow only decimal values during runtime, and restrict non-numeric characters. (The **Allow decimal notation** option not been tested on a non-English language Operating System.)
13. In the **Alpha numeric keyboard** and **Numeric keyboard** areas, configure the X, Y location as well as the Width and Height values.
14. Select **Save**.

Configure visual characteristics of WindowViewer

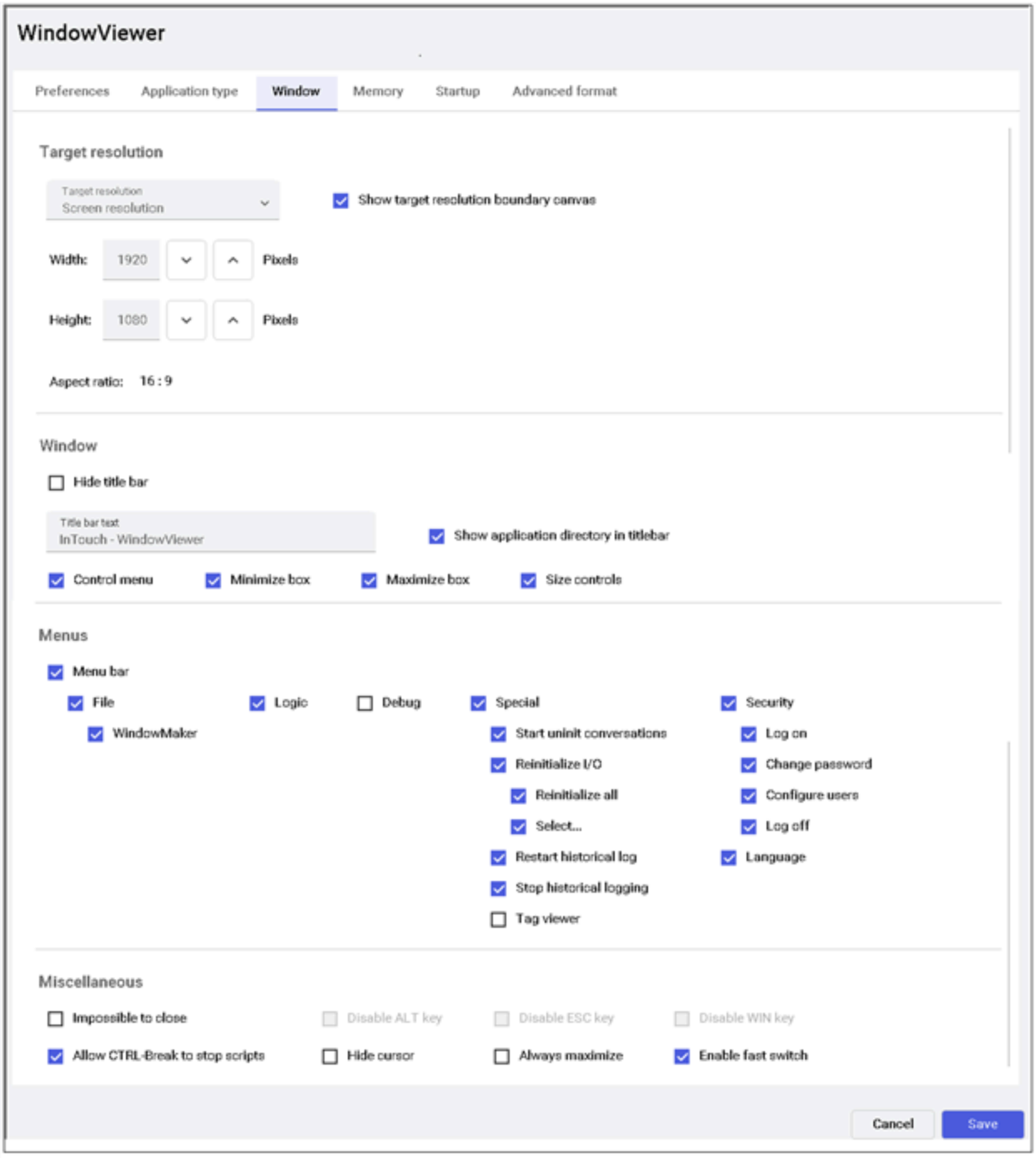
You can configure properties that determine the visual characteristics of WindowViewer as it runs an InTouch application. These properties determine the menus, commands, and standard controls that appear on a WindowViewer window.

Configure WindowViewer visual characteristics

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.
3. Select the **Window** tab.

   Select the check boxes for the visual characteristics.



4. Restart WindowViewer.

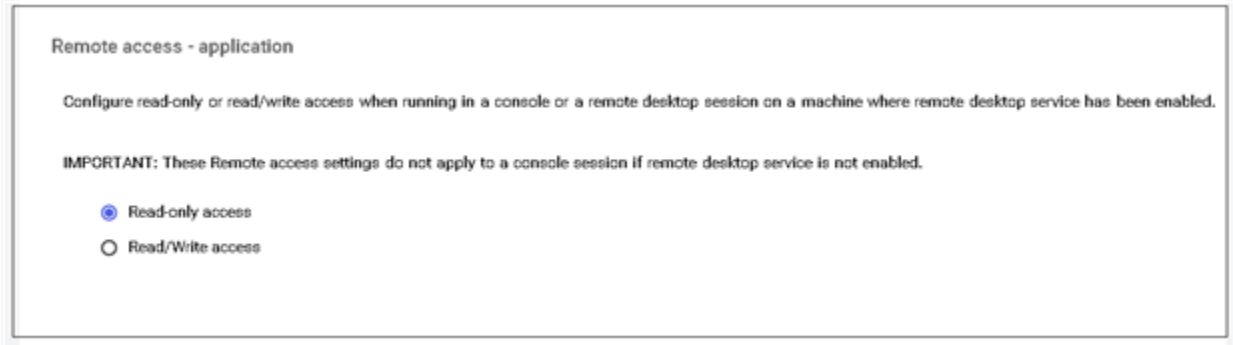
Configure user access to applications running in remote sessions

You can assign Read Only access to a distributed InTouch application running in a Remote Desktop Connection or Terminal Services session. Read Only access is appropriate for non-operators who need to view an application, but who should not have write permission. For example, managers need the capability to view an application on a mobile device with InTouch Access Anywhere over an unsecured public network. Using Read Only access provides better protection for distributed InTouch applications accessible from public networks.

Configure user access to applications running on remote nodes

1. Open the application in WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.
3. In the WindowViewer configuration screen, select the **Application type** tab.

   The **Remote Access** section appears with options to grant Read/Write or Read Only access to the application open in WindowMaker. Read/Write access is the default.



4. Make your selection and select **OK** to close the dialog box.

During initialization, WindowViewer verifies if the application is running in a remote session and is specified as Read Only. Also, a check is made that the remote node has a Read Only license. If all of these conditions are true, the application's InTouch Links and User Input animations are viewable only in Read Only mode.

About managing memory for WindowViewer

You can configure how WindowViewer uses memory for windows and popup symbols to improve performance at run time. Windows and popup symbols displayed using ShowSymbol animation and the ShowGraphic script function can be kept in memory at run time in certain conditions to allow for fast retrieval.

In-memory caching of Industrial graphics is available only in Managed or Published InTouch applications. This capability is disabled in Standalone InTouch applications.

You can also specify the interval for a periodic memory health check and settings for the heap memory segment. Each time a new popup symbol opens, it triggers a memory health check regardless of the pre-set interval.

Fast switching to WindowViewer or from WindowViewer to WindowMaker clears both the graphic cache and the window memory cache.

For information on memory management for symbols displayed by the ShowGraphic function, see Industrial Graphic Editor User documentation.

Configure memory usage for WindowViewer windows

You can configure how WindowViewer uses memory for application windows to improve performance at run time.

Reopening closed windows that have been cached retrieves them from memory rather than loading them from disk in certain conditions.

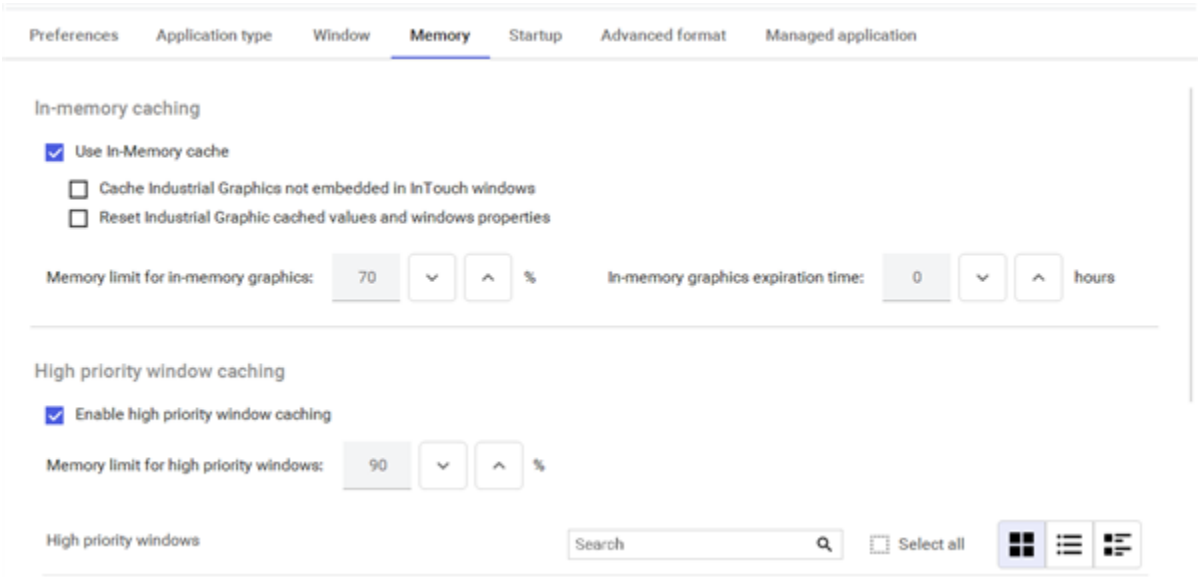
You can designate certain windows to have a higher priority for memory usage and configure separate memory settings just for those windows.

After you modify any of the WindowViewer memory options, you must restart WindowViewer to apply your changes.

Set the memory properties

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.
3. Select the **Memory** tab.



4. In the **In-Memory Caching** area, do the following:
   a. Select the **Use In-Memory Cache** check box if you want to save all closed windows to be cached in memory at run time.
   b. Select the **Cache Industrial Graphics not embedded in InTouch Windows** check box to enable Industrial graphics symbol caching.
   c. Select the **Reset Industrial Graphic cached values and windows properties** checkbox if you want to reset custom properties and window properties, when the window is closed and reopened.
   d. In the **Memory Limit for In-Memory Windows** box, enter the limit for keeping closed in-memory windows in cache memory at run time. The default memory limit is 70% of process memory.

      If the memory limit is exceeded, the system automatically removes the oldest closed in-memory window from the cache at run time, unless it is marked as a high-priority window.

      The memory limit for in-memory windows will always be less than the memory limit for high-priority windows.
   e. In the **In-Memory Window Expiration Time** box, enter the maximum duration for which the closed in-memory windows will remain in cache memory at run time. You can enter a value between 0 and 8760 hours. The default value is 0 hours, which designates no time limit.

      Note: InTouch windows and Industrial Graphics shares the in-memory cache. If the set memory limit is exceeded, the system automatically removes the oldest in-memory graphic from the cache.

   The memory limit or the expiration time limit is applied depending on which limit is reached first.
5. In the **High Priority Window Caching** area, do the following:

a. Select the **Enable High Priority Window Caching** check box to allow some windows to be marked as high priority. These windows will always be kept in cached memory after they are closed at run time.

b. In the **Memory Limit for High Priority Windows** box, enter the limit for keeping closed high-priority windows in cache memory at run time. The default memory limit is 90%. The system removes the oldest in-memory window first, and then removes the oldest high-priority window when the percentage of used memory exceeds this limit at run time.

c. In the **High Priority Windows** box, select the windows you want to mark as high priority.

Note: When the **Use In-memory cache** or **Enable high priority window caching** checkboxes are enabled, then the **Reset Industrial Graphic cached values and windows properties** checkbox will also be enabled. When the **Reset Industrial Graphic cached values and windows properties** checkbox is selected, the reset action will also affect the **Use In-memory cache** or **Enable high priority window caching** options.

6. Select **Save**.

Configure the memory health check interval

The system checks the memory and Graphical Device Interface (GDI) count at regular intervals. If the memory or GDI count limit is exceeded, the system starts removing windows. By default, this interval is set at 5 minutes.

If you want to change the interval, you can add a new entry in the InTouch.ini file and then specify a new interval value.

If you want to turn off the check, you can add the new entry and set the value to 0.

After modifying the interval, you must restart WindowViewer to apply the changes.

For more information on how windows will be removed, see Configure memory usage for WindowViewer windows.

Configure the memory health check interval

1. In the application folder, open the InTouch.ini file.
2. Under the [Intouch] section, add the following script, where *<interval>* is the desired interval, in minutes:

```
MemoryHealthCheckInterval = <interval>
```

Opening a new popup symbol or a new window will trigger a memory health check regardless of the pre-set interval.

Configure wwHeap memory settings

wwHeap is a memory manager that manages the heap memory segment. The memory manager provides a mechanism for one or more programs to share virtual memory.

Caution: Modify the wwHeap memory settings only if you are experiencing memory conflicts reported in the Logger.

You can configure the wwHeap Memory settings by specifying the wwHeap size and start location. The default size, default start location, and allowable location range vary by operating system.

The default sizes are described in the following table:

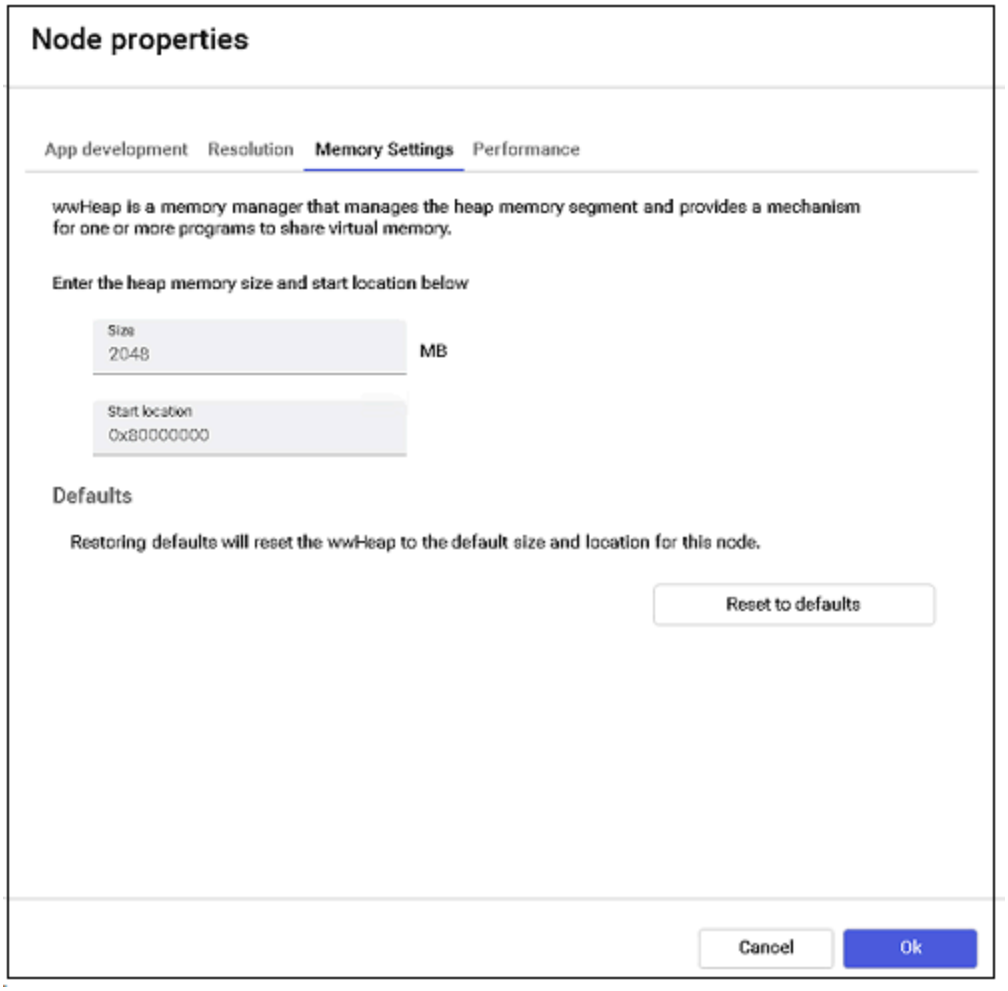
Operating System	Default Size
32-bit	1519 MB
32-bit with the /3GB switch enabled	2048 MB
64-bit	2048 MB

The default locations and allowable location ranges are described in the following table:

Operating System	Default Start Location	Allowable Range
32-bit	0x21000000	0x00010000 to 0x7FFEFFFF
32-bit with the /3GB switch enabled	0x40000000	0x00010000 to 0xBFFEFFFF
64-bit	0x80000000	0x00010000 to 0xFFFEFFFF

Configure wwHeap memory settings

1. Start **Application Manager**.
2. On the **Tools** menu, select **Node Properties**.
   The **Node Properties** screen appears.
3. Select the **Memory Settings** tab.



4. Do the following:
   - In the **Size** box, enter the size of the wwHeap memory. You can enter any value between 1 MB and 2048 MB.
   - In the **Start location** box, enter the start address.
5. To restore the default values, select **Restore to Defaults**.
6. Select **OK**.

Select the regional settings WindowViewer option

InTouch WindowMaker includes a WindowViewer Advanced Format **Regional Settings** configuration option.

To enable numeric formatting by regional locale, the **Regional Settings** option must be selected during design time to format Industrial graphic numbers to the country selected in the **Region** setting. By default, the **Regional Settings** option is disabled.

WindowViewer checks the OS Regional Settings only on startup. This means that you may need to restart WindowViewer if you do either of the following:

- Select the **Regional Settings** option while WindowViewer s running.
- Change the OS Regional Settings while WindowViewer is running with the **Regional Settings** option is selected.

Set advanced formatting properties

You can configure the number of decimal places to be shown at run time in WindowViewer, based on the size of the value. This feature is available only if you select Real in Value Display or User Input animation.

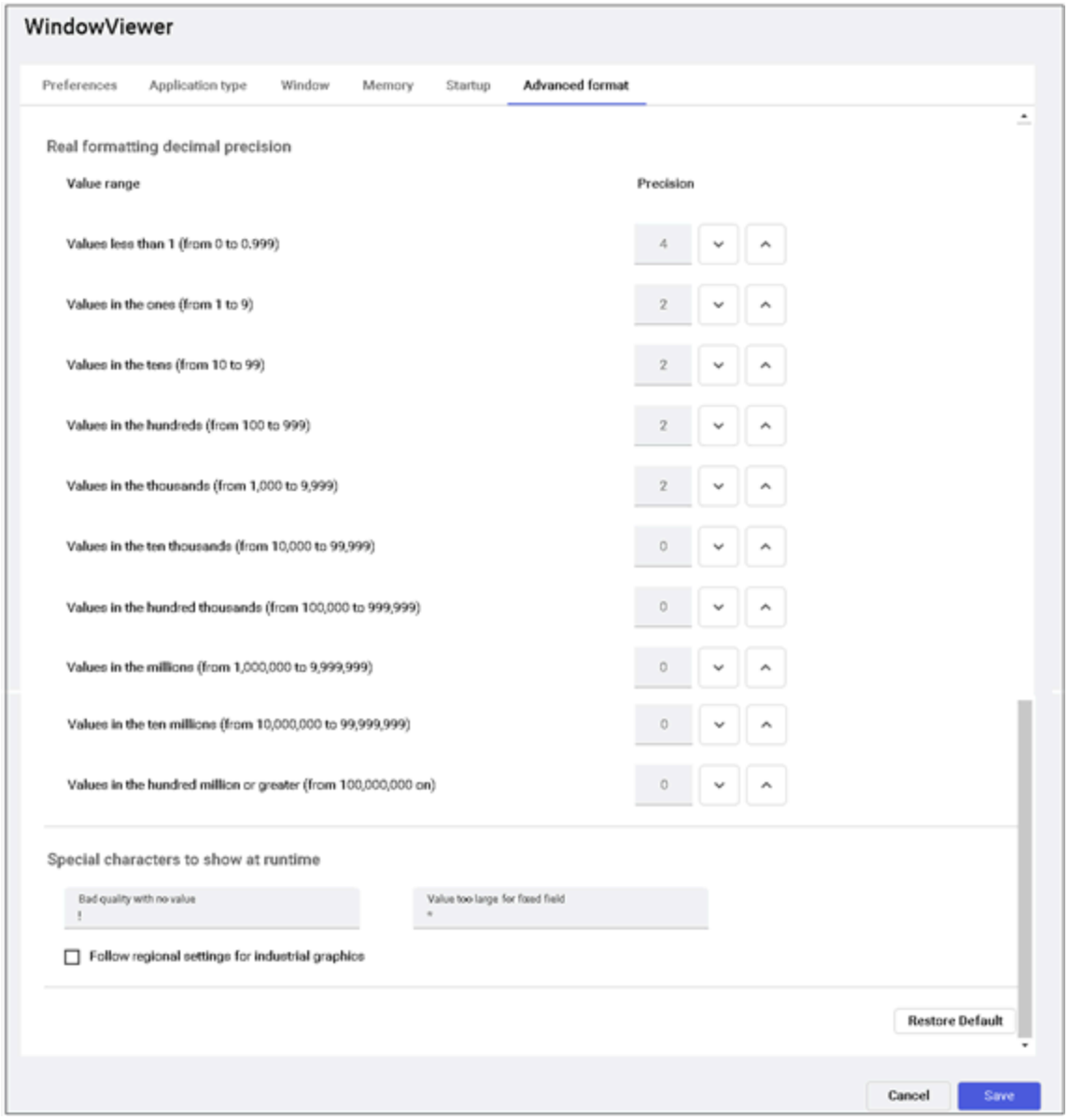
You can specify the special characters to be shown at run time if the data collected from the field devices is of bad quality or too large to be shown.

Numbers that appear in Industrial Graphics can be shown in the format of the country set as the home location with the Windows Control Panel's **Region** setting. During run time, Industrial Graphic numeric values can be displayed with thousands and decimal separators that match the numeric format of the country specified in the OS Regional Settings.

Set the advanced formatting properties

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.
3. Select the **Advanced Format** tab.



4. In the **Real Formatting Decimal Precision** section, enter the number of decimal places that you want to be shown at run time for each real type number range.
5. In the **Special Characters to Show at Runtime** section, do the following:
   - In the **Bad Quality with No Value** box, enter the character you want to be shown at run time when the quality of the data point is bad and there is no last known good value. The default character is **!**. You can enter any ASCII character, except a space.
   - In the **Value Too Large for Fixed Width** box, enter the character you want to be displayed at run time when the value is too large to be displayed. The default character is *****. You can enter any ASCII character, except space.
6. To show numbers within Industrial Graphics in the format of the country specified by computer's **Region** setting, select the checkbox for **Follow regional settings for Industrial Graphics**.

   By default, the **Regional Settings** option is disabled.

   Note: WindowViewer checks the OS Regional Settings only on startup. This means that you may need to

restart WindowViewer if you do either of the following:
1) Select the **Regional Settings** option while WindowViewer is running.
2) Change the OS Regional Settings while WindowViewer is running with the **Regional Settings** option selected.

7. To restore the default values, select **Restore Default**

Set the regional locale of the computer hosting the HMI/SCADA application

To enable numeric formatting by regional locale, set the region of the computer running an HMI/SCADA application to the country in which you want Industrial graphic numbers to be formatted.

The **Region** setting is accessible from the Windows Control Panel. If you want to display Industrial graphic numbers in a non-U.S. format, select the **Formats** tab and select a country in the **Format** field.

Configure core affinity for WindowViewer in a terminal server environment

When running on a Terminal Services client, WindowViewer can use a CPU (core) other than CPU 0 for its execution, if the computer has multiple processors. This is so that InTouch applications that run in a Terminal Server environment can take advantage of the multi-core capabilities of the Terminal Server. When WindowViewer runs on a Terminal Server console, however, this option is not available. An InTouch application always runs on a single processor, regardless of the number of processors available.

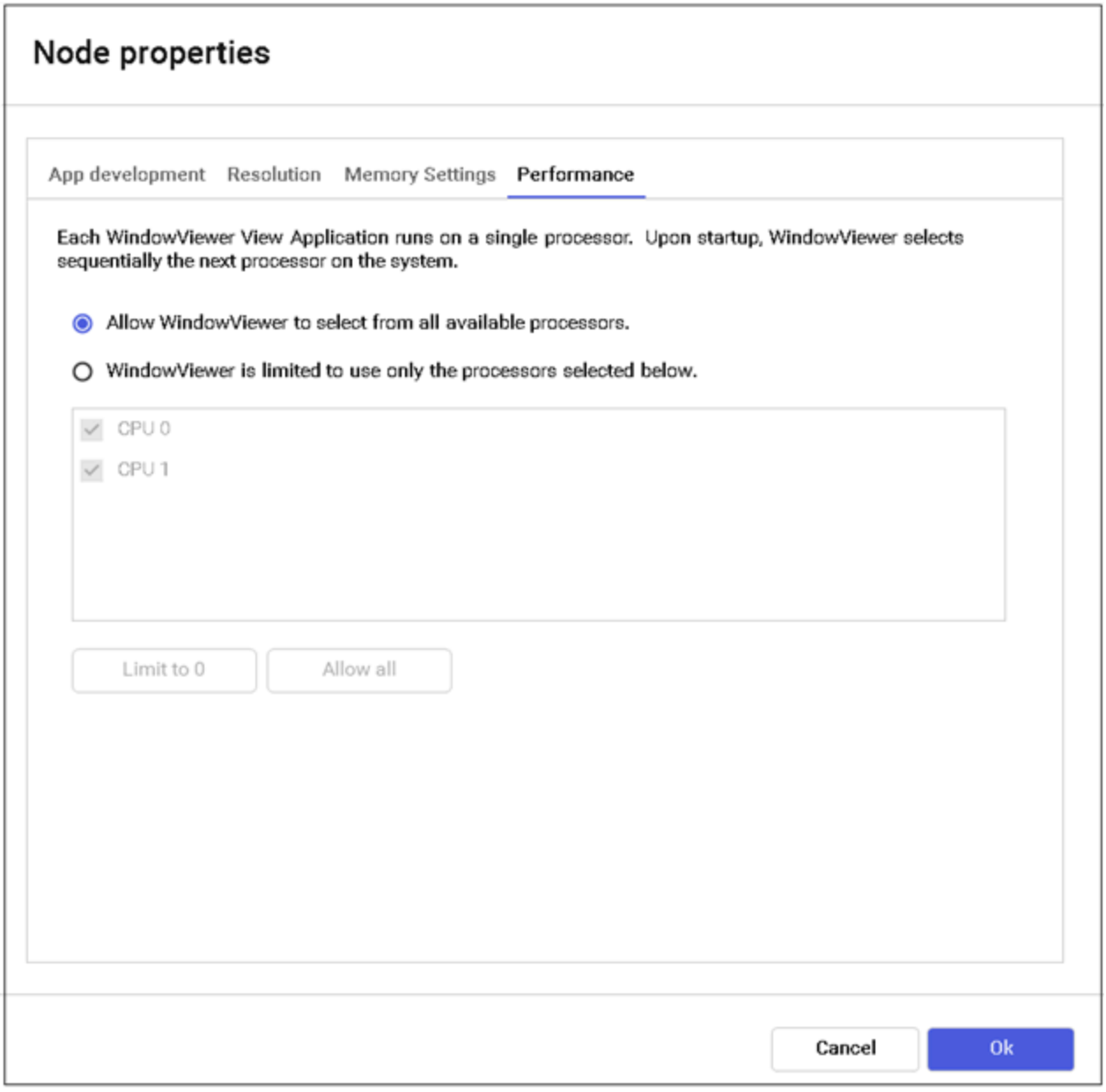
When WindowViewer starts, the system checks if the computer has multiple processors and which processors are allowed to run WindowViewer instances. WindowViewer then checks the processors sequentially and starts running on the processor that has the least number of View instances running on it.

If you have administrative privileges to access the **Performance** tab, you can configure the "pool" of processors from which WindowViewer selects the processor to run on.

You set the core affinity for WindowViewer within Application Manager. Avoid using Task Manager to manually adjust the core affinity for WindowViewer, as the WindowViewer core selection process does not take into consideration the core affinity settings configured in Task Manager.

Configure the processor "pool"

1. Start **Application Manager**.
2. On the **Tools** menu, select **Node Properties**. The **Node Properties** dialog box appears.
3. Select the **Performance** tab.



4. To allow WindowViewer to use any available processor, select **Allow WindowViewer to select from all available processors**.
5. To restrict the processors that WindowViewer can use, select **WindowViewer is limited to use only the processors selected below** and then do any of the following:
   - Make sure the **CPU** check box is selected for each processor you want WindowViewer to be able to run on.
   - Select **Limit to 0** to only allow WindowViewer to run on processor 0. When you select this button, the **CPU 0** check box is automatically selected.
   - Select **Allow All** to select all check boxes.

     You can clear a selected processor at any time and select a new processor from the list. You can also select multiple processors at a time. If you clear a processor check box, the WindowViewer instance continues to run on that processor.

6. Select **OK**. WindowViewer starts on the next CPU based on the other View sessions.

Work with WindowViewer windows

A typical InTouch application includes at least several windows that operators interact with to manage their industrial processes. Based on the properties you set from the **Window** tab on the WindowViewer configuration screen, operators can run standard commands from the WindowViewer **File** menu to open and close windows.

View WindowViewer in different modes

Select **Details** to change from the list view to the details view. The details show the window's type and the date and time when a window was last modified.

In the details view, you can select and deselect any unopened window by selecting on any portion of its row, not just the check box. The entire row is highlighted when selected.

- To open selected windows select **OK**.
- To cancel your selections and close the dialog box, select **Cancel**.
- To return the dialog box to list view, select **List**.
- To select all listed windows, select **Select All**.
- To clear all selected windows, select **Clear All**.
- To sort the list in ascending or descending order, select the column header.

Open windows from WindowViewer

If you configured WindowViewer to show the **File** menu, operators can open or close InTouch application windows. When operators select either the **Open Window** or **Close Window** command from the File menu, the respective dialog box for the selected command appears.

The names of all the windows that are applicable for the selected command appear in the list. For example, the **Windows to Show** screen appears after selecting on the **Open Window** command.

Operators can open InTouch application windows if WindowViewer is configured to show the **File** menu.

Open windows from WindowViewer

1. On the **File** menu, select **Open Window**.

   The **Windows to Show** dialog box appears.
2. Select the check box next to the name of each window that you want to open.
3. Select **OK** to close the dialog box and open windows you selected.

   Note: If a "Replace" type window is selected, it closes any windows that it intersects.

Close Windows from WindowViewer

Operators can close InTouch application windows if WindowViewer is configured to show the **File** menu.

Close open windows

1. On the **File** menu, select **Close Window**.

   The **Select windows to close** screen appears.
2. Select the check box next to the name of one or more windows that you want to close.
3. Select **OK** to close the windows you selected.

Transfer from WindowViewer to WindowMaker

When you develop an InTouch application, you can easily transfer between WindowMaker and WindowViewer by selecting the WindowMaker command in the **File** menu, or the **Development** command in the toolbar. This is called fast-switching.

Fast-switching is for rapid development testing only. Do not use it in a production environment. You can hide the command for switching to WindowMaker.

Transfer from WindowViewer to WindowMaker

1. Launch WindowViewer.
2. On the **File** menu, select **WindowMaker**.

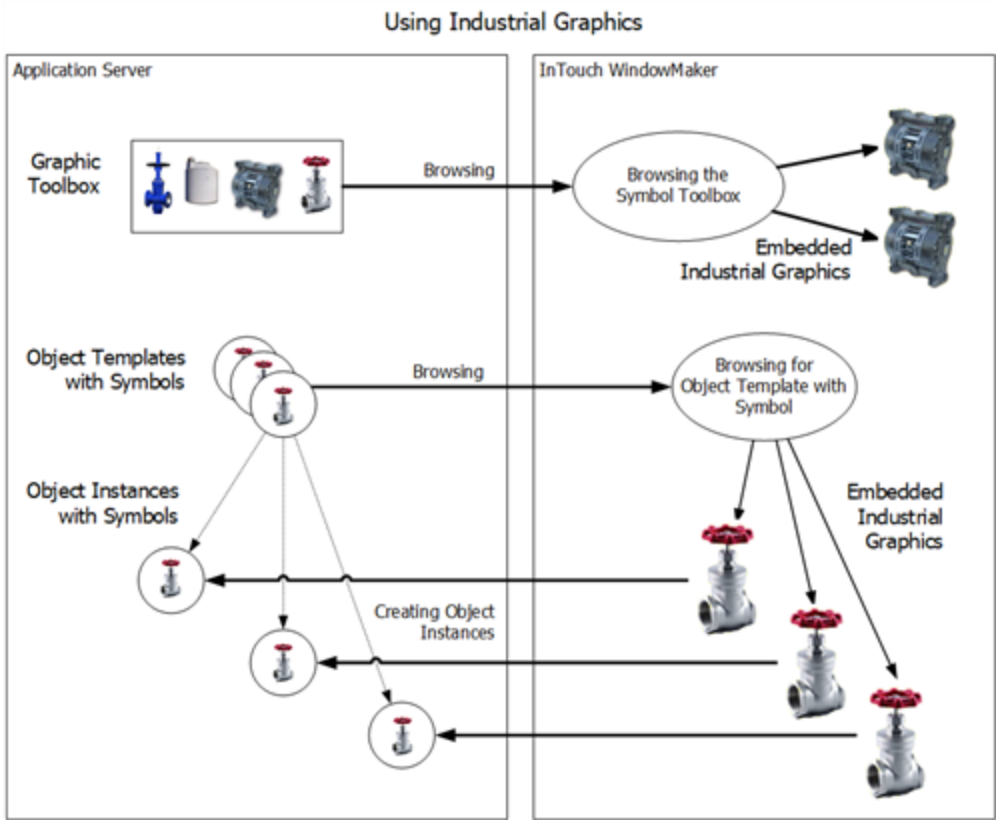
   The **Windows to Edit** dialog box appears.
3. Select the check box next to the name of each window that you want to open when you transfer to WindowMaker.
4. Select **OK** to close the dialog box and transfer to WindowMaker.

Note: If the application developer selected the **Close WindowViewer** option when WindowViewer's properties were configured during development, WindowViewer automatically closes when you transfer to WindowMaker.

Industrial Graphics

The System Platform IDE includes a Industrial Graphic Editor to create symbols that represent production processes and provide an HMI interface to AutomationObjects.

The following diagram shows how symbols that are created with the Industrial Graphic Editor can be used in InTouch applications.



Create Industrial Graphics

Industrial Graphics are created with the Industrial Graphic Editor in the IDE.

You can create:

- Industrial Graphics in the Industrial Graphic Toolbox. These are not associated with any specific object template or object instance.
- Industrial Graphics contained in a specific object template or instance.

Industrial Graphic Editor

In addition to the benefits of managing InTouch applications within the System Platform IDE, you can model your manufacturing environment by creating graphics with the Industrial Graphic Editor. The Industrial Graphic Editor is fully integrated into the System Platform IDE and supports powerful graphics editing capabilities.

The Industrial Graphic Editor includes the ability to apply an Element Style to a symbol element. An Element Style is a defined set of visual properties for text, element fill, element outlines, and lines that are applied to graphic elements. Using Element Styles makes it easy to apply standard visual styles to graphic elements and establish consistent symbol standards.

The Industrial Graphic Editor also includes the Symbol Wizard Editor, which enables you to create Symbol Wizards containing different visual and functional configurations. Using the Symbol Wizard Editor, you can, for example, create a configurable pump symbol with separate left and right inlet valve configurations. Situational Awareness Library symbols are examples of Symbol Wizards containing pre-built configurations that are selected

by choosing from the symbol's Wizard Options. A single Situational Awareness Library symbol can be easily configured to represent a thermometer, a flow meter, or a pressure meter. You select the configuration you need when the Symbol Wizard is embedded in a managed InTouch application.

WindowMaker incorporates the Industrial Graphic Toolbox to select Industrial and Situational Awareness Library symbols for InTouch applications without opening the Industrial Graphic Editor. Symbols can be added by dragging and dropping them directly from WindowMaker into an InTouch window. For more information about working with Industrial and Situational Awareness Library symbols, see Industrial Graphic Editor documentation or WindowMaker help.

Note: In recent versions of Microsoft's rendering technologies, certain gradient features have been deprecated. To accommodate and future proof graphics, the affected features have been removed from the configuration environment. Graphics previously configured with deprecated features will continue to render as expected. See "Load graphics with deprecated features" in Industrial Graphic Editor User documentation.

Applications

You can manage your InTouch applications with the InTouch Application Manager or with the IDE.

InTouch applications are categorized by how they are managed, the types of symbols they support, and where they were published from:

- Standalone InTouch applications: A standalone application is the default application created in Application Manager that allows the flexibility of InTouch symbols and Industrial Graphics.
- Managed InTouch applications: A managed application is created and managed with the IDE.
- Published InTouch applications: A published application is created by exporting a managed application from the derived InTouchViewApp template.
- InTouchView application: A InTouchView application can be created either from InTouch Application Manager or the IDE.

InTouch HMI supports the migration and upgrading of older Modern applications.

Standalone InTouch applications

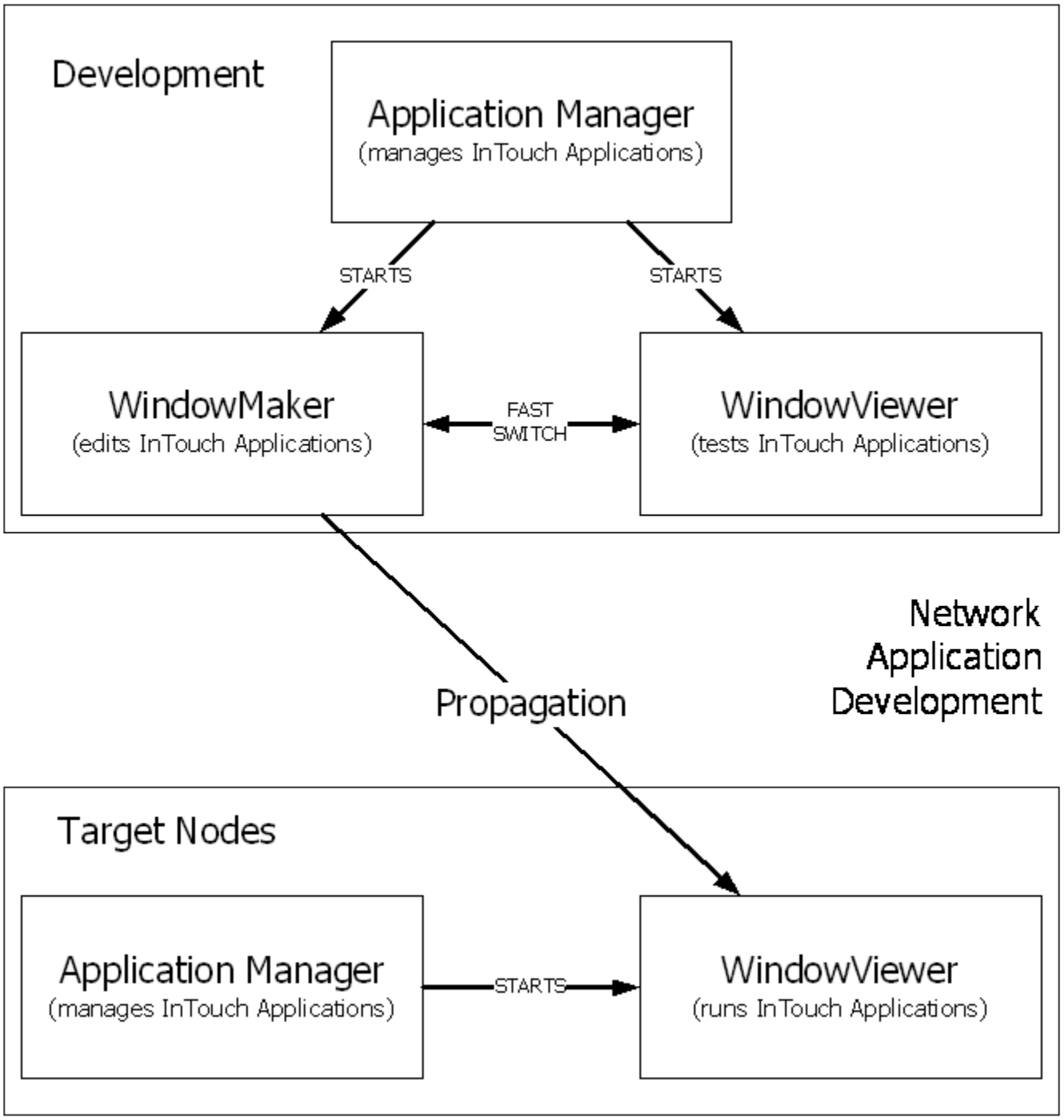
Standalone InTouch applications are managed by the InTouch Application Manager. They appear in the InTouch Application Manager as **Standalone**.

Using the Application Manager, you can:

- Create and manage standalone InTouch applications.
- Start WindowMaker to edit InTouch applications
- Start WindowViewer to run InTouch applications.

You can also switch directly between WindowMaker and WindowViewer to test or run your applications and switch back to make modifications to your applications. A standalone application consists of a set of files maintained by the InTouch HMI in the directory file system. It can also contain Industrial Graphics. A standalone application can be deployed across multiple network nodes and is not restricted to a single node. It can be imported to the IDE and converted to a managed application.

Network Application Development manages the propagation of changes from your InTouch application on a development node to running InTouch applications on target nodes.



Managed InTouch applications

You can manage your InTouch applications using the System Platform Integrated Development Environment (IDE). These applications are called Managed InTouch applications.

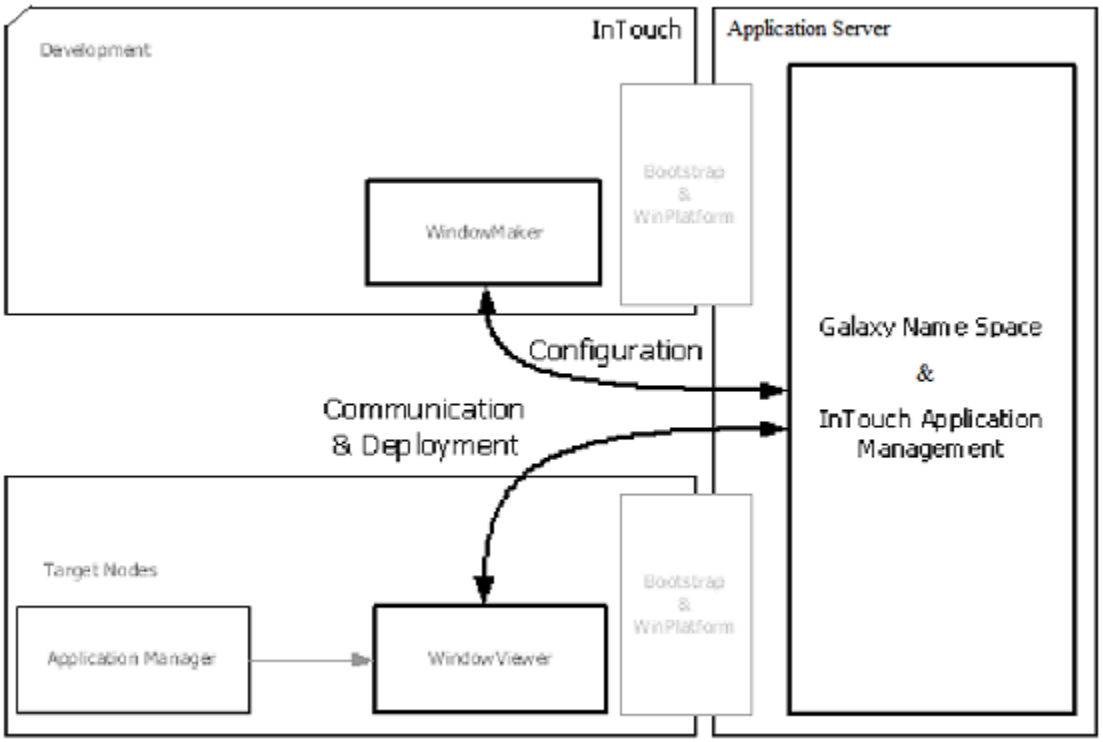
They appear in InTouch Application Manager as **Managed**. They are more integrated into the ArchestrA

environment and support advanced graphics.

You develop an InTouch application on one node of the Galaxy with WindowMaker. You then deploy it to one or more target nodes running WindowViewer.

When you use the system platform functionality of the IDE to manage your InTouch applications, you can:

- See which InTouch applications are running on what node.
- Use a central repository for InTouch applications.
- Deploy changes to WindowViewer running on the remote nodes.



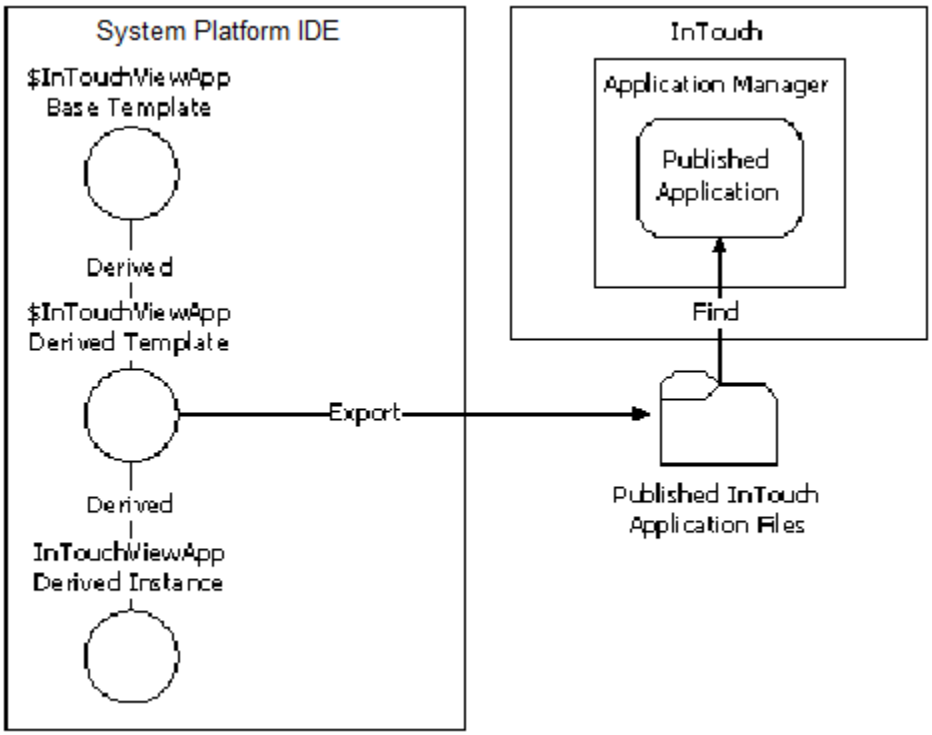
You can only start a managed application in WindowViewer, from the Application Manager. When WindowViewer starts, it copies a managed application's files to a folder during run time.

Each managed application is associated with an InTouchViewApp object, which is derived from a base template. The InTouchViewApp object only contains a reference to the managed InTouch application folder and other behavior-specific information of the managed InTouch application. Application files are stored in separate folders in the Galaxy file repository. One folder contains the most recent checked-in version of the InTouch application and the other contains the most recent checked-out version. The IDE includes the Industrial Graphic Editor, which you can use to create symbols that represent production processes in your InTouch HMI application. You can fast switch from WindowMaker and WindowViewer to test a managed application only if the WindowMaker was opened from the IDE.

Published InTouch applications

After you edit your managed InTouch application, you can publish it. You can publish a managed application from the derived InTouchViewApp template. When you publish a managed application, a user-defined folder is created containing InTouch application files and any Industrial graphics embedded in the application. You use

Application Manager's **Find** utility to locate the folder. Thereafter, the converted application appears in Application Manager as a published application.



The advantage of published InTouch applications is that they can be distributed like standalone InTouch applications, but continue to support the functionality of Industrial graphics.

However, you can no longer:

- Use System Platform IDE to deploy InTouch applications.
- Edit or add Industrial Graphics in InTouch applications.
- Edit the published InTouch application.

You can use WindowMaker to migrate a published application from a version of InTouch prior to version 11.1 (2014 R2 Patch 01) to version 11.1 Patch 01. You can then modify the published application by updating the application source files and re-publishing. After you publish a managed application, you can still use embedded Industrial graphics to write data to a Galaxy or visualize data. A published application cannot be imported again into a Galaxy. After migrating a published Managed application, you need to republish the application. The Embedded Alarm Control will be upgraded to the new version upon republishing.

InTouchView applications

InTouchView applications show visual interfaces for use in an Application Server environment. InTouchView applications run in WindowViewer, with Application Server providing most of the HMI functionality. An application can also be configured to serve as InTouch Tag Server application, providing other nodes with secure tag information. InTouchView applications are useful in scenarios where a client node only needs to access data and does not need the full functionality of a development node.

InTouchView applications offer only some of the standard functions available from full-featured InTouch applications. InTouchView applications:

- Cannot connect to I/O sources other than the Application Server Galaxy or InTouch Tag Server.
- Cannot generate alarms. However, you can display and acknowledge alarms from remote alarm providers, such as Application Server objects and InTouch alarms.
- Do not log application data or events. An InTouchView application generates only SYS and USER-related events.
- When the application consumes data from a Galaxy, it can be secured only with ArchestrA security.
- Cannot reference tags within embedded Industrial Graphics.

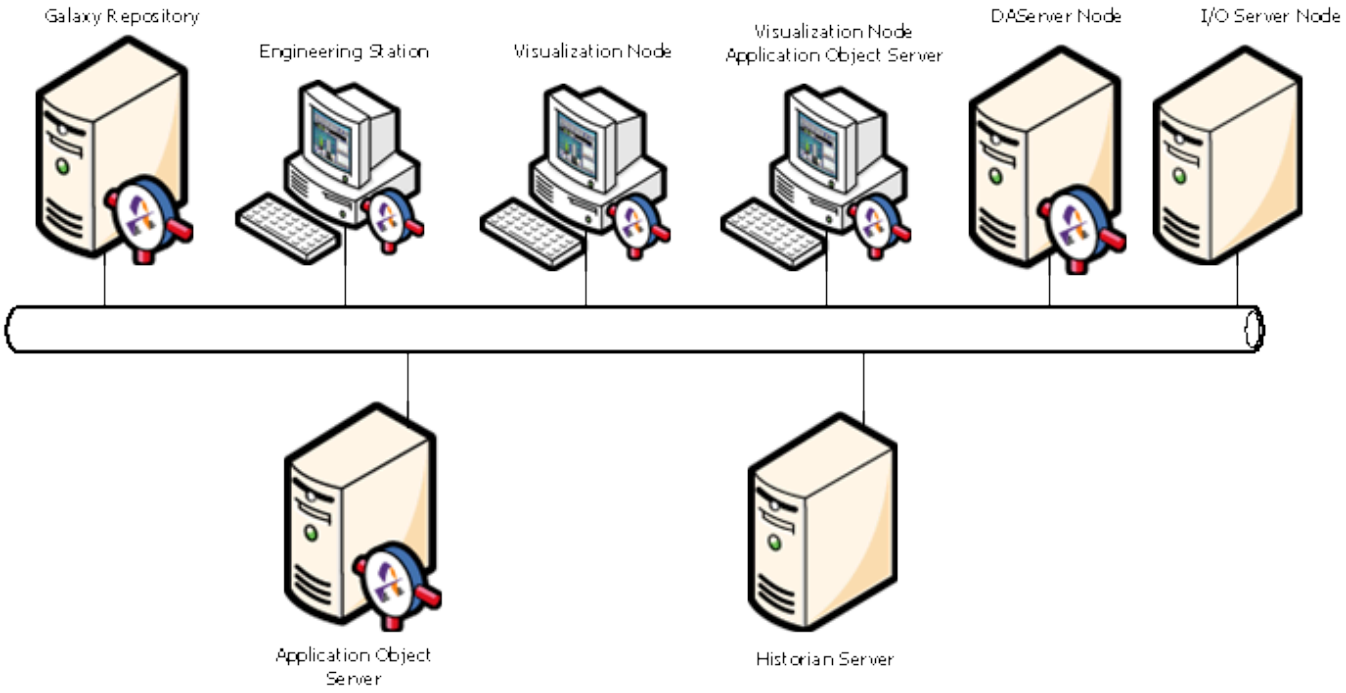
You develop InTouch applications with WindowMaker, and run them in WindowViewer. You can then change an InTouch application to an InTouchView application that will allow you to manage your InTouch applications through the Application Server. Likewise, you can change an InTouchView application to an InTouch application.

The following lists show which WindowMaker commands and Tagname Dictionary options are unavailable when creating InTouchView applications.

- Unavailable commands:
  - Access Names
  - Alarm Groups
  - Configure...Alarms
  - Configure...Historical Logging
  - Configure...Distributed Name Manager
- Unavailable Tagname Dictionary options:
  - Alarms
  - Details & Alarms
  - Log Data
  - Log Events
  - Priority

Application Server architecture

Application Server relies on ArchestrA technology to provide services for distributed InTouch HMI applications. Application Server services are distributed across a set of nodes with the InTouch HMI installed on the Visualization Node to provide the application interface.



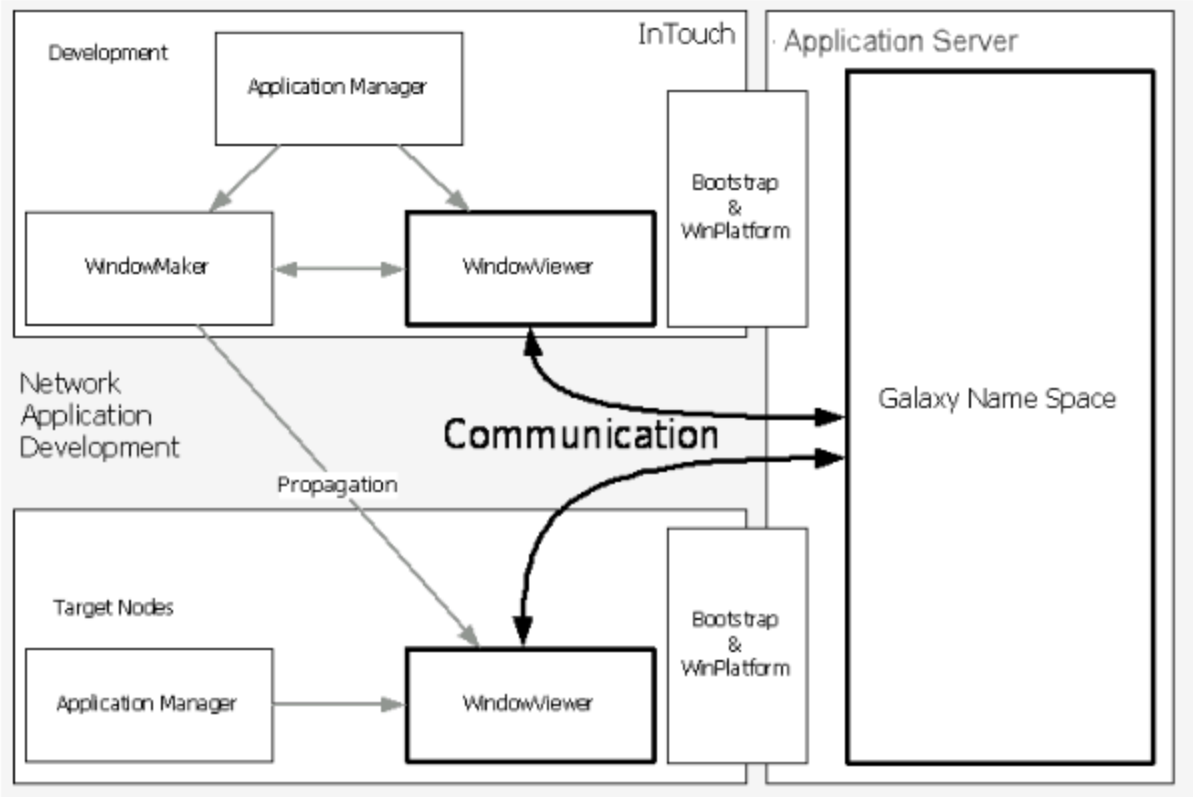
An Application Server Galaxy represents the system's configuration based on a single logical namespace that incorporates a variety of components that are distributed across multiple nodes.

Application Server can store production data similar to InTouch tags, but it supports more data types and arrays. It can provide alarm capabilities that are compatible with the InTouch alarm system. Application Server provides an InTouch HMI compatible scripting language with significant enhancements for .NET function support. To be able to read and write Galaxy data, you need to install the Application Server Bootstrap component on the InTouch node. To be able to browse the Galaxy, you need to install the Application Server IDE component.

For more information, see the Application Server documentation.

Communication within the Galaxy

ArchestrA enables the use of a Galaxy-wide name space to contain and process production-related data. It also allows a high-level visualization and administration of data access from various nodes running InTouch in the manufacturing environment.



Compare standalone, managed, and published InTouch applications

Standalone, managed, and published InTouch applications have some differences and some similarities, as described in the following table:

	Standalone InTouch Applications	**Managed InTouch Applications**	**Published InTouch Applications**
Create Application	Application Manager	System Platform IDE • New application • Importing standalone application • Importing SmartSymbols	Not possible
Edit Application	WindowMaker started from the Application Manager	WindowMaker started from within the IDE	Not possible

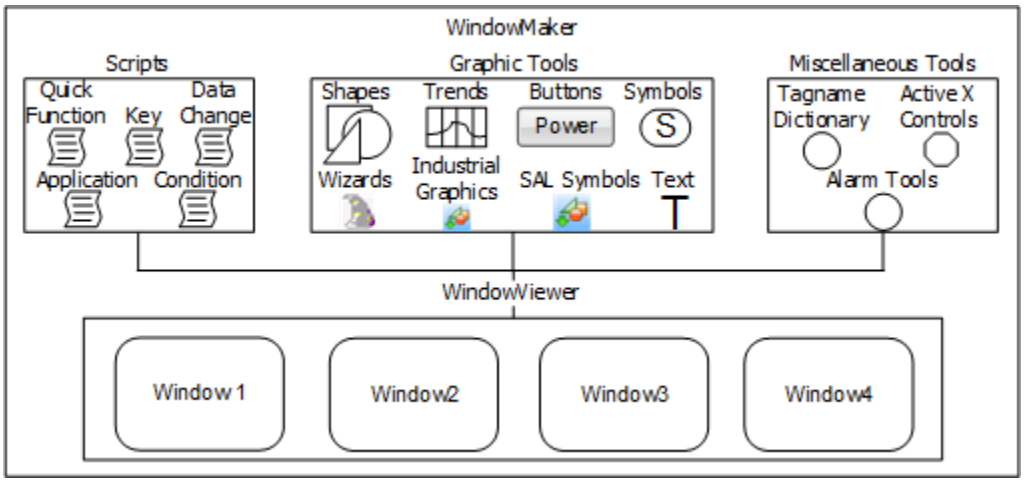
	Standalone InTouch Applications	Managed InTouch Applications	Published InTouch Applications
Delete Application	Remove from Application Manager.	Delete InTouchViewApp template	Remove from Application Manager.
Support of Industrial graphics, including Symbol Wizards	Yes	Yes for all operations	Yes, but only for viewing, not for creating and editing
Support of DB Dump and DB Load	Yes, function within Application Manager	Yes, function within the IDE	Yes, function within Application Manager
Editing application in original resolution requires conversion	Yes	No	Yes
Management of Distributed Applications	Network Application Development (NAD)	System Platform IDE	Network Application Development (NAD)
Configuring how new InTouch application versions are accepted	Configured in Application Manager (Network Application Development)	Configured in WindowMaker	Configured in Application Manager (Network Application Development)
Use Fast-Switch to test application	Yes	Yes	Yes
Use tag value and tag parameter retention	Yes	Yes, also requires configuration of the local working directory	Yes

Build applications

WindowMaker provides graphic tools, a scripting language, and tag management utilities to define the behavior of objects that appear in your application's windows. Using WindowMaker, you can create tags that represent

data points associated with window objects. Data from a manufacturing process is ultimately associated as a tag value. This tag data can be used in your application for alarm monitoring, creating trends, and determining how the application behaves during run time.

The following figure shows some of WindowMaker's tools to create InTouch HMI applications.



You can use a wide variety of graphic tools that range from simple shapes that can be combined to create more complex objects to standard Industrial Graphics with predefined properties. You can create different types of scripts based upon their triggering mechanism. You can also insert predefined InTouch functions into your scripts. You can define a variety of tag value thresholds with the Tagname Dictionary that determines when a tag is in a normal or an alarmed state.

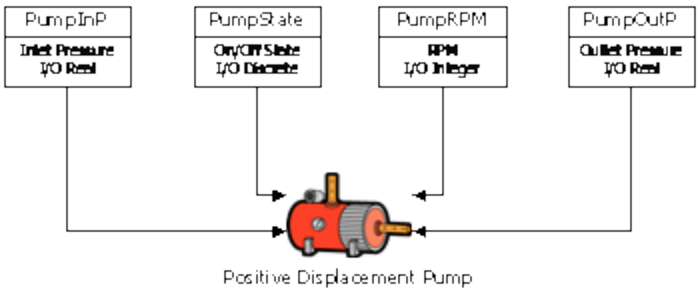
Run applications

You use WindowViewer to run all types of InTouch applications. After you deploy a managed application from the System Platform IDE, you open it in WindowViewer from Application Manager. You can use a wide variety of run-time triggers to start scripts while an application is running. You can configure WindowViewer to store application data and alarms in files or SQL Server databases. You can enforce security by requiring operators to log on to WindowViewer and preventing operators from making any changes to the computer running WindowViewer. Operators can start and stop an application's historical logging by selecting WindowViewer menu commands. You can configure the computer that runs WindowViewer to act as a client or a server based upon how tag data is stored and distributed.

Tags

An InTouch Human Machine Interface (HMI) application is a graphical representation of the components in a manufacturing environment. Plant operators work with this graphical interface to monitor and administer their manufacturing processes.

The figure below shows an example of a pump that is a component of a manufacturing process. The pump has properties with associated values. Pressure, RPM, and status are pump properties whose values are monitored through an HMI.

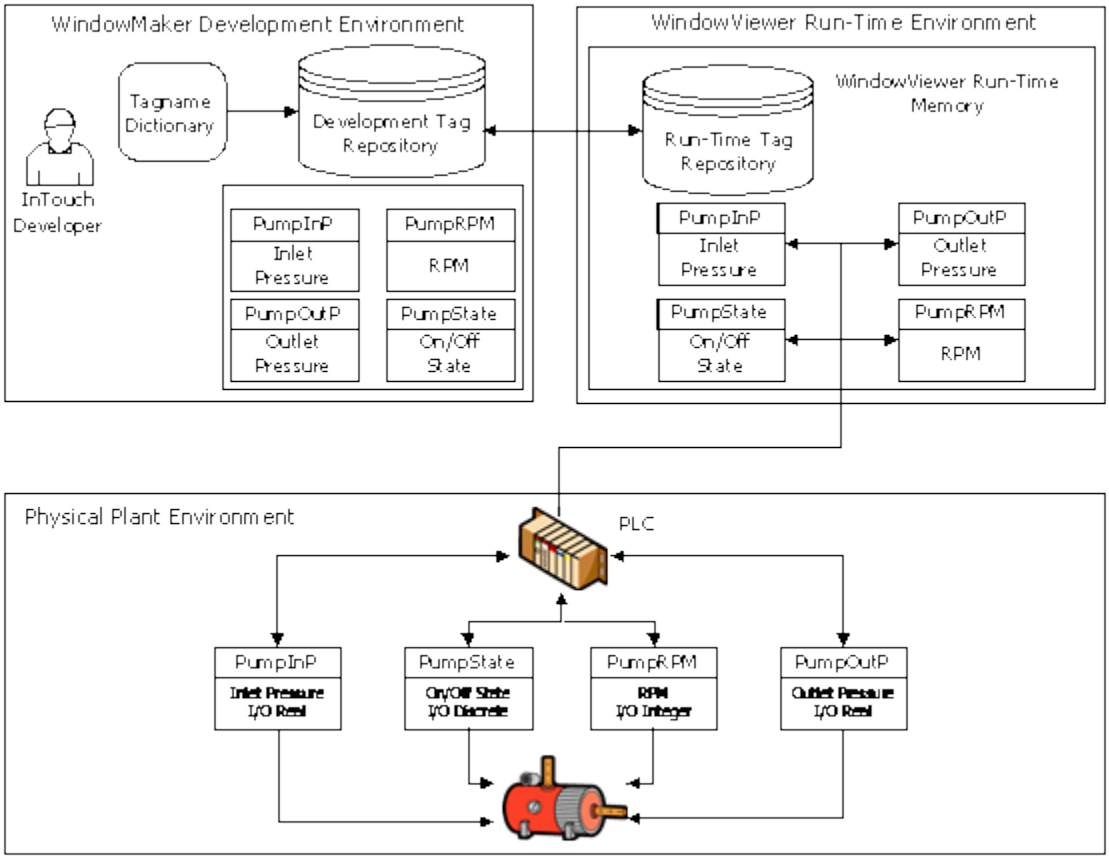


A *tag* represents a data item in an InTouch HMI application. You use tags to make specific component properties accessible as data items from a manufacturing environment. In the figure above, the PumpState tag indicates whether the pump is on or off. You create tags for components in your manufacturing environment whose properties you want to monitor or control in your InTouch application.

You can use different types of tags for the different types of data collected from a manufacturing component. For example, the PumpState tag returns a Boolean On/Off value to indicate if the pump is running or stopped. You assign the appropriate type of InTouch tag for the type of data that you want to be part of your application.

Work with InTouch tags

You start by creating an InTouch application. You define tags for the application using the Tagname Dictionary, which is a WindowMaker tool. The figure below shows the InTouch development and run-time environments.



You assign the name and type of tag with the Tagname Dictionary. For some types of tags, you have other options in the Tagname Dictionary to specify additional properties of tags. For example, I/O type tags include

additional options to specify the connection to a remote data source.

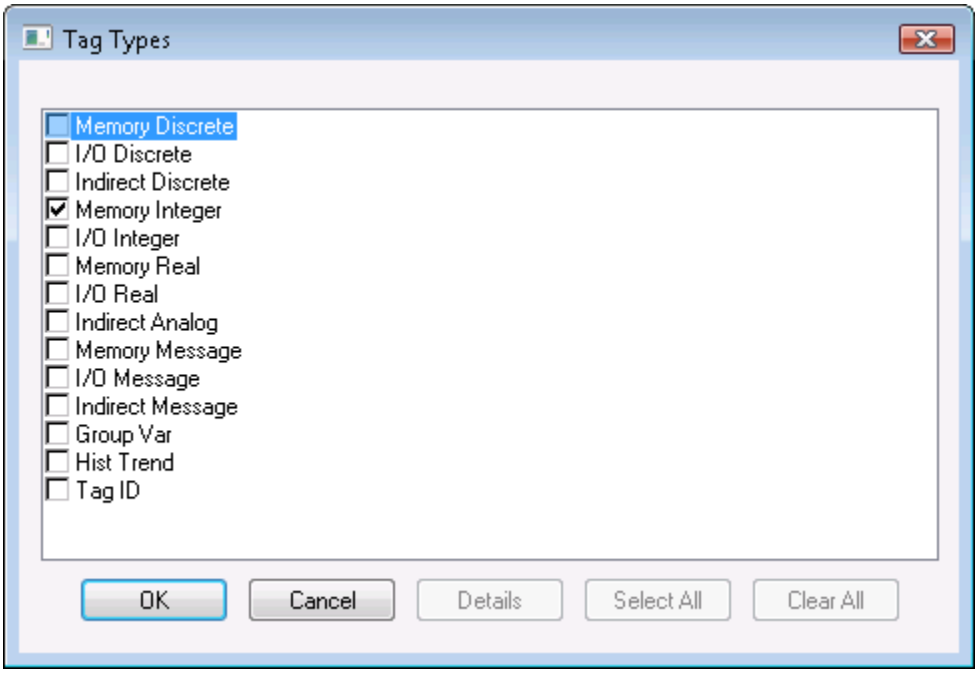
InTouch applications run in the WindowViewer environment. When WindowViewer starts an application, it reads the tags from the development repository and places them into run-time memory.

The InTouch application communicates with the tags placed into run-time memory using animation links or scripts. The InTouch application tracks the current values and other status information from the component properties assigned to tags.

Types of InTouch tags

When you define a tag, you assign to it a specific type according to the process data that will be associated with the tag. For example, if a tag shows the RPM of a pump, then you assign the tag as an integer tag type.

In the Tagname Dictionary, you use the **Tag Types** dialog box to assign the tag type to any tag you created.



After you assign a tag type, the Tagname Dictionary lists specific options for the type of tag you selected.

Memory tags

Memory tags define internal system constants and variables within InTouch applications. For example, you can define an internal constant as a real number of 3.414. In process simulations, memory tags can control the actions of background QuickScripts by acting as a counter. Based upon the count associated with the tag, the QuickScript can trigger various animation effects. Memory tags can also act as calculated variables that are accessed by other programs.

Select from the four types of memory tags, based upon the process data associated with the tag.

- **Memory Discrete**

  Memory discrete tags are associated with the state properties of a process component. The values assigned to memory discrete tags are two possible Boolean states such as:

- 0 or 1
- False or true
- On or off
- High or low

- **Memory Integer (Analog)**

  You can assign memory integer tags 32-bit signed-integers between -2,147,483,648 and 2,147,483,647.

- **Memory Real (Analog)**

  You can assign memory real tags floating decimal point numbers between -3.4 x 1038 and 3.4 x 1038. When two real tags values are compared, the difference of two real tag values should be greater than FLT_EPSILON (value 1.19209290E-07F, in decimal 0.0000001192092896). All floating point calculations are performed with 64-bit resolution, but the results are stored as 32-bit decimal numbers. For more information about the maximum precision of real numbers, see IEEE decimal units.

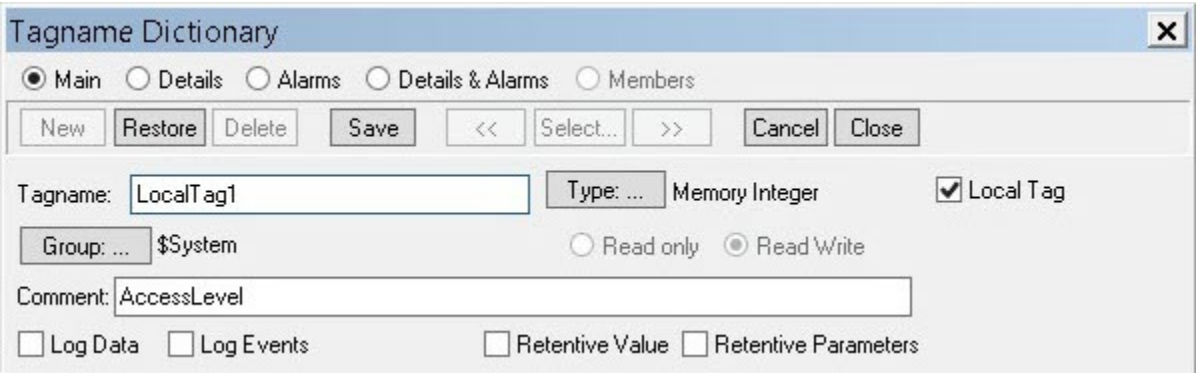
- **Memory Message**

  You can assign memory message tags text strings up to 131 single-byte characters in length.

Local tags

Local tags allow the user to create per-session memory tags for use in the Web Client. For example, the user can configure a local memory tag to be used for navigation. In runtime when the user selects the navigation symbol linked to that tag, each session will behave independently.

Note: .Dot fields are not supported for local tags.



Create a local tag

1. Create a memory tag and select the **Local Tag** checkbox. Configure other tag properties. For more information on creating new tags, see Create new tags.
2. Select **Save**.

I/O tags

I/O tags read or write InTouch application data to or from an external source. External data includes input and output from programmable controllers, process computers, and network nodes. I/O tag data values are remotely accessed through the following protocols:

- Microsoft Dynamic Data Exchange (DDE)

- SuiteLink™

When the value of an I/O tag changes in run-time memory, the InTouch HMI updates the remote application. Conversely, I/O tag values in InTouch are updated whenever the values of corresponding data items change in a remote application.

The InTouch HMI provides four types of I/O tags based upon the process data associated with the tag. These four types of I/O tags are similar to memory tag types.

- **I/O Discrete**

  I/O discrete tags are associated with component process properties whose values are represented by two possible states such as:

  - 0 or 1
  - False or true
  - On or off
  - High or low
- **I/O Integer (**Analog)

  I/O integer tags can be assigned 32-bit signed-integers between -2,147,483,648 and 2,147,483,647.
- **I/O Real** (Analog)

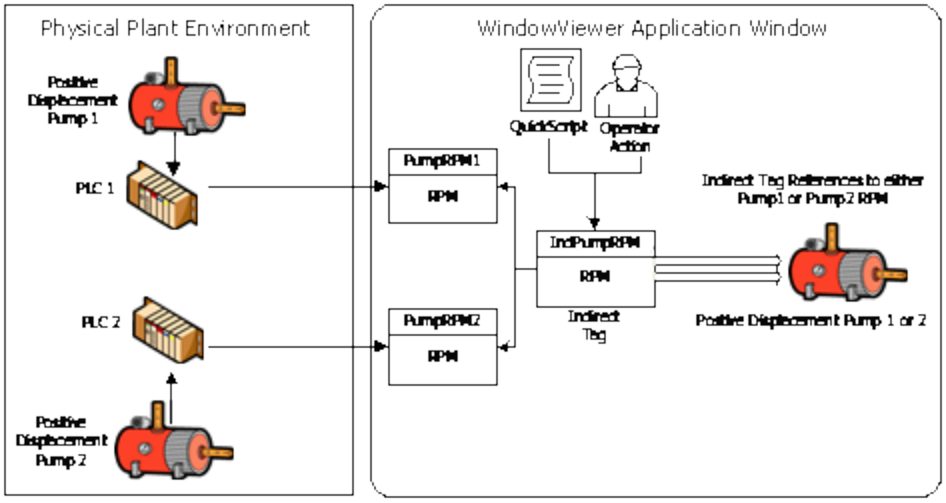
  I/O real tags can be assigned floating decimal point numbers between -3.4 x 1038 and 3.4 x 1038. When two real tags values are compared, the difference of two real tag values should be greater than FLT_EPSILON (value 1.19209290E-07F, in decimal 0.0000001192092896). All I/O real tag floating point calculations are performed with 64-bit resolution, but the results are stored as 32-bit numbers. For more information about the maximum precision of I/O real numbers, see IEEE decimal units.
- **I/O Message**

  I/O message tags can be assigned text strings up to a maximum of 131 single-byte characters.

Indirect tags

Indirect tags act as "pointers" to other tags. For example, you can create a single InTouch window and use indirect tags to show data from multiple different sets of tags.

The following figure shows an example of an application window that is capable of displaying several pumps. Instead of creating separate windows for each pump, you can use indirect tags in one window to show the values of different source tags associated with individual pumps.



A QuickScript or operator action points the indirect tag to the source tags. For example, the following script statements assign the two PumpRPM tags to an indirect analog tag called **IndPumpRPM** based on the value of the PumpNo tag.

```
IF PumpNo == 1 THEN
   IndPumpRPM.Name = "PumpRPM1";
   ELSE
   IndPumpRPM.Name = "PumpRPM2";
ENDIF;
```

When you equate an indirect tag to another source tag, the indirect tag acts as if it is the source tag. The values associated with the original source and indirect duplicate tags are synchronized together. If the value of the source tag changes, the indirect tag reflects the change. If the indirect tag's value changes, the source tag changes accordingly.

You can use discrete, analog, and message types of indirect tags. These three types of indirect tags are comparable to similar memory and I/O types of tags.

For more information about indirect tags, see Define indirect tags.

Miscellaneous tags

You can use other types of InTouch tags designed for specific, restricted purposes. You can use these tags to create dynamic alarm displays, create historical trends, and change the tags assigned to historical trend pens.

Hist trend tags

Hist Trend tags can be used to reference historical trend graphs. All of the **dotfields** associated with historical trends can be applied to **Hist Trend** tags.

For more information about defining and using Hist Trend tags, see Trending tag data.

Tag ID tags

Tag ID tags retrieve information from tags whose values are plotted in an InTouch historical trend graph. Typically, you use a Tag ID tag to show the name of the tag assigned to a specific trend pen or change the tag assigned to a trend pen.

You can include a statement in a QuickScript to assign a new tag to a pen in any historical trend using a Tag ID type tag. For example, the following QuickScript statement changes the tag associated with a historical trend pen:

```
HistTrend.Pen1=MyLoggedTag.TagID;
```

When this QuickScript runs, Pen1 of the historical trend begins trending the historically logged data for the MyLoggedTag.

For more information about defining and using Hist Trend tags, see Use the historical trend wizard.

Supertags

A Supertag is a template that contains a set of related tags. For example, you can create a Supertag template containing a set of tags assigned to all the properties of a pump.

Use Supertags when you have identical equipment in your production process. Instead of creating a set of tags for each piece of equipment, assign an instance of the Supertag template to each of the identical process items.

For more information about Supertags, see Define reusable tag structures.

Obsolete tags

Using a Group Var tag, you can create dynamic alarm displays, dynamic logging on disk, and dynamic printing with the standard alarm system of InTouch. Group Var tags are included only for backward compatibility with applications developed with InTouch version 7.11 and earlier. Do not use Group Var tags in applications developed with InTouch versions later than 7.11.

System tags

Using system tags provides system-related information and standard functions like date and time for InTouch scripting. System tags are part of all applications. You can also use system tags in scripts to manage a running application like:

- Monitoring and managing application security.
- Establishing I/O communications to a remote node.
- Detecting when an alarm occurs.
- Starting and stopping historical logging.
- Updating an application to a new version with NAD.

System tags are identified by a dollar sign ($) as the first character of a tag's name in the Tagname Dictionary. System tags cannot be deleted. You can only change the comment associated with the system tag.

System tags reference

The InTouch system tags are described in the following table:

System Tag	Purpose	For More Information
$AccessLevel	Read-only integer tag that specifies the access level associated with the currently logged-on operator. This information can be used in animation links or scripts to control the operator's access to specific InTouch functions.	See Securing InTouch
$ApplicationChanged	Read-only discrete tag that indicates whether the master application has changed in a NAD environment.	See Distributing Applications
$ApplicationVersion	Read-only real tag that specifies the current version number of the application running in WindowViewer.	See Distributing Applications
$ChangePassword	Discrete write-only tag that shows the **Change Password** dialog box when set to 1.	See Securing InTouch
$ConfigureUsers	Write-only discrete tag that shows the generic **Configure Users** dialog box to edit the security user name list.	See Securing InTouch
$Date	Read-only integer tag that shows the whole number of days that have passed since January 1, 1970.	See Built-In functions
$DateString	Read-only message tag that shows the date in the same format specified from the Windows **Regional and Language Options** dialog box.	See Built-In functions

System Tag	Purpose	For More Information
$DateTime	Read-only real tag that shows the fractional number of days that have passed since January 1, 1970.	See Built-In functions
$Day	Read-only integer tag that shows the current day of the month (1-31).	See Built-In functions
$False	Discrete read-only tag that returns a FALSE value within an expression. The $False system tag is used to replace any instance of obsolete system tags when updating applications from earlier versions of InTouch to the current version.	No further information.
$HistoricalLogging	Read/write discrete tag used to start and stop historical logging while an InTouch application is running.	See Record tag values.
$Hour	Read-only integer tag that shows the current hour as a value from 0 to 23.	See Built-In functions
$InactivityTimeout	Read-only discrete tag that indicates the user inactivity period has elapsed. When set to 1, the inactivity period has elapsed and the user is automatically logged off from WindowViewer.	See Securing InTouch
$InactivityWarning	Read-only discrete tag that indicates the inactivity warning period has elapsed. The value of $InactivityWarning can be used to issue an inactivity warning to an operator.	See Securing InTouch
$LogicRunning	Read/write discrete tag that can be used to start and stop an application script.	See Securing InTouch

System Tag	Purpose	For More Information
$Minute	Read-only integer tag that shows the current minute (0-59).	See Built-In functions
$Month	Read-only integer tag that shows the number of the current month (1-12).	See Built-In functions
$Msec	Read-only integer tag that shows the current millisecond (0-999).	See Built-In functions
$NewAlarm	Read/write discrete tag that indicates when a new local alarm has occurred.	See Control alarm properties of tags and groups at runtime
$ObjHor	Read-only integer tag that shows the horizontal pixel location of the center of a selected object on the screen.	See Animate objects in AVEVA™ InTouch HMI Application Development
$ObjVer	Read-only integer tag that shows the vertical pixel location of the center of a selected object on the screen.	See Animate objects in AVEVA™ InTouch HMI Application Development
$Operator	Read-only message tag that shows the name of the operator logged on to an InTouch application.	See Securing InTouch
$OperatorDomain	Read-only message tag that contains the domain or machine name specified at log on when the application is secured with operating system-based security.	See Securing InTouch

System Tag	Purpose	For More Information
$OperatorDomainEntered	Write-only message tag assigned the domain of the operator for a logon attempt to an InTouch application. The logon attempt does not start until you assign a value to the $PasswordEntered system tag.	See Securing InTouch
$OperatorEntered	Read/write message tag assigned the user account name of an operator for a logon attempt to an InTouch application. The logon attempt does not start until you assign a value to the $PasswordEntered system tag.	See Securing InTouch
$OperatorName	Read-only message tag that shows the full name of the operator if operating system-based or ArchestrA® authentication is used.	See Securing InTouch
$PasswordEntered	Write-only message tag assigned the password of an operator for a logon attempt to an InTouch application. When you write a value to this tag, a logon attempt is started using the values of the $OperatorDomainEntered, $OperatorEntered, and $PasswordEntered system tags.	See Securing InTouch
$Second	Read-only integer tag that shows the current second (0-59).	See Built-In functions
$StartDdeConversations	Read/write discrete tag used to start uninitiated conversations during run time.	See Data access with I/O.

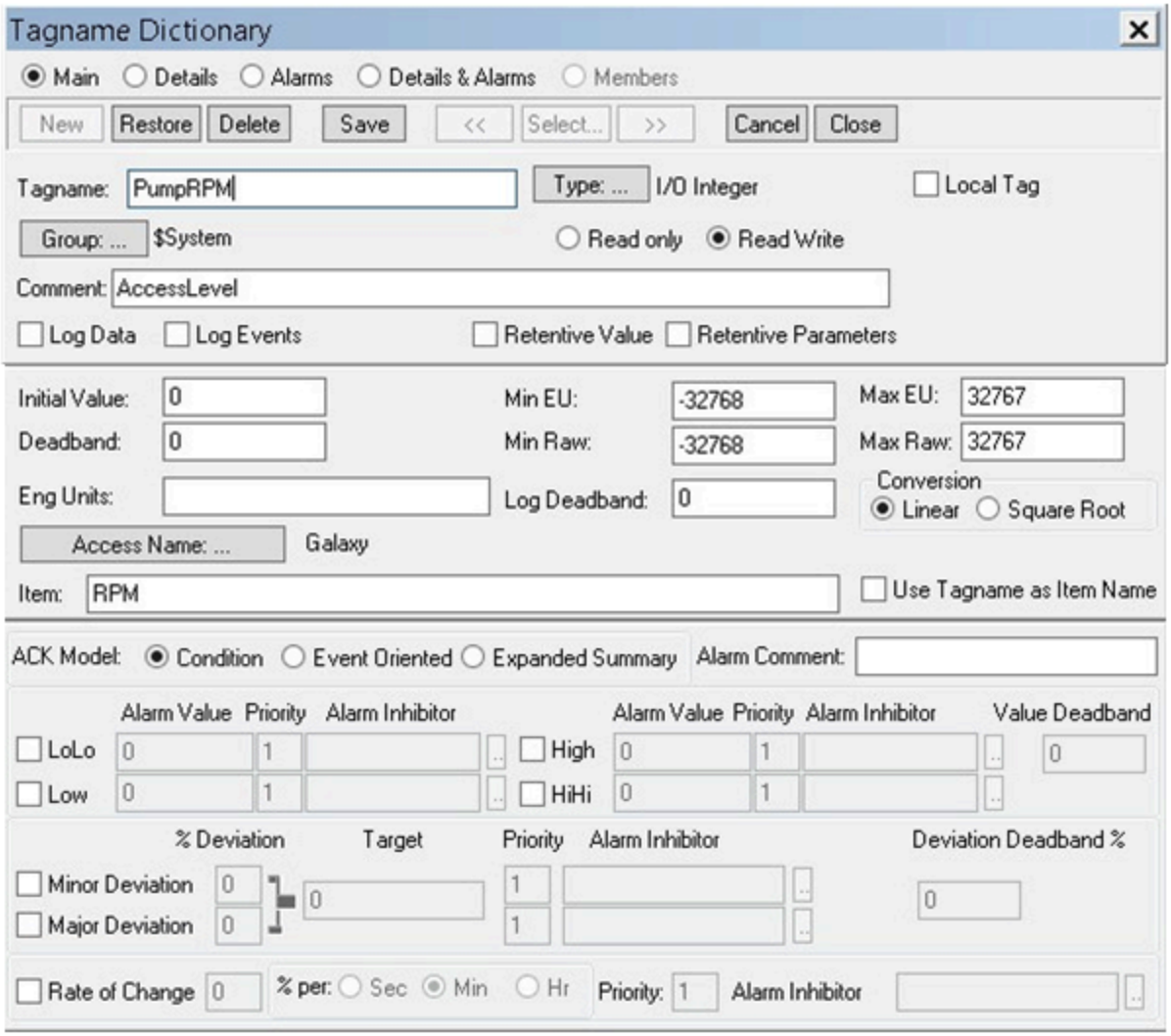
System Tag	Purpose	For More Information
$System	Read-only tag that identifies the root alarm group.	See *About Alarms and Events*
$Time	Read-only integer tag that shows the elapsed time in milliseconds since midnight of the current day.	See Built-In functions
$TimeString	Read-only message tag that shows the current time in the same format specified from the Windows **Regional and Language Options** dialog box.	See Built-In functions
$VerifiedUserName	Read-only message tag that contains either a verified user's full name or null.	See Securing InTouch
$Year	Read-only integer tag that shows the current year as a four-digit number.	See Built-In functions

Tag properties

Each InTouch tag type has a set of properties that describe the characteristics of data associated with the tag. The four principal data types associated with InTouch tags are:

- discrete values
- integer numbers
- real numbers
- text messages

All tag properties are assigned initial values when you create a tag with the Tagname Dictionary. The following figure shows the properties of an I/O Integer tag.



After setting the initial values of tag properties from the Tagname Dictionary, you can dynamically change most tag properties using dotfields. A dotfield identifies a tag property that can be monitored or modified by a script when the InTouch application is running. You append the dotfield to the name of the tag in a script.

For more information about using dotfields to dynamically change tag properties, see Use tag dotfields to view or change tag properties.

Memory tag properties

The following table lists the properties of the four types of memory tags. Each property can be selected or modified as an option of the Tagname Dictionary. For more information, see Create new tags.

Tag Properties	Discrete	Integer	Message	Real
% Deviation		•		•
% per		•		•
ACK Model	•	•		•

Tag Properties	Discrete	Integer	Message	Real
Alarm Comment	•	•	•	•
Alarm Group	•	•	•	•
Alarm Inhibitor	•	•		•
Alarm State	•			
Alarm Value		•		•
Comment	•	•	•	•
Deadband		•		•
Eng Units		•		•
Deviation Deadband %		•		•
High		•		•
HiHi		•		•
Initial Value	•	•	•	•
Log Data	•	•		•
Log Deadband		•		•
Log Events	•	•	•	•
Lo		•		•
LoLo		•		•
Maximum Length			•	
Major Deviation		•		•
Max Value		•		•
Min Value		•		•
Minor Deviation		•		•
Off Msg	•			
On Msg	•			
Priority	•	•	•	•

Tag Properties	Discrete	Integer	Message	Real
Rate of Change		•		•
Read Only	•	•	•	•
Read Write	•	•	•	•
Retentive Parameters		•		•
Retentive Value	•	•	•	•
Target		•		•
Value Deadband		•		•

I/O Tag properties

Like memory tags, I/O tag properties can be selected or modified as options in the Tagname Dictionary. For more information, see Create new tags.

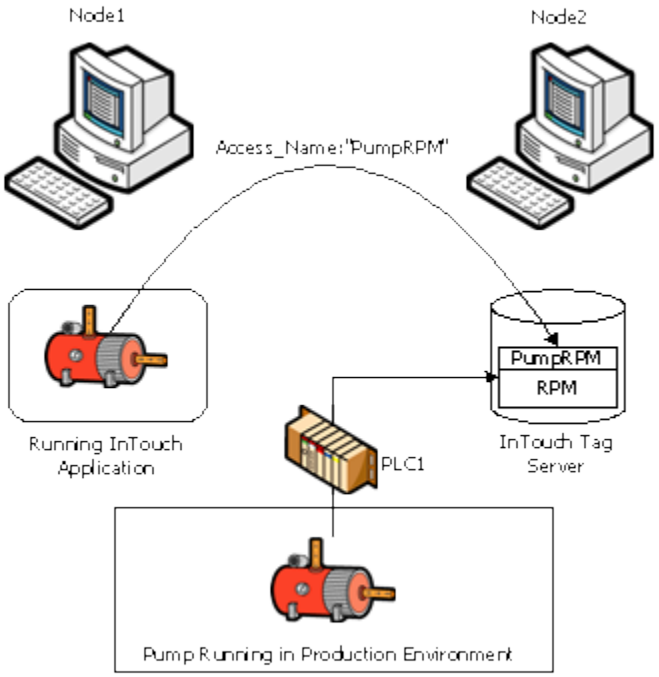
Tag Properties	Discrete	Integer	Message	Real
% Deviation		•		•
% per		•		•
Access Name	•	•	•	•
ACK Model	•	•		•
Alarm Comment	•	•	•	•
Alarm Group	•	•	•	•
Alarm Inhibitor	•	•		•
Alarm State	•			
Alarm Value		•		•
Comment	•	•	•	•
Conversion		•		•
Deadband		•		•
Deviation Deadband %		•		•

Tag Properties	Discrete	Integer	Message	Real
Eng Units		•		•
High		•		•
HiHi		•		•
Initial Value	•	•	•	•
Input Conversion	•			
Item	•	•	•	•
Log Data	•	•		•
Log Deadband		•		•
Log Events	•	•	•	•
Lo		•		•
LoLo		•		•
Maximum Length			•	
Major Deviation		•		•
Max EU		•		•
Max Raw		•		•
Max Value		•		•
Min EU		•		•
Min Raw		•		•
Min Value		•		•
Minor Deviation		•		•
Off Msg	•			
On Msg	•			
Priority	•	•	•	•
Rate of Change		•		•
Read Only	•	•	•	•
Read Write	•	•	•	•

Tag Properties	Discrete	Integer	Message	Real
Retentive Parameters		•		•
Retentive Value	•	•	•	•
Square Root Conversion		•		•
Target		•		•
Use Tagname as Item Name	•	•	•	•
Value Deadband		•		•

Remote tag references

You can create distributed InTouch applications with a tag server running on a separate node from the node running the InTouch application. The following figure shows an InTouch application that makes a remote reference to the PumpRPM tag from a tag server running on another node.



You create an InTouch application to reference tags located on a remote node by two methods:

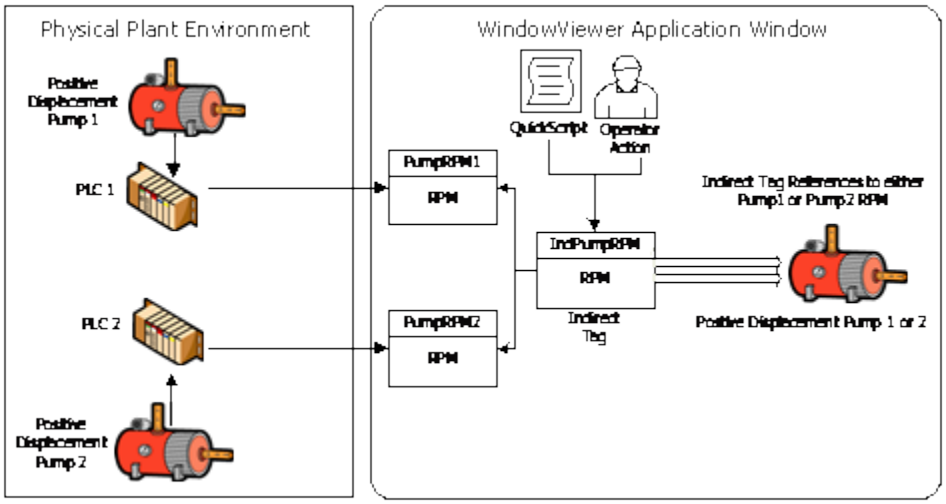
- Associate I/O tags with an Access Name that identifies a remote server as the tag source. For more information about defining an Access Name for an I/O tag, see Set up access names.
- Use a remote reference directly to **the tag**. For example, PLC1:PumpRPM.

For more information, see Access I/O data by remote references.

Define indirect tags

Using indirect tags, you can create applications with window objects that show values from multiple tags.

The figure below shows a pump object within an application window. The pump object represents two possible process pumps based upon values set from an indirect tag. A QuickScript or operator action selects the source tag associated with the indirect tag.



Indirect tags minimize your development time. You create fewer application windows because a single window object can represent multiple processes running in the production environment.

Use indirect tags with scripts

You can use scripts to assign input source tags to an indirect tag. You assign an input source tag to an indirect tag by assigning the source tag's name to the indirect tag's .**Name** dotfield.

For example, if you create an indirect analog tag called **IndPumpRPM**, the two source **PumpRPM** tags are assigned to it with script statements similar to the following example:

```
IF PumpNo == 1 THEN
   IndPumpRPM.Name = "PumpRPM1";
   ELSE
   IndPumpRPM.Name = "PumpRPM2";
ENDIF;
```

The indirect tag assignment script can be triggered by an application event or an operator action like selecting a window button.

When you equate an indirect tag to another source tag, the indirect tag behaves as if it is the source tag. If the value of the source tag changes, the indirect tag reflects the change. If the indirect tag's value changes, the source tag changes accordingly.

Because the .Name dotfield of an indirect tag is a simple string, you can dynamically assign the indirect tag target at run time. For example, if you create a Data Change QuickScript that runs each time the value of the **Number** tag changes, the source tag assigned to the indirect **IndPumpRPM** tag changes accordingly:

```
IndPumpRPM.Name = "PumpRPM" + Text(Number, "#" );
```

When this script runs, the value of the analog tag **Number** is converted to text and appended to the string **PumpRPM. If Number equals 1, this sets the name of the indirect IndPumpRPM tag** to **PumpRPM1**.

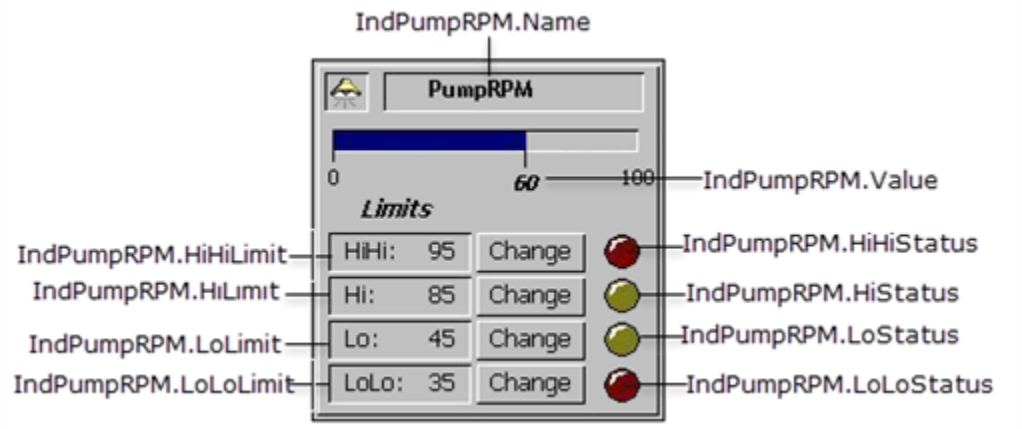
Indirect analog-type tags are used for both integer and real tags. Indirect tags can be mapped to any other tag as long they are the same tag type.

You can also assign retentive attributes to indirect tags. With retention, the indirect tag retains its most recent tag assignment when the application starts again.

Use indirect tags with local tags

Indirect tags are typically used with tags defined in a local Tagname Dictionary. An indirect local tag enables you to create visual objects that show multiple attributes of a local tag. For example, you can create a faceplate within an application window. The faceplate contains selectable items with links to an indirect local tag assigned to different dotfields. In the following example, operators modify dotfield alarm limits linked to a local indirect **PumpRPM** tag.

The following figure shows a faceplate with animation links to pump RPM alarm limits. An indirect tag assigns the attributes of different local tags to the alarm limit faceplate.



To redirect the faceplate to the appropriate tag, include a statement within a QuickScript.

```
Indirect_tag_name.Name = "tag_name";
```

In this script example, tag_name is the name of an actual tag defined in the local Tagname Dictionary. When the script runs, all dotfield values associated with this local tag become accessible to the application object through the indirect tag.

Use Indirect tags with remote references

Remote indirect tag references differ from local tag references. The syntax for a remote reference is:

AccessName:Item

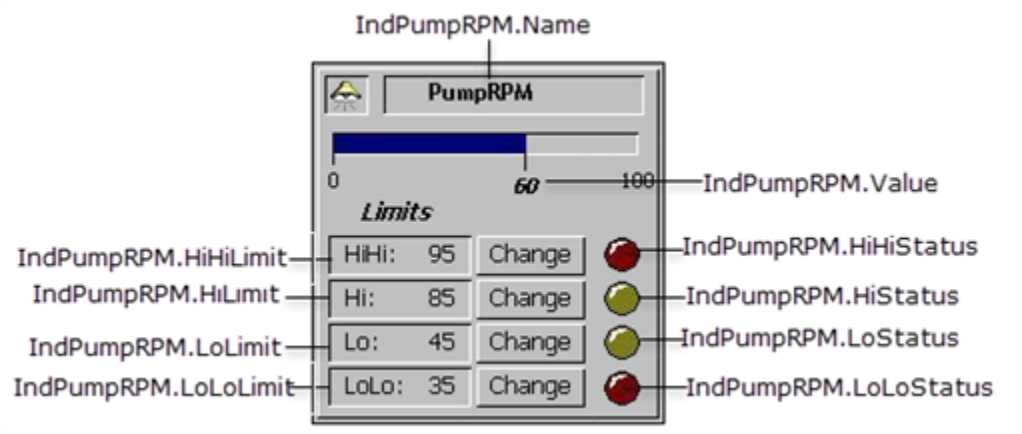
where

- *AccessName* is any valid InTouch Access Name.
- *Item* is any valid Item Name that is supported by the I/O Server specified in the Access Name definition.

When you use remote references, the server returns a value to the client, not a tag structure. The value includes a time stamp and a quality stamp. Thus, an indirect tag assigned to a remote reference cannot access any tag

dotfields other than those related to value, time, and quality. For example, an indirect tag cannot access tag attributes through a remote reference to specify alarm limits.

One possible solution is to create a faceplate with a set of indirect tags. The following figure shows a faceplate to modify the alarm limits for a pump.



In this example, the faceplate uses 10 indirect tags that are associated with an implied .**Value** dotfield. The alarm faceplate is being redirected to the remote reference tag, IndPumpRPM, on a remote InTouch node named TagServer1. An InTouch Access Name is configured as follows:

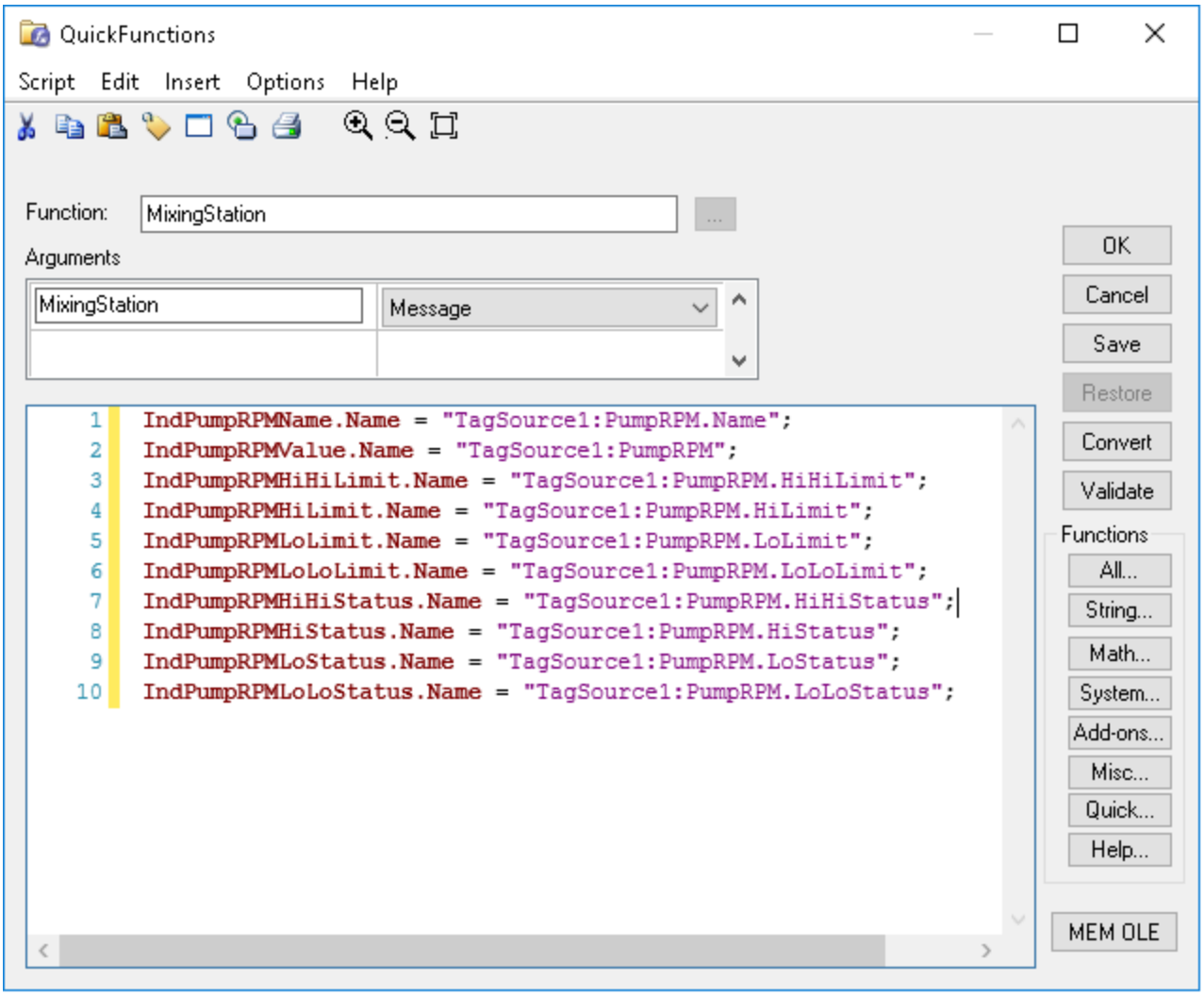
Access Name:	TagSource1
Node Name:	TagServer1
Application Name:	View
Topic Name:	Tagname

To redirect the faceplate to the remote reference tag PumpRPM, run the following QuickScript:

```
IndPumpRPMName.Name = "TagSource1:PumpRPM.Name";
IndPumpRPMValue.Name = "TagSource1:PumpRPM";
IndPumpRPMHiHiLimit.Name = "TagSource1:PumpRPM.HiHiLimit";
IndPumpRPMHiLimit.Name = "TagSource1:PumpRPM.HiLimit";
IndPumpRPMLoLimit.Name = "TagSource1:PumpRPM.LoLimit";
IndPumpRPMLoLoLimit.Name = "TagSource1:PumpRPM.LoLoLimit";
IndPumpRPMHiHiStatus.Name = "TagSource1:PumpRPM.HiHiStatus";
IndPumpRPMHiStatus.Name = "TagSource1:PumpRPM.HiStatus";
IndPumpRPMLoStatus.Name = "TagSource1:PumpRPM.LoStatus";
IndPumpRPMLoLoStatus.Name = "TagSource1:PumpRPM.LoLoStatus";
```

The script must run each time the faceplate is redirected. Another solution is to create an InTouch QuickFunction that enables you to write a single script and pass it the name of the remote reference. You can reduce the amount of script coding by using multiple faceplates that call the same QuickFunction.

For example, using a similar set of script commands, you can define a QuickFunction called **RedirectAlarmFacePlate**:



You can call the **RedirectAlarmFacePlate** function to handle the entire redirection. To do this, the function must be called by another InTouch QuickScript. For example:

```
CALL RedirectAlarmFacePlate ("TagSource1:PumpRPM");
```

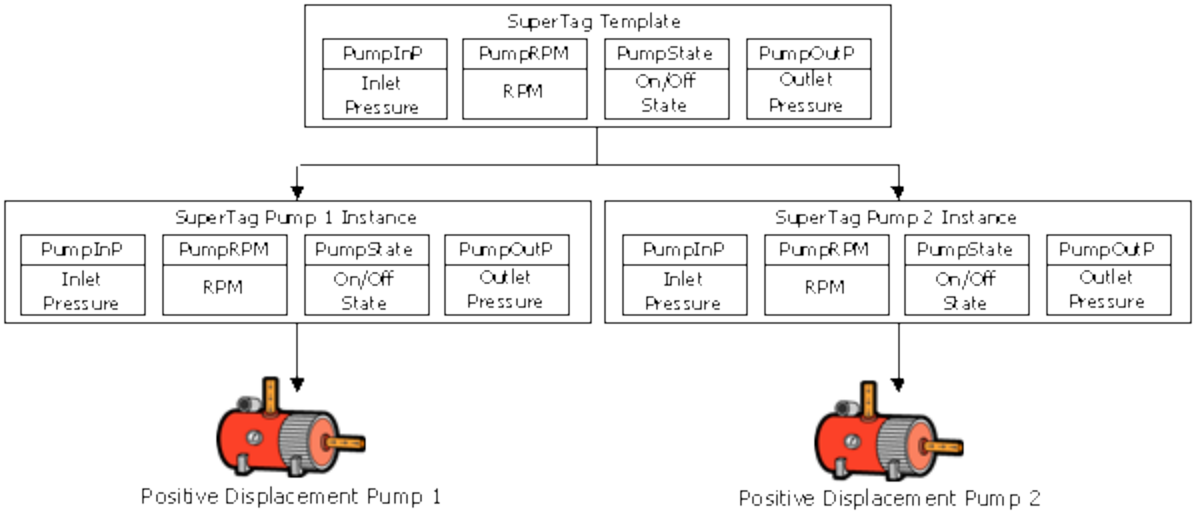
Define reusable tag structures

A Supertag is a template for a set of related tags. The tags that belong to a Supertag template are associated with the common properties of a component in a manufacturing process.

The Supertags save development time. Instead of creating a set of tags for every component in the manufacturing process, you can replicate a single Supertag template and create individual instances for all process components that have the same properties.

Supertag Templates

The figure below shows a single Supertag template consisting of a set of related tags associated with pump data. The template can be replicated to create instances of the Supertag of the identical pumps in the process.



All tags that belong to an instance of a Supertag behave exactly like normal tags. The member tags can be assigned as InTouch discrete, integer, real, and message data types. The tags support trending, alarming, and all tag dotfields.

A Supertag template can organize its member tags in two nesting levels. A Supertag template can contain up to 64 embedded child tags. Each child tag can contain up to 64 sub-member tags. This gives you a total of 4095 tags in a Supertag template.

The following figure shows how tags are organized within a Supertag template.

When one Supertag template parent is embedded into another Supertag template, the embedded tag becomes a child member.

After you create a Supertag parent template, the Tagname Dictionary lists it as a tag type in the **Tag Types** dialog box. The template can be selected immediately as a tag type when you create a new tag. You do not need to restart WindowMaker to define tags that use a newly created Supertag type.

Saving Supertag Templates

By default, all Supertag templates are saved in the Supertag.dat file in the **C:\ProgramData\Wonderware\InTouch** folder. You can choose to save the Supertag templates at a different location.

Specify the location to save the Supertag templates

1. Open WindowMaker
2. On the **File** menu, select **Configure**, and then select **WindowMaker**.

   The WindowMaker configuration screen appears.
3. In the **Supertag path** text box, specify the location to save the Supertag templates.

Import Supertags

To import the Supertags during the InTouch Application migration, see Importing Assets during InTouch Application Migration.

Manage Supertag templates and member tags

With the release of InTouch HMI 2023, the Template Maker utility has been deprecated. You can manage the Supertags using the Supertag window on the canvas. This allows you to create, edit, and delete Supertag templates and their member tags. After you create a Supertag template, the Tagname Dictionary **Tag Types** dialog box lists the Supertag template and an indirect tag form of the template as tag types.

You can modify a Supertag templates or member tags at any time. Existing Supertag instances derived from a template do not inherit the changes to the template. However, all new instances inherit the changes of the modified template.

You can delete an entire Supertag template or selected member tags that belong to the template. Deleted templates are no longer listed in the Tagname Dictionary **Tag Types** dialog box.

Edit an existing Supertag template or member tag

Note: We recommend you to use the Supertags window pane to manage the Supertags.

1. Open the **Supertags** pane.
2. Navigate to the Supertag template name or member tag that you want to edit.
3. Right-click the Supertag template and select **Edit Supertag**.

   Alternatively, double-click the Supertag template.

   The Supertags window appears on the canvas.
4. You can rename the Supertag template or modify the description.
5. Make your changes.
6. Select **OK**.

Delete a Supertag template or member tag

1. Open the **Supertags** pane.
2. Navigate to the Supertag template name or member tag that you want to delete.
3. Right-click the Supertag template and select **Delete**.

   A message requests confirmation to delete the selected item.
4. Select **Yes** to delete the selected template.

Important: Member tags of an instance of a Supertag template cannot be deleted. For example, if PumpRPM is a member tag of the TankPump Supertag template, it cannot be deleted from any instance of TankPump. You can only delete tags from the Supertag template.

Supertag instances

Supertag templates and a template instance are different. An instance is an actual implementation of a Supertag template in an InTouch HMI application.

The most important difference between a template and an instance is that the parent template name is replaced by the name of the instance tag. The child template names and the sub-member tags do not change.

Create a Supertag instance

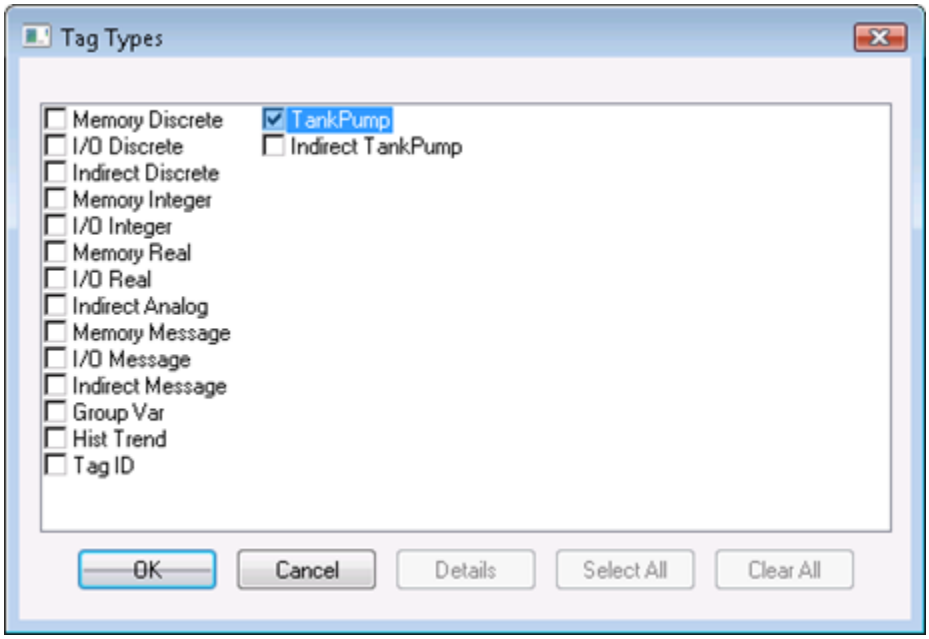
You can use the Tagname Dictionary to create a Supertag instance. The Tagname Dictionary automatically creates all member tags and sub-member tags when the you define the new Supertag instance.

Create an Supertag instance from a template

1. In the **Supertags** pane, select the Supertag template for which you want to create an Instance.
2. Right-click the Supertag template, and select **New Instance**.

   A new Supertag Instance is created.
3. Double-click the new instance to open the Tagname Dictionary.
4. In the **Tagname** box, type the name you want for the new Supertag instance.

   A Supertag instance name can be up to 32 characters. An instance name follows the same naming rules as regular InTouch tags.
5. Select **Type** to show the **Tagname Types** dialog box.
6. Select a Supertag template name from the list.



7. Select **OK**. The **Tagname Dictionary** dialog box expands to show additional options.

   The new tag that you entered in the **Tagname** box becomes the parent for all member tags that belong to the selected Supertag template.
8. Set the properties of the tag. Do the following:
   a. In the **Member List** box, select a tag from the Supertag template list.
   b. From **Data Access**, select **Memory** or **I/O** to show the respective Memory or I/O details dialog box.
   c. Enter the details as you do for a standard InTouch tag.
   d. Select the remaining member tags from the list and configure them.
9. Select **Close** after you specify all details for the member tags that belong to the Supertag instance.

Replicate a Supertag instance

You can use the Tagname Dictionary to replicate an existing instance. After you replicate an instance, the tags become immediately available for use in animation links and InTouch QuickScripts.

Replicate a Supertag instance from the Tagname Dictionary

1. Open the Tagname Dictionary.
2. Select **Select** to show the **Select Tag** dialog box with a list of tags defined for the application.
3. Select a Supertag instance from the list to use as the template for your new instance.
4. Select **OK**.

   The name of the selected template appears in the **Tagname** box.
5. Select **New**. A message requests confirmation to replicate the Supertag instance.
6. Select **Yes**. The **Enter Name** dialog box appears and you are prompted for the name of the new Supertag.

   Enter a name up to 32 characters using standard tag naming conventions.
7. Select **OK**. The new Supertag instance appears in the Tagname Dictionary.
8. If needed, edit the member tags as you do for a normal InTouch tag.
9. Select **Close**.

Add tags to a Supertag instance

You can add a tag as a member of an existing Supertag instance using the Tagname Dictionary.

When you add a tag, you enter the exact name of your Supertag instance followed by the backslash (\) delimiter and the name of the new member tag.

For example:

```
Pump_8\PumpSTS
```

Note: If you plan to use the Bulk Import Utility to migrate the tags from your InTouch HMI application to Application Server, see Import Supertags with the Bulk Import Utility for more information about substituting the standard backslash delimiter.

Add a tag to a Supertag instance

1. Open the Tagname Dictionary.
2. To add a tag to a Supertag instance, do the following:
   a. Select **New**.
   b. In the **Tagname** box, type the exact name of your Supertag instance followed by the backslash (\) delimiter and the name of the new member tag.
   c. Select **Type**.
   d. Select the tag type for the new member tag you are adding to the instance.
   e. Select **OK**. The details dialog box for the member tag's type appears.
   f. Enter the required details as you do for a normal InTouch tag.
3. Select **Save**.
4. Select **Select**.

5. Select the Supertag instance that you added a member tag.
6. Select **OK**.

   In the **Member List** box, all member tags that belong to the Supertag template are listed.

   The new member tag appears in the list.

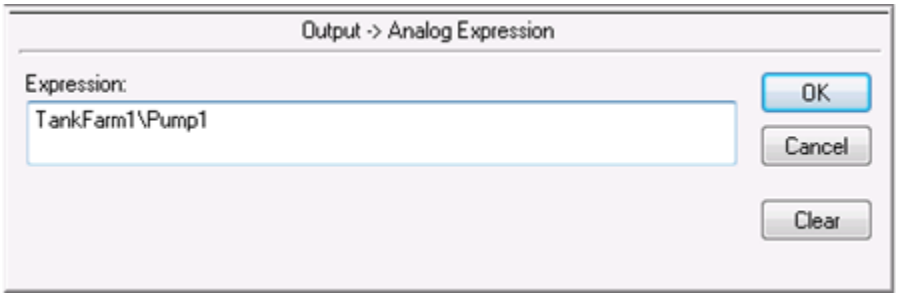
Other ways to create Supertags

You can also create Supertags using the following methods:

- In the Animation link expression input box
- Within InTouch QuickScripts.
- In an external file that you load into an application with the InTouch DBLoad utility

  Note: When you use an alternative method to create the member, the Supertags configuration screen does not show the member in the Supertag template definition.

When you create a Supertag through an animation expression or InTouch QuickScript, you must use the valid Supertag format. For example:



Note: If the Supertag instance and member tag you specify in an animation expression or QuickScript are not currently defined, you are prompted to define the tag. Select OK. The Tagname Dictionary appears and shows the Supertag instance and member tag that you created.

The following syntax examples are valid:

```
ParentInstance\ChildMember ParentInstance\ChildMember\Submember
```

The following syntax examples are not valid:

```
ParentInstance\
ParentInstance\ChildMember\
```

If an invalid format is used, an error message box indicates the Supertag syntax contains an error.

Set a Supertag instance as the Owning Object

Owning Object can be used for relative referencing. The OwningObject property is available when you have one graphic embedded into a second graphic. You can set the owning object properties in Industrial Graphic Editor, under **Runtime Behavior** section of the Configuration **Properties** tab. Enter the instance name with the `InTouch:` prefix for both managed and standalone InTouch applications. Example, InTouch:Tank01. You can also set the owning object using scripting. OwningObject property sets the owning object of the graphic shown by the ShowGraphic() script function. It can be a concatenation of constant strings and reference strings. Example: graphicInfo.OwningObject = "UserDefined_001";

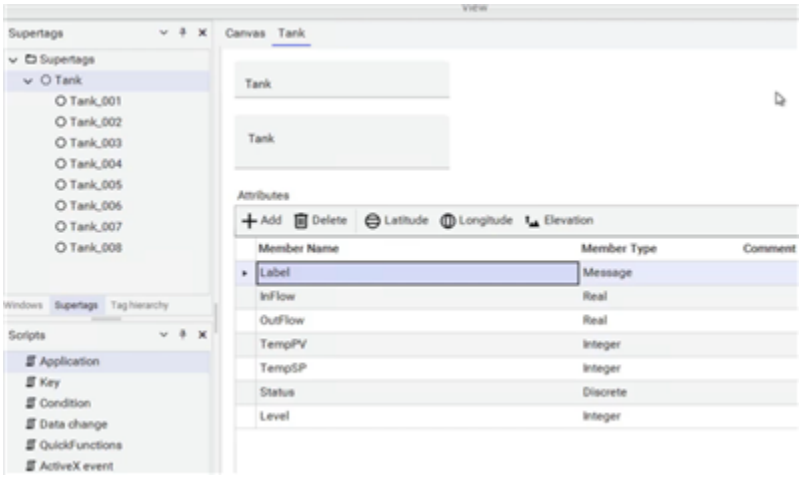
Note: The OwningObject property sets references for the graphic, but is not associated with the GraphicName property if the symbol is part of an Object Wizard. Therefore, if you are scripting a symbol with an owning object, specify the owning object name as part of the GraphicName property, for example, UserDefined_001.Pump_001.

Owning Objects in Supertags

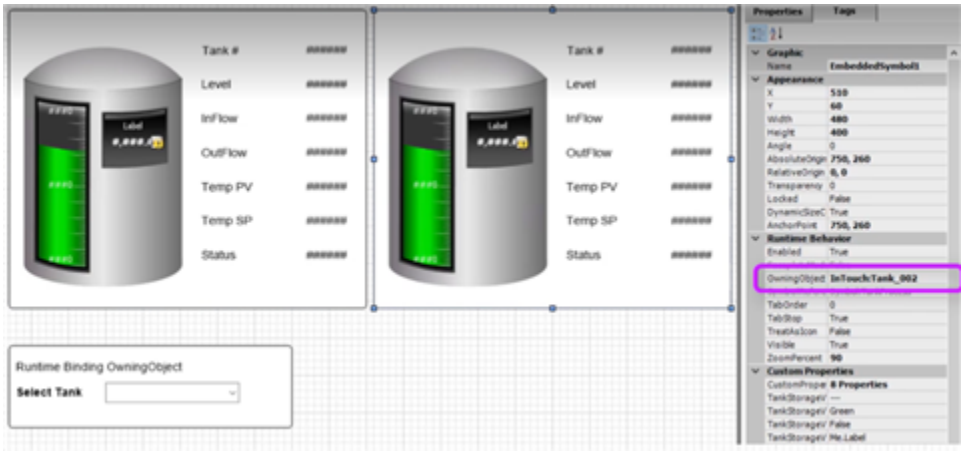
You can use relative references with a Supertag. You can create a symbol and use the owning object property based on a Supertag. This feature allows you to set Supertag instances as the owning object for a graphic. This helps you to rapidly develop the application without the need of configuring individual symbols.

For example:

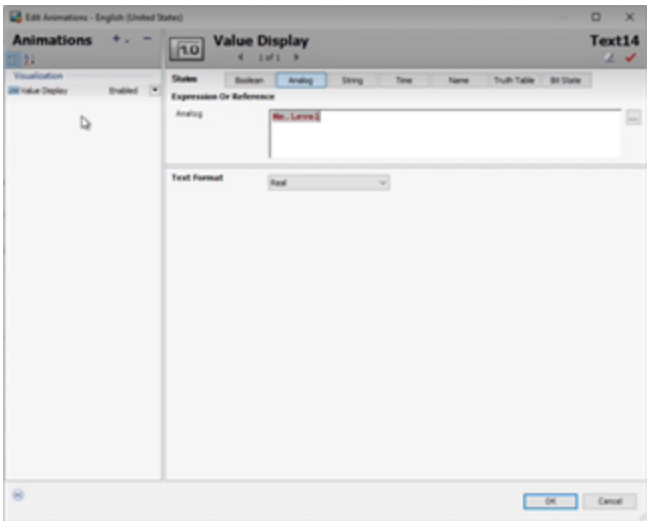
Consider a Supertag named Tank with seven attributes added and has eight instances created namely, Tank_001, Tank_002 till Tank_008 as shown below:



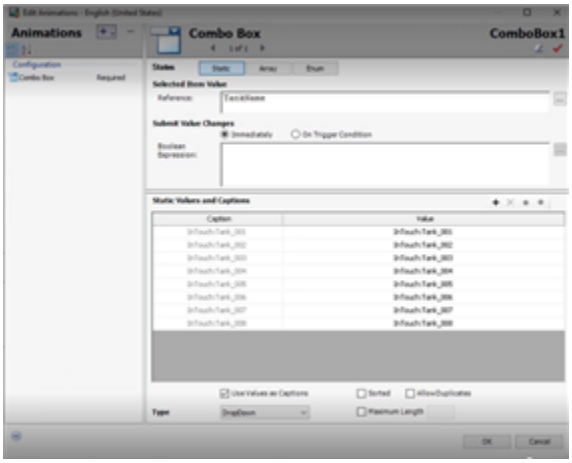
Each of these tank Supertag instances is set as an owning object. To represent these tank instances, in the Industrial Graphic Editor, these tank symbols are added.



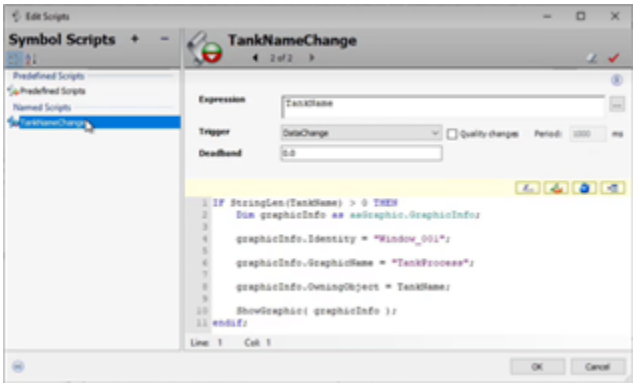
For each tank symbol, the value display for the attributes is set as relative references.(The graphic being embedded should include "me.Attribute" for relative reference to work. The attribute being referenced, for example "me.Level" will bind to the attribute of that Supertag instance at runtime.)



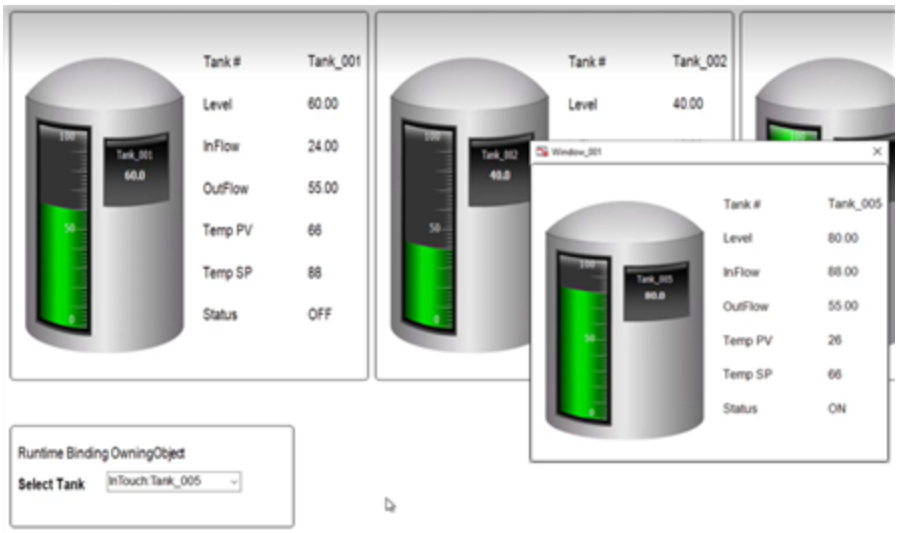
A combo box which is referencing a custom property name as Tank Name is added to the graphic and the values in the combo box are set as the names of the Supertag instances.



In the scripts for this graphic, a data change script is configured to execute when the tank name custom property is changed. This script uses show graphic option to show a tank and it sets the owning object to be the value of the custom property tank name.



After switching to runtime, in the combo box you can select any tank and the corresponding tank with its different attribute values is displayed.



Supertag attributes

Using the Supertag window on the canvas, you can add attributes to the Supertag templates. For each attribute, you can also configure the properties such as member name and member type.

Add or modify a member attribute

1. Open the Supertag window in the canvas.
2. In the **Attributes** section, select **Add**.

   A new attribute is added to the grid.
3. Double-click the **Member Name** cell to modify the name.
4. Double-click the **Member Type** cell to select from the available options are Integer, Real, Message, and Discrete. By default, the Member Type field is set to Integer.
5. Double-click the **Comment** cell to modify the description.

Add or modify a location attribute

1. Open the Supertag window in the canvas.
2. In the **Attributes** section, select **Latitude, Longitude,** or **Elevation**.

   A new attribute for latitude, longitude, and elevation is added to the grid.
3. The Member Name displays **Latitude, Longitude, or Elevation** respectively for the selected location attribute.

   Note: You cannot modify the Member Name of a location attribute.
4. Double-click the **Member Type** cell to select from the available options are Integer, Real, Message, and Discrete. By default, the Member Type for a location attribute is set to Real.
5. Double-click the **Comment** cell to modify the description.

Delete an attribute

1. Open the Supertag window in the canvas.

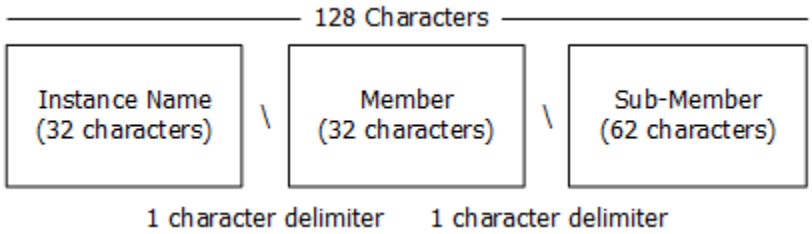
2. In the **Attributes** section, select an attribute that you want to delete.
3. Select **Delete**.

   The selected attribute is deleted.

Reference Supertag members

InTouch tag names can be a maximum of 128 characters. Each Supertag ParentInstance\ChildMember\Sub-member can be a maximum of 128 characters. The maximum length of a tag name places a restriction on references to Supertags.

A Supertag reference can be a maximum of two templates (ParentInstance\ChildMember) and one member deep, as shown in the following figure.



Note: A Supertag sub-member can be up to 62 alpha numeric characters if the attribute type is not set to any existing Supertag. A sub-member based on a Supertag type can be only up to 32 alpha numeric characters long.

Each member in a Supertag template is accessible in the standard format currently used to access the dotfields of standard InTouch tag types. The Supertag reference syntax is supported throughout InTouch where standard tags can be used. For example, a valid Supertag dotfield reference is:

```
TankFarm\Tank1\Pump1RPM.RawValue
```

Remote tag references also support Supertags. For example, a valid Supertag remote reference is:

```
PLC1:"TankFarm\Tank1\Pump1RPM.RawValue"
```

Import Supertags with the Bulk Import Utility

You can use the Bulk Import Utility to transform tag definitions from InTouch into an Application Server object structure. The Bulk Import Utility enables you to more efficiently migrate your InTouch tags to Application Server. In addition to standard InTouch tags, the Bulk Import Utility can also migrate Supertags.

If you plan to migrate Supertags from an InTouch application to Application Server, replace the backslash character (\) within your Supertag reference to a supported character like an underscore (_).

Example:

```
TankFarm_Tank1_Pump1RPM.RawValue
```

Note: At the time of this release, there is a known limitation with the DBDump and DBLoad process. The Supertag Instances created by importing from a .CSV file will be displayed only in the tag dictionary and not in the Supertag pane.

Use MapApp widget with Supertags

The MapApp Widget allows you to view a map containing graphics within a running application. At run time, the

map enable users to pan to different areas of the map and zoom in or out to show more or less map detail. The graphics placed in a map typically represent business assets located within an area shown by the map.

For more information, see Map_App Widgets Properties in the Widgets Help.

You can integrate Supertags with MapApp widget to display it in WindowViewer or Web Client.

Follow the workflow to configure the MapApp widget in InTouch HMI.

Step 1: Create Supertags and Supertags instance.

Step 2: Create graphic for the Supertags

Step 3: Configure the Map Settings for the Supertags

Step 4: Configure and use an MapApp widget

Step 5: View the MapApp in WindowViewer or Web Client.

Create Supertags and Supertags instance

The MapApp widget uses the Supertag detail and the location attributes of the instances configured in the Supertag pane of the WindowMaker.

Add the location attributes:

1. Create Supertags and instances.
   For example, you can create an asset $Tank and instances called Tank1 and Tank2.
2. Add attributes to each of the instances. Example, Flow and Temp.
3. Add the location attributes such as "Latitude" and "Longitude".
4. Use the tag dictionary to set the attribute values.
   Example:
   Double click on Tank1 to open the tag dictionary. Set below member (attribute) values:
   Tank1\Flow = 11
   Tank1\Latitude = 33
   Tank1\Longitude = -114
   Double click on Tank2 to open the tag dictionary. Set below member (attribute) values
   Tank2\Flow = 21
   Tank2\Latitude = 36.746
   Tank2\Longitude = -119.773

Create Graphic for the Assets

Use the Graphic Toolbox to create graphics to represent the assets.

Example: Graphics created areTank1Sym and Tank2Sym

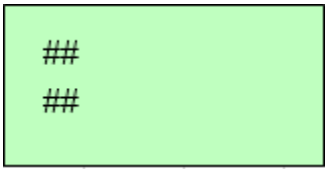
Add a graphic to an asset

- Type the name of the graphic
  Or, drag & drop the graphic from the Graphic Toolbox.

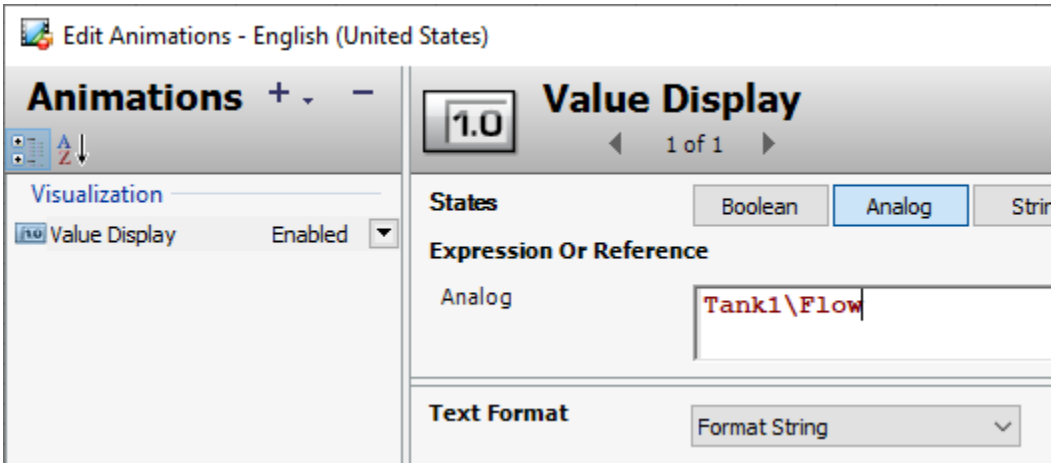
Example:

Use Tank1Sym for Tank1 asset's graphic

Use Tank2Sym for Tank2 asset's graphic



The graphic can have reference to the other attributes of the asset instance, such as the flow and temp.

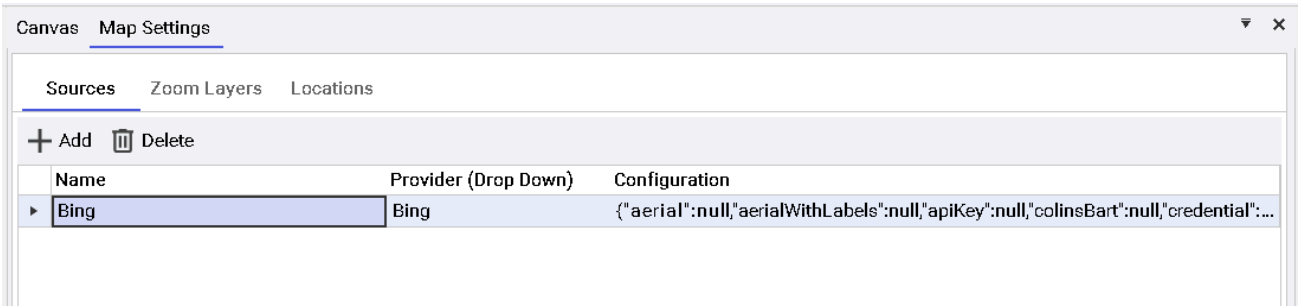


Configure the Map Settings for the Assets

Use the MapApp widget to configure the map settings for each asset instance.

Configure the map settings

1. Open WindowMaker.
2. In the Industrial Graphic Toolbox pane, navigate to **Widgets**, and double-click **Map_App**.
   The **Map Settings** configuration screen appears as a tab on the canvas.
3. Configure the Sources for the Map



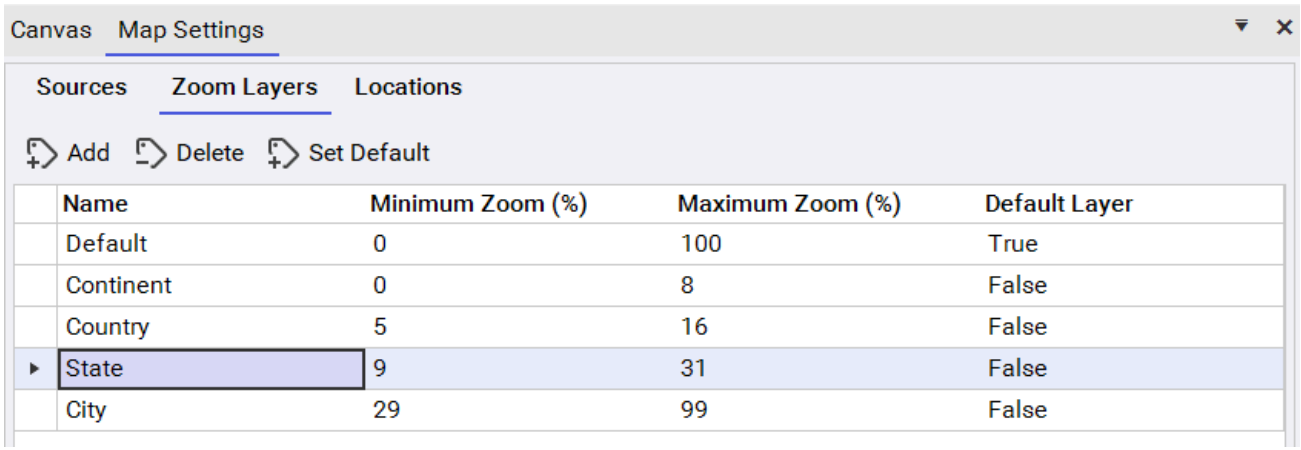
   a. Select **Add**.
      A new entry is created in the Sources grid.
   b. Enter a name of the source.
   c. Select a map provider from the Provider list, such as Bing.

This is optional. If no map provider is selected, the MapApp uses the default Map App provider.

d. Enter the Configuration value
APIKey: An6Grh5_YKz-_CYnREoNn3nOaBU_rlnVEoc7o8HpSj3GWCvjseEguPvN3rJkZ95T

4. Configure the zoom layers.

The zoom layers allow you to add the zoom percentage for the map.



Name	Minimum Zoom (%)	Maximum Zoom (%)	Default Layer
Default	0	100	True
Continent	0	8	False
Country	5	16	False
State	9	31	False
City	29	99	False

To add a zoom layer:

a. Select **Add**.

b. Enter a name to the zoom layer. The Name provided here populates in the Layer column of the Locations tab.

For example, you can set the layer for Tank1 as Default, and the Tank2 as "State.

c. Enter the minimum and maximum zoom percentage.

For example, if you set the minimum and maximum zoom percentage for the State layer as 9% and 31% respectively, then the graphic Tank2Sym is only shown in the map if the zoom percent is between 9% to 31%.

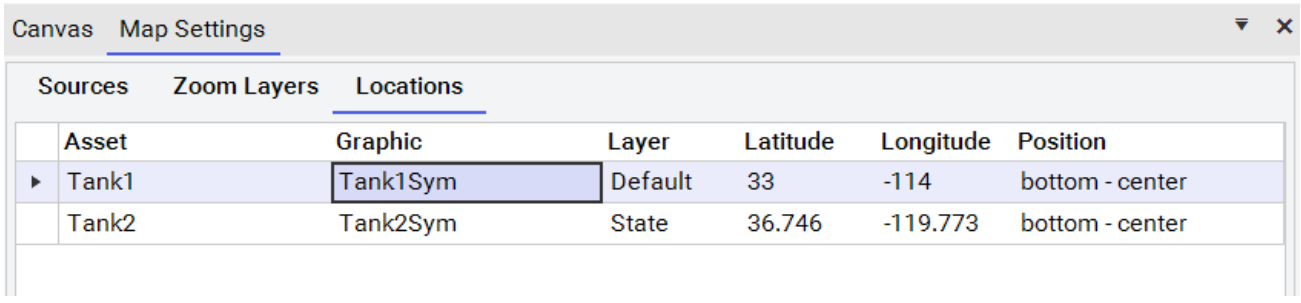
d. Set a Default Layer: The zoom layer which is set as default shows True, and all other entries show False.

To delete a Zoom Layer, select **Delete**.

To set a zoom layer as default, select **Set Default**.

5. Configure the Locations

The location details of each of the asset is retrieved from the Asset attribute value configured in the Asset configuration screen.



Asset	Graphic	Layer	Latitude	Longitude	Position
Tank1	Tank1Sym	Default	33	-114	bottom - center
Tank2	Tank2Sym	State	36.746	-119.773	bottom - center

The Layer information is retrieved from the Zoom Layers tab.

For example, you can set the layer for Tank1 as Default, and the Tank2 as "State".

This means that the graphic Tank2Sym is only shown in the map if the zoom percent is between 9% to 31%.

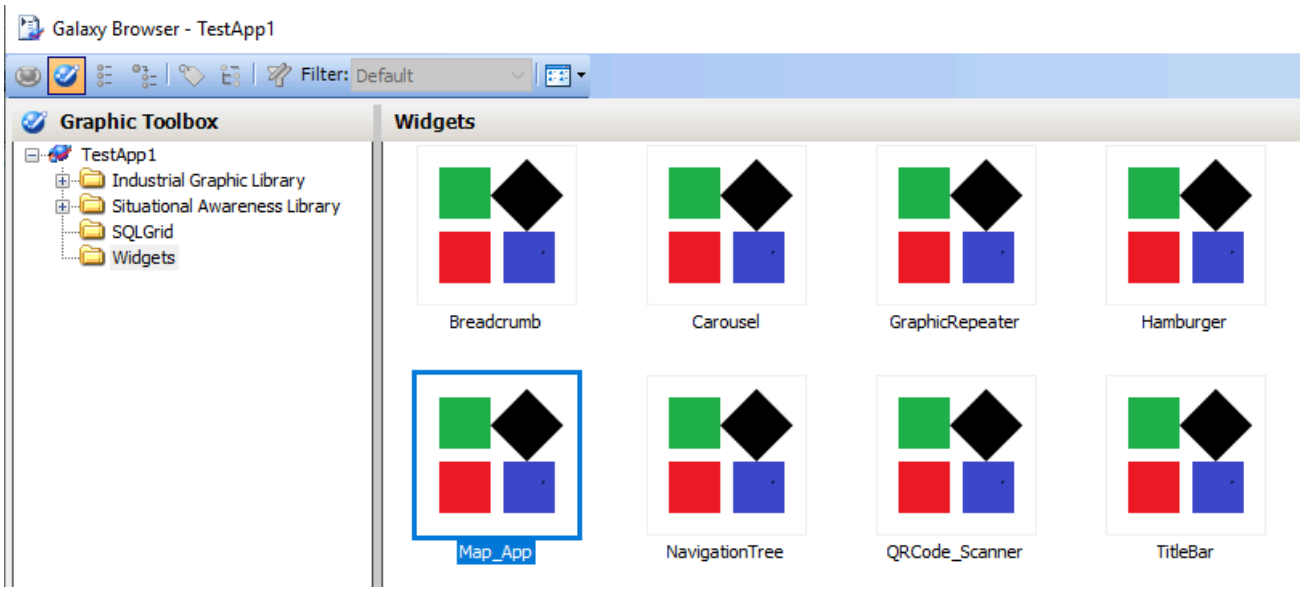
6. Select **OK** to save and close the page.

Embed the MapApp widget

You must embed the MapApp widget into a graphic to view it on a map.

Embed the MapApp widget:

1. Create a new graphic. Example, MyApp.
2. Edit this graphic in graphic editor.
3. Right-click anywhere in the graphic editor and select **Embed Industrial Graphic**.
4. In the **Galaxy Browser**, select **Widgets** in the Graphic Toolbox.
5. From the list of Widgets, select **Map_App**.



6. Select and embed the Map_App widget on to the canvas and adjust the size.
7. Edit the widget property for the Map_App widget.

   The Config name is the mandatory field. We recommend you update atleast the following attribute values – ConfigName, CurrentZoom, InitialLatitude, InitialLongitude, and InitialZoom.

   For example, you can set the attribute values as follows:

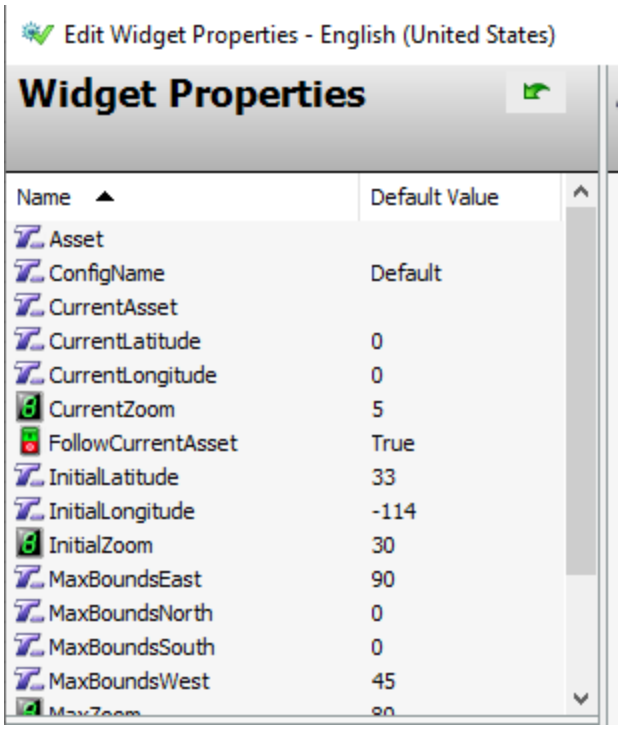
   ConfigName = Default

   CurrentZoom = 5

   InitialLatitude = 33

   InitialLongitude = -114

   InitialZoom = 30



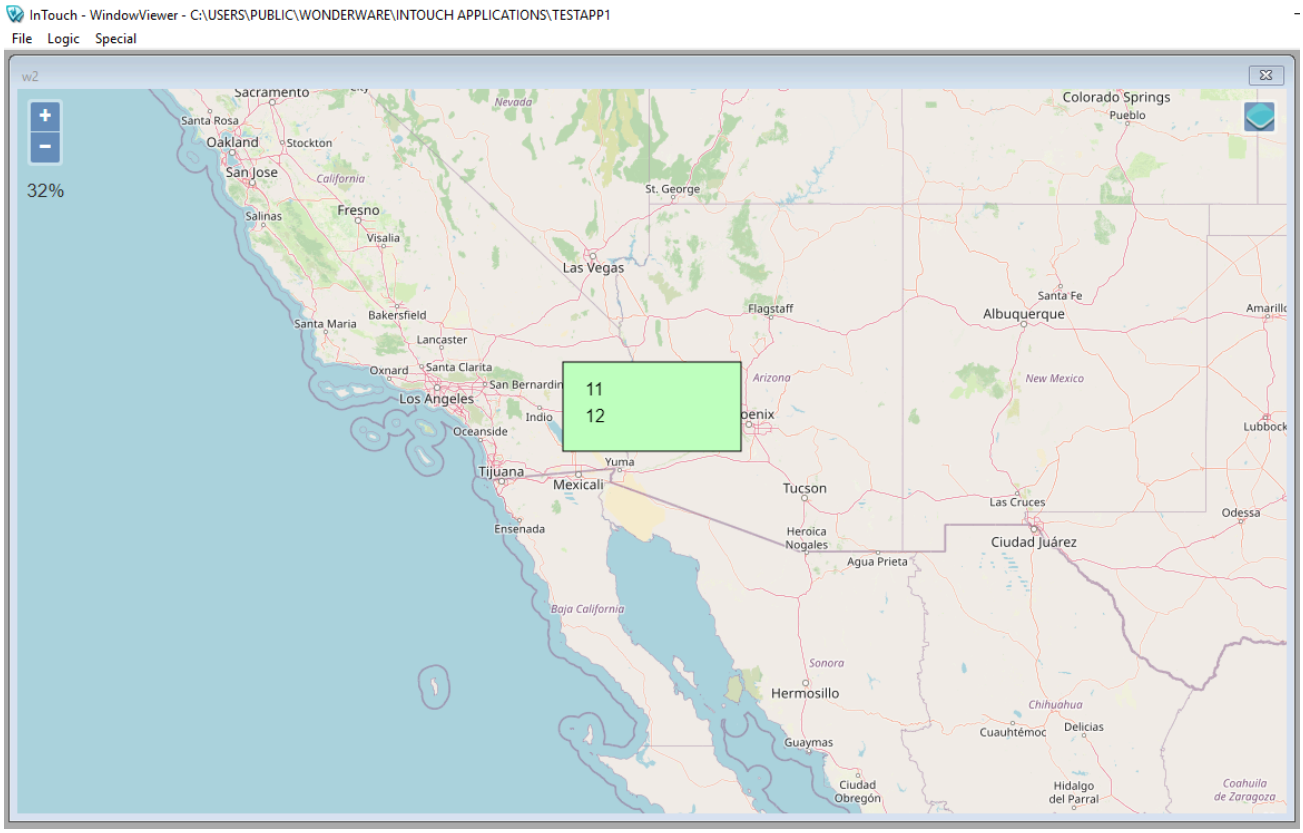
8. Select **OK**.
9. Embed the graphic on to a frame window and then close and save the window.

View the MapApp in WindowViewer or Web Client

You can view the configured asset and MapApp either in WindowViewer or Web Client.

View the MapApp in WindowViewer

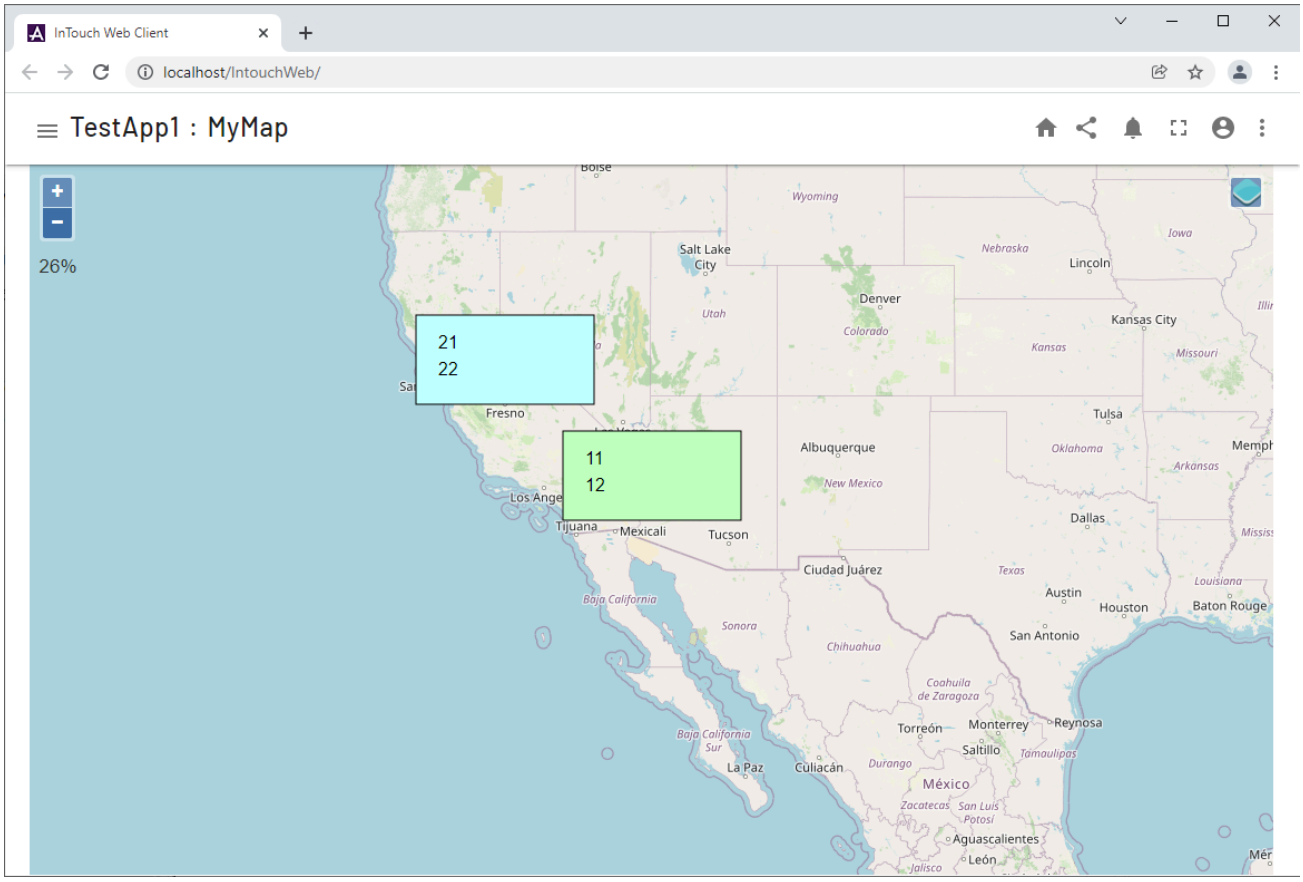
1. Fast switch to WindowViewer and then open the window containing the MyMap graphic.
2. The Map_App widget and the Supertags are displayed in the configured location on the map.



3. Adjust the zoom percentage to zoom-in our zoom-out on the map.

View the MapApp in Web Client

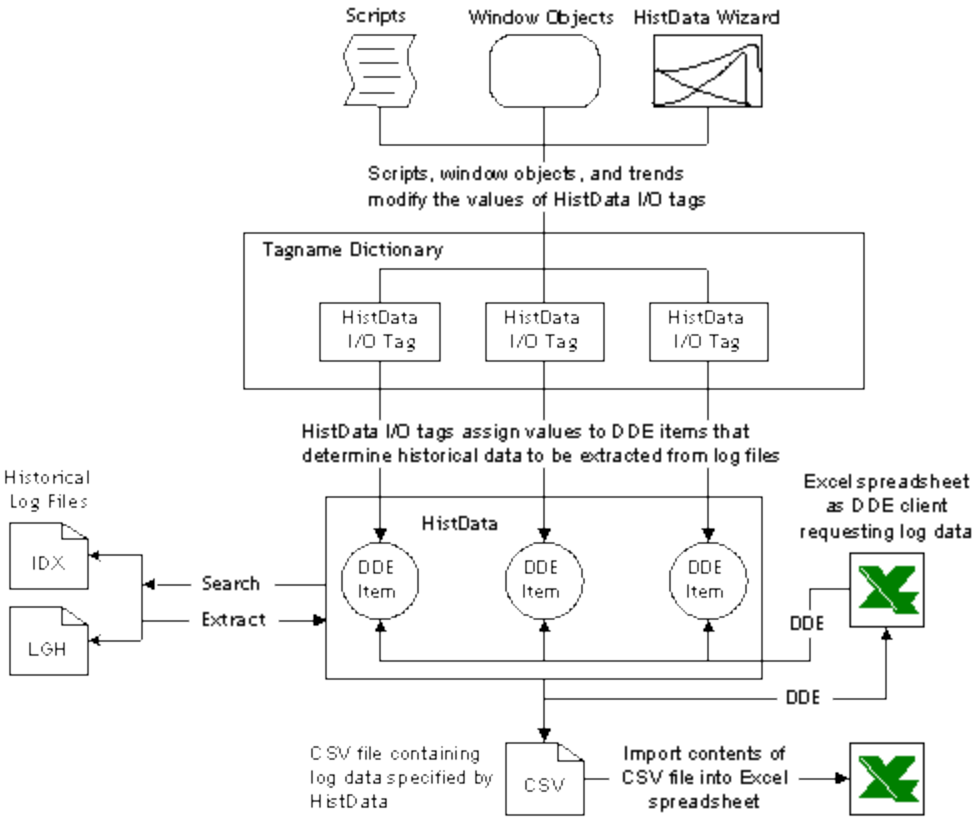
1. Install Chrome browser and make it the default browser.
2. With WindowViewer running, launch .
3. From the menu, select "MyMap" graphic.Web Client
   The Map_App widget and the Supertags are displayed in the configured location on the map.
4. Adjust the zoom percentage to zoom-in our zoom-out on the map.



Access historical tag values from other applications

You can use the InTouch HistData utility to extract data from historical log files to a comma separated value (.csv) file. Applications like Excel can extract InTouch log data directly from HistData as a DDE client or import log data from the output file created by the HistData utility.

The figure below shows the process to save selected historical log data to a file or a DDE client application.



Use DDE items to show historical data

The HistData program includes a set of DDE items that specify how historical data is extracted from log files. These items are part of the HistData internal database. You assign a value to each item.

The following table summarizes HistData items defined in the HistData program.

Item	Data Type	Descriptions
DATADIR	Message	Path of the folder containing historical log files.
DBDIR	Message	Path of the folder containing the contents of the InTouch Tagname Dictionary.
STARTDATE	Message	Start date to extract data from the log file. The format of the start date is MM/DD/YY.
STARTTIME	Message	Start time to extract data from log files. The format of the start time is HH:MM:SS using a 24-hour clock.

Item	Data Type	Descriptions
DURATION	Message	Length of the data collection interval from log files. **DURATION** can be expressed as: • Weeks (w) • Days (d) • Hours (h) • Minutes (m) • Seconds (s) Fractional **DURATION** periods can be specified. For example, `DURATION=0.5m` is equivalent to 30 seconds. To request a single sample, set **DURATION** to 0.
INTERVAL	Message	Length of time between data collection intervals. **INTERVAL** can be expressed in weeks, days, hours, minutes, and seconds. The units of time of an **INTERVAL** period are the same as a **DURATION** period. Fractional intervals can be specified. For example, `INTERVAL=0.25d` represents 6 hours. The maximum period for **DURATION** or **INTERVAL** is six weeks. The maximum six week period applies to any time value assigned to **DURATION** or **INTERVAL**. For example, 42 is the maximum number of days of an **DURATION** or **INTERVAL** period.
FILENAME	Message	Name and folder location of the file containing data extracted from the historical log file.
WRITEFILE	Integer	Flag that indicates the status of HistData write operation to the output file. When set to 1, HistData writes the requested data to the file specified by the **FILENAME** Item

Item	Data Type	Descriptions
		Name. When the file update is complete, **WRITEFILE** automatically resets to 0.
ERROR	Message	String containing a description of the last error that occurred while extracting data from log files. When **STATUS** is set to 1, the **ERROR** string is set to None. When **STATUS** is set to 0, the **ERROR** string contains an error message.

TAGS1, TAGS2,...	Message	String containing the name of one or more tags whose data is extracted from the log files.
		The **TAGS** string can be 131 characters in WindowViewer and 255 characters in Excel.
		The string can be appended for longer requests by adding tag items named Tagsn, where *n* represents an incrementing integer.
		If a tag needs additional tag text, place a plus (**+**) at the end of the string.
		For example:
		`TAGS="$Date,ProdLevel,ProdTemp,+"`
		`TAGS1="ReactLevel,Temp,GasLevel,+"`
		`TAGS2="MotorStatus"`
		Duplicate tags are not allowed and the maximum length of each tag's string is 512 bytes.

PRINTTAGNAMES	Discrete	Flag that indicates whether the names of tags are placed above the associated column of values. When set to 1, tag names are printed. When set to 0, tag names are not printed.

Item	Data Type	Descriptions
DATA	Message	This item holds the requested data in the HistData program in comma separated values format. It is used by other applications to **ADVISE** or **REQUEST** data by DDE.
STATUS	Discrete	Status of the most current HistData operation. A value of 1 indicates HistData successfully extracted historical data from the log file. A value of 0 indicates that an error occurred.
SENDDATA	Integer	Flag that indicates the status of the HistData update operation. When set to 1, HistData updates the **DATA** item with the requested data. When the update is complete, **SENDDATA** automatically resets to 0. If you receive an error message indicating too much data was requested using **SENDDATA**, shorten the **DURATION** period or reduce the number of requested tags. Duplicate tags are not allowed and the maximum length of each tag's string is 512 bytes.

Access log data with DDE

You can use two methods to extract log data to an output file.

- Use the manual method if you want to save historical log data from eight or more tags to the output file.
- Use the HistData Wizard instead if you only want to save log data mapped to the pens currently assigned to a historical trend.

Manually extracting log data with HistData

You can manually extract log data to an output file. Complete the steps in the following order:

1. Create a HistData Access Name
2. Create I/O tags for HistData

3. Create a HistData window
4. Run HistData

Create a HistData Access Name

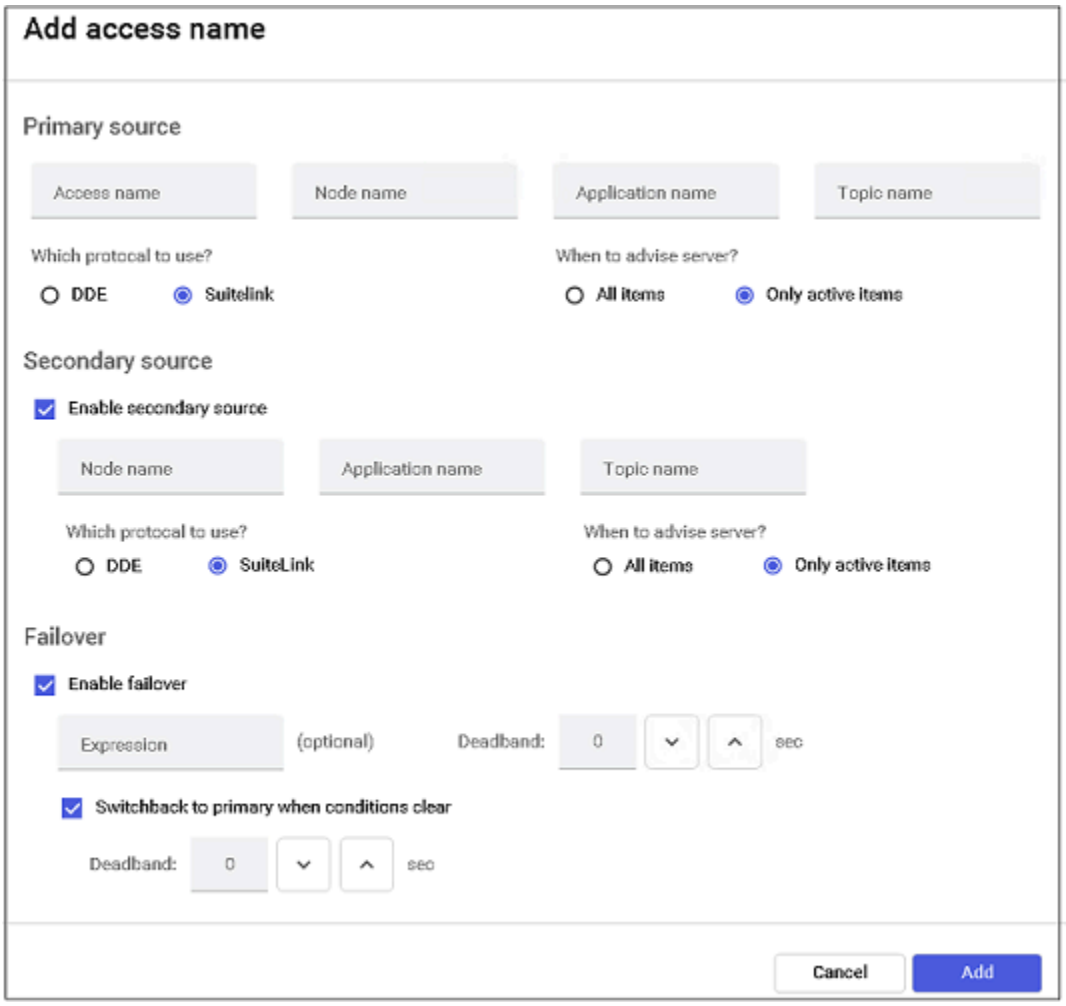
For InTouch to request data from the HistData program, you must define an Access Name.

Define an access name

1. On the **Home** menu, in the **Tags** group, select **Access names**.

   The **Access Names** dialog box appears.
2. Select **Add**.

   The **Add access name** dialog box appears.



3. In the **Access name** box, type a name up to 32 alphanumeric characters. The values assigned to **Access name** and **Topic name** should be the same.
4. In the **Node name** box, type the name of the node where the log files are currently located.
5. In the **Application name** box, type HistData without the .exe file name extension.
6. In the **Topic name** box, type the name you specified from the **Access name** box. **Access name** and **Topic name** should be the same.

7. Select the appropriate communication protocol from the available options: **DDE** and **Suitelink**.
8. In the **When to advise server** area, select **All items** whenever HistData is used.
9. Select **Add**.

Create HistData tags

After defining an Access Name, create the following I/O type tags to generate one output file that contains the data from the log files. Assign the Access Name created in the previous step to the tags.

Tag	I/O Tag Type	Item
HDWDATADIR	Message	DataDir
HDWDBDIR	Message	DbDir
HDWDURATION	Message	Duration
HDWERROR	Message	Error
HDWFILENAME	Message	FileName
HDWINTERVAL	Message	Interval
HDWSTARTDATE	Message	StartDate
HDWSTARTTIME	Message	StartTime
HDWSTATUS	Message	Status
HDWTAGS, HDWTAGS1, HDWTAGS2	Message	Tags
PRINTTAGNAMES	Discrete	PrintTagNames
HDWWRITEFILE	Integer	WriteFile

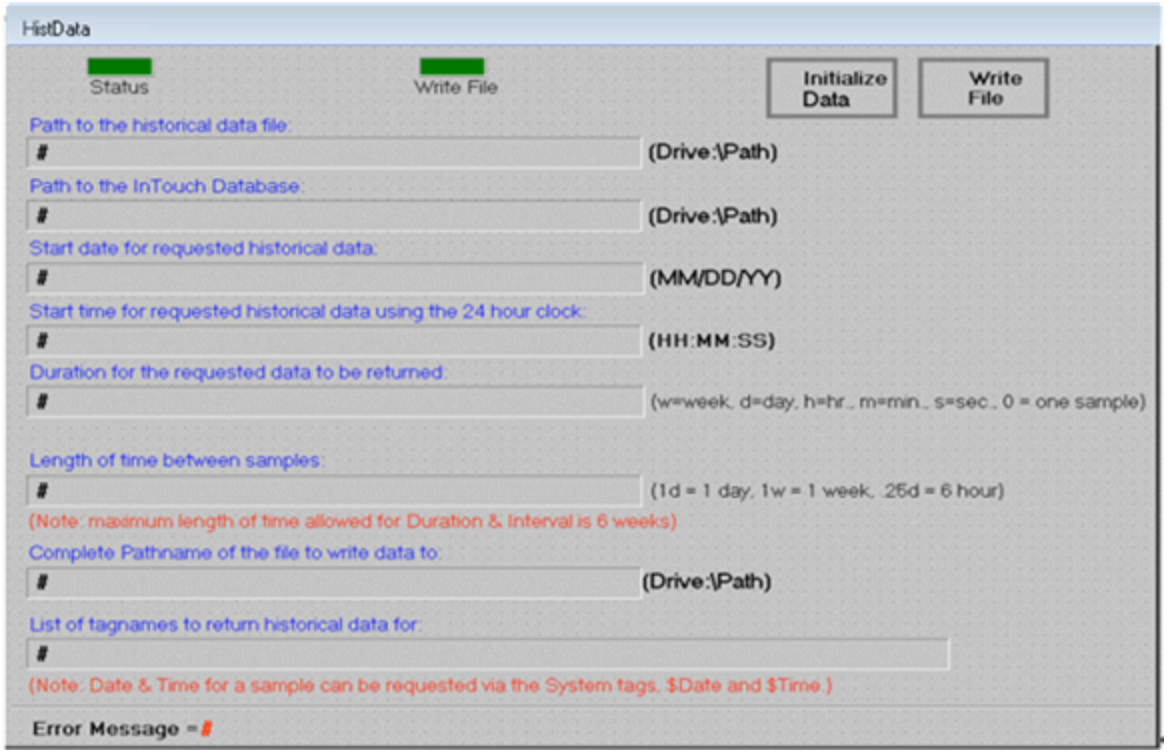
Note: The HistData Wizard creates these tags automatically except for the PRINTTAGNAMES tag.

Create two additional tags if you want to send log data to the Data item so that it can be accessed from other applications. Also, the HistData Wizard does not automatically create the HDWSendData and HDWData tags.

Tag	I/O Tag Type	Item
HDWSendData	Discrete	**SendData**
HDWData	Message	**Data**

Create a HistData window

After you create the I/O type tags, create a new window called **HistData** similar to the following example:

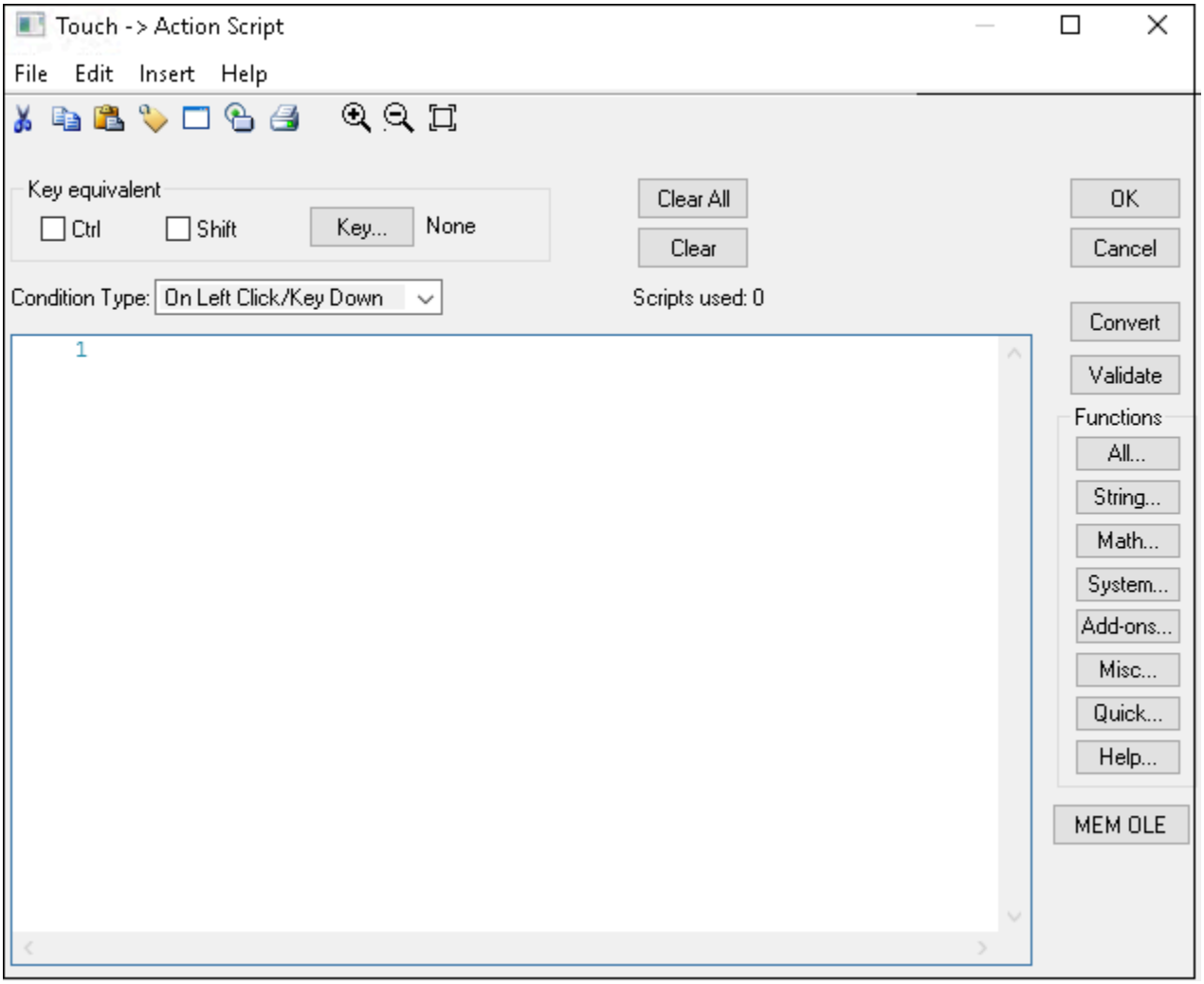


The # symbols are linked to a user input link. For example, the # symbol has a User Inputs/String link to the HDWDataDir tag. The user input link allows you to change the value of the tags during run time.

The **Status** button is linked to a fill color—discrete expression, based on the HDWStatus tag.

The **Write File** button is linked to a fill color—discrete expression, based on the HDWWriteFile tag.

The **Initialize Data** button is linked to a Touch Pushbutton Action script.



When the **Initialize Data** button is selected, the HistData items are initialized with the desired values. If needed, these values can also be changed during run time by using the User Inputs Links.

The Write File button is linked to a Touch Pushbutton Action script:

When selected, the **WriteFile** button generates the output file.

Run HistData

After creating the HistData window, complete the following steps to run it under WindowViewer.

Run the HistData window

1. Start HistData and minimize it.
2. Start WindowViewer and open the HistData window.
3. Select the **Initialize** button and make changes to the HistData items.
4. Select the **Write File** button.

   If the operation is successful, the value of Status is ON and the color associated with the ON status appears. If the operation is not successful, the value of Status is OFF and Error Message shows the cause of the failure.

Use the HistData Wizard to extract log data

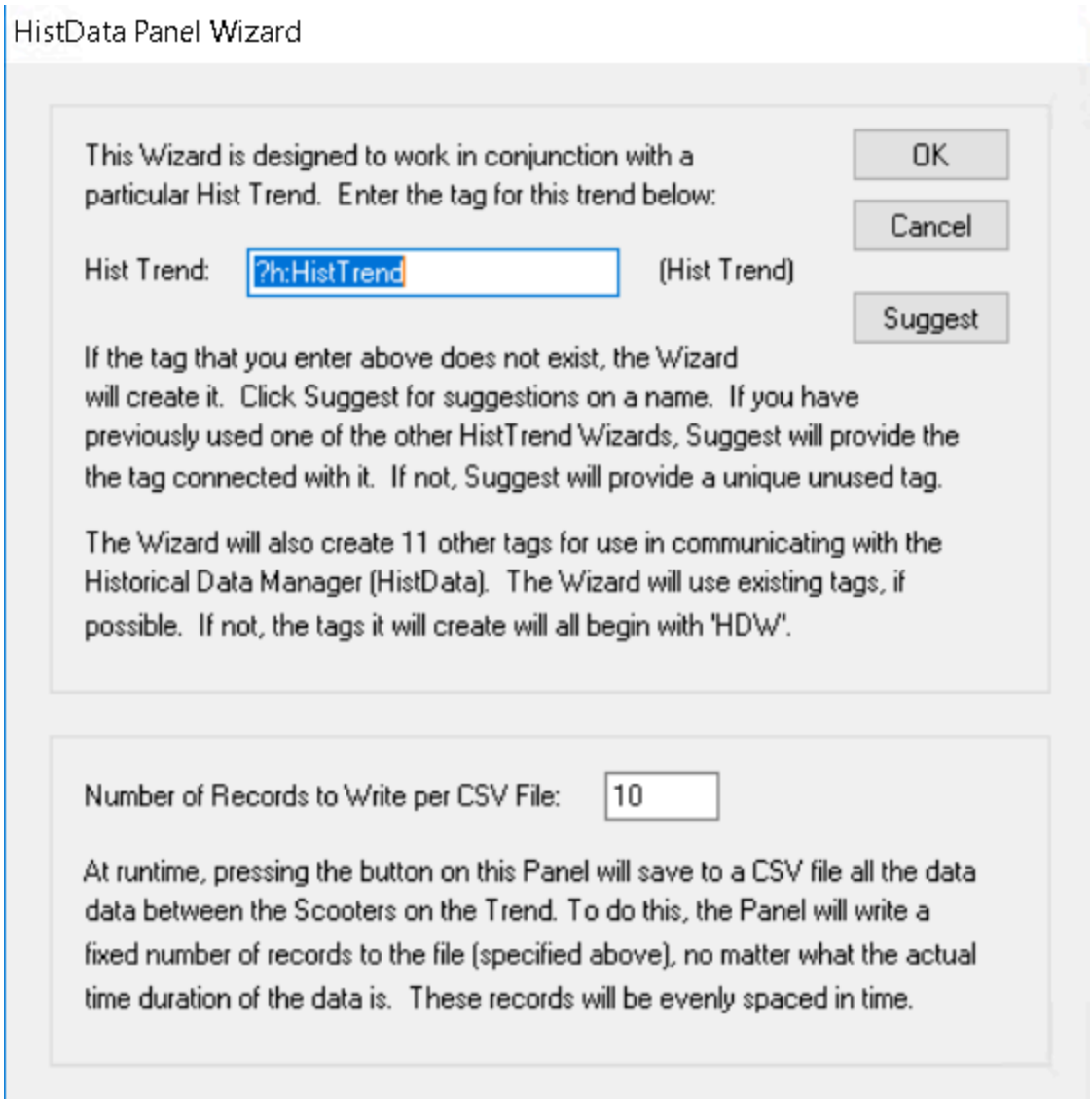
You can create an output file containing log data that appears in a historical trend. InTouch includes the HistData Wizard to automate the steps to extract data from a log file.

Because the HistData output file writes log data shown in a historical trend, the file can only contain data from the tags currently assigned to the pens of the historical trend.

Use the HistData Wizard to extract log data

1. Start WindowMaker.
2. Open a window that includes a historical trend.
3. On the **Draw** menu, in the **Insert** group, select **Wizards**.

   The **Wizard Selection** dialog box appears.
4. Select the **Trends** group from the left pane.
5. Select the **HistData Wizard** icon from the right pane and select **OK**.
6. Move your mouse pointer over an area of the trend window where you want to place the HistData object.
7. Select to place the HistData object in the trend window. The HistData Wizard creates a window object consisting of a button and the **Filename** box that shows the path where the output file will be created.
8. Double-click the HistData Wizard object placed in the trend window. The **HistData Panel Wizard** dialog box appears.



9. In the **Hist Trend** box, type a name for the HistTrend tag.
10. In the **Number of Records to Write per CSV File** box, type the number of records to write to an output file.
11. Select **OK**. The HistData Wizard creates a set of tags that are identified with a HDW prefix.

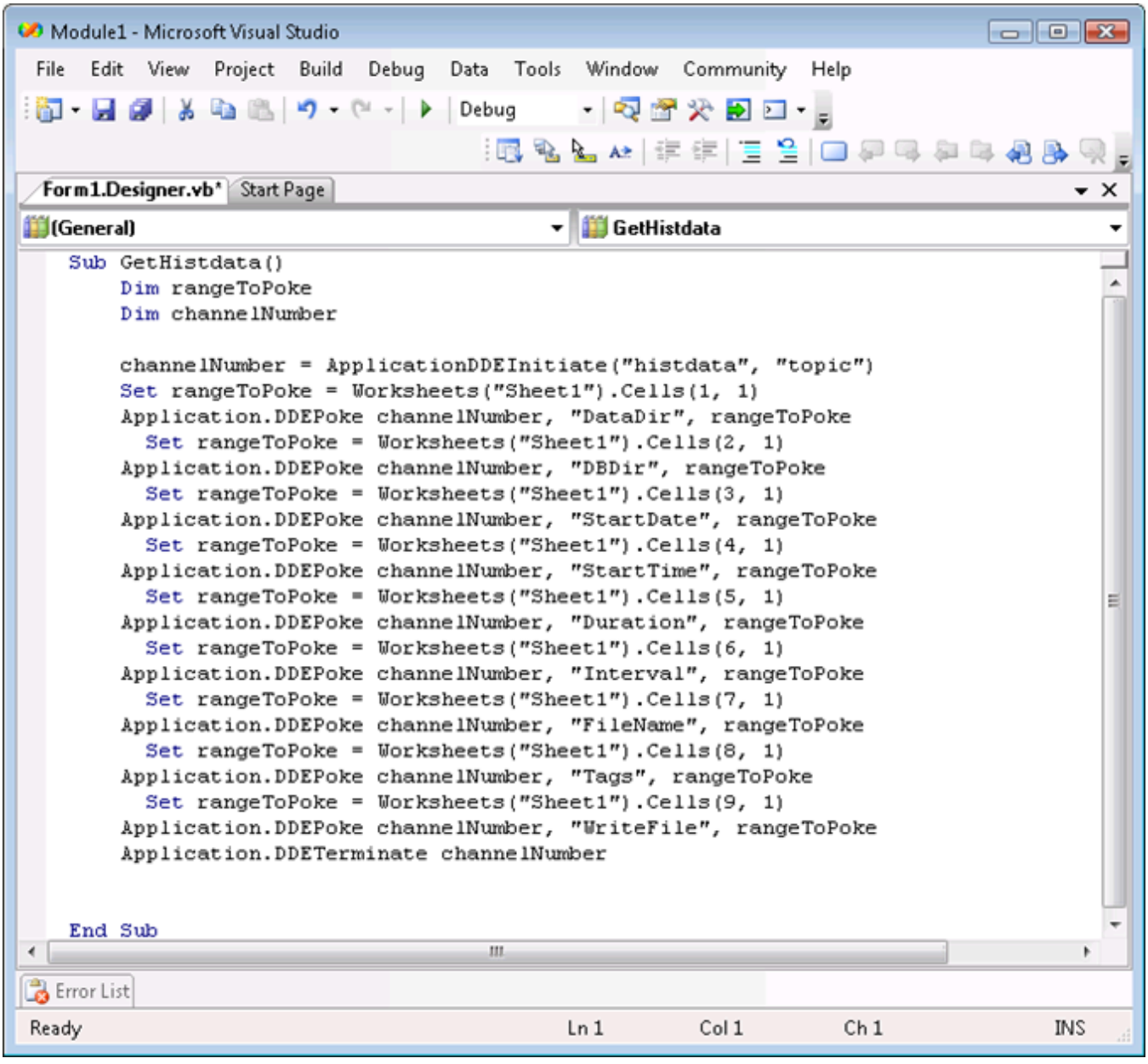
    The HistData Wizard creates the tags listed in Create HistData tags. The HistData Wizard assigns the tags to the HisDataViewSt Access Name.
12. Run the historical trend window with WindowViewer.
13. Select **Save to File** that is part of the HistData window object. HistData creates the output file in the folder location shown in the window object.

Note: HistData does not populate the .csv file properly if the tag values do not change within an hour of starting InTouch with the newly-created *.idx and *.lgh files.
The HistData data does not have the same resolution as for historical trend scooters. The HistData resolution is based on the number of values requested within the time span. It is not an exact reflection of the values that are in the *.idx and *.lgh files.

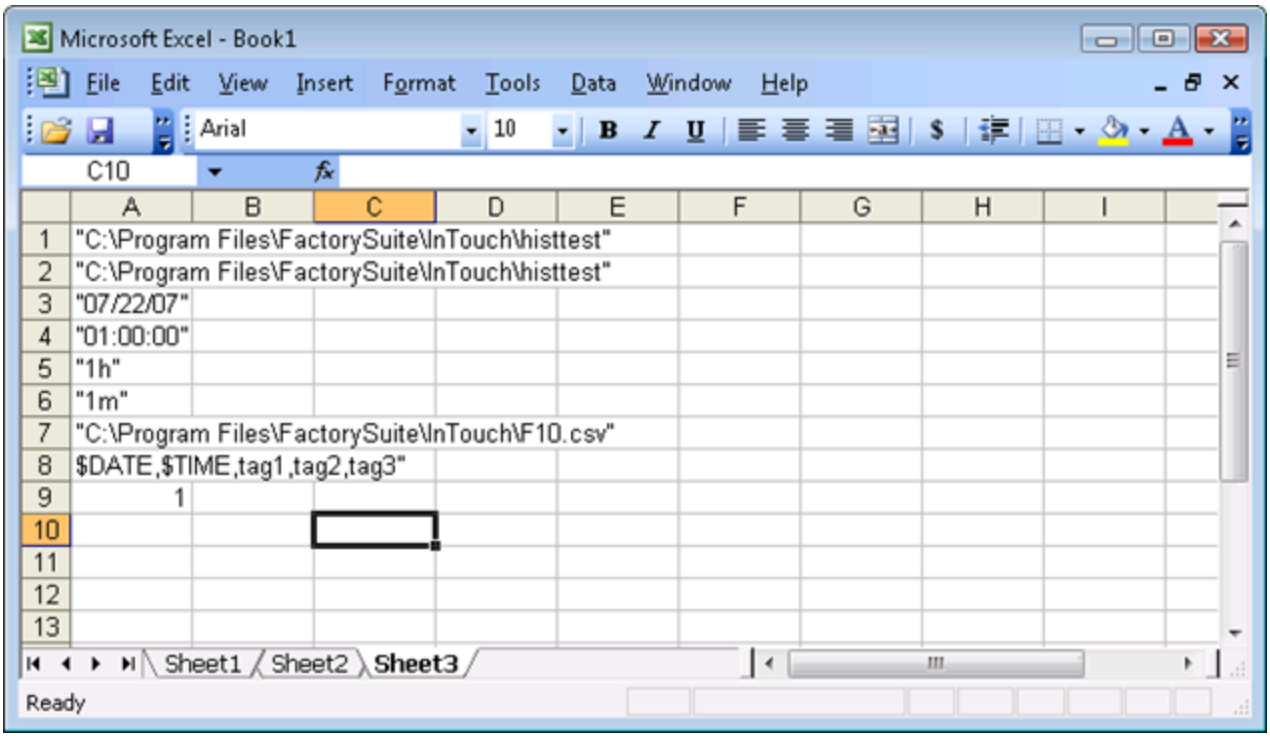
Access historical data from other applications

You can write Excel macros to extract data from a HistData file. The HistData program responds to the INITIATE, POKE, and TERMINATE functions within the macro. The POKE function with a keyword (an internal database item) sets the parameters that define a query. After the query is properly specified, run the macro to request the selected historical data from the HistData file.

The following example shows a macro written with VBA:



In the example above, the data to be poked is in Sheet1. The following example shows the data to be poked:



Troubleshoot HistData errors

You may see errors that can occur when extracting log data with HistData. The following table lists typical HistData problems or error messages in the left column. The right column describes possible causes and solutions to the problem.

Error Messages or Condition	Cause and/or Solution
Error Message: `Too much data requested – shorten the duration or reduce the number of tagnames.`	This error occurs when too much data is requested by the SendData item. If the only purpose is to create an output file containing data from the log files, do not use the **SendData** item.
Error Message: `Could not open file C:\FILES1\HISTDATA.CSV`	The folder path does not exist or the spelling of the folder path is incorrect.
Error Message: `Could not open file C:\FILES\`	No output file is defined.
Error Message: `DATADIR item invalid`	The destination folder path specified by the **DataDir** item does not exist. Verify the spelling of the folder path.

Error Messages or Condition	Cause and/or Solution
Error Message: `STARTDATE item invalid`	The **StartDate** item contains an invalid format for the starting date. From Windows, change the computer's date format to mm/dd/yy.
Error Message: `No log files found`	There are no log files for the requested date in the path specified by the **DataDir** item.
Error Message: `Could not find tagname TAG• in database`	The requested tag does not exist in the application's Tagname Dictionary. Verify the name of the tag is spelled correctly.
Error Message: `Could not find tagname.x in: C:\IT6.0B\HISTEST`	The file **tagname.x** does not exist or it is corrupted.
No output .csv file is created and no errors appear.	• HistData is not running. • The **Tags** item does not list any tags that have been specified for logging. • The HDWWritefile is incorrectly defined. Make sure the tag is an integer tag, the DDE Access Name is correct, and the item is **WriteFile**. Also, make sure there is no scaling where **MinEU**=**MinRaw** and **MaxEU**=**MaxRaw**.
The output .csv file contains date and time stamps but does not contain any logged data for the requested tags.	There are no entries in the Historical log for the tags during the requested time period. Display a historical trend to verify if the log file contains data during the requested period.
The WWLogger contains the following message: `Error for DDE HistData|Viewstream1! WriteFile: Poke was rejected by the server.`	The error message is written to the WWLogger each time the creation of the .csv file fails because of errors assigning values to the HistData items. This error can also occur if you try to set the **WriteFile** item to 0, or if you try to write to the error item.
The .csv file contains only a single record when there should be many records from the log file.	The **Interval** item may be assigned an incorrect value, which creates a very small collection interval. Also, the **Duration** item may have an invalid format such as `HDWDuration=1-` (no increment specified).

IEEE decimal units

The Historian HMI uses the Institute of Electrical and Electronics Engineers (IEEE) 754 standard to transform 32-bit binary values to floating-point decimal numbers.

IEEE 754 32-bit numbers are stored in 16-bit Programming Logic Controllers (PLCs) as two 16-bit words. The floating point registers in the PLC are usually sequential in their numbering scheme for the low and high hexadecimal words. The current generation of 32-bit personal computers use a single 32-bit register. The register's bit numbering scheme follows the same format as two sequential 16-bit registers.

To use floating point numbers in an Historian application, the Historian must be able to transform the values stored in the two 16-bit PLC registers. Bit conversions must be made because Historian always regards the raw PLC register values as individual integers. It is not possible to perform a Boolean **AND** operation of the two register integers and transform them into a real number. Historian cannot perform a type conversion for dual integer registers.

Show floating point numbers in the Historian HMI

The Historian HMI uses the IEEE 32-bit floating point format to show real numbers in an application. The IEEE floating point format is only an approximation of an actual real number. Unless the real number is an even power of two, it cannot be represented exactly using the IEEE 32-bit floating point format. The precision of an IEEE 32-bit floating point number is approximately eight decimal places.

When you want to show a real number in an Historian application, make sure the number does not exceed eight digits. The following floating-point number formats show valid real numbers within an Historian application:

#

#,###

#.##

0#

###.##

#.#######

###.#####

######.##

ABCDEF

###.####

Any floating-point numbers with more than eight digits are subject to rounding errors.

If you do not include a decimal in the format of the text, then the number is displayed with decimals, per the real format decimal precision configured in WindowViewer's Advanced Format properties.

Note: If you add "#" to the left of a decimal, or if there is no decimal, do not limit the number of digits displayed.

Example 1

A Historian application should show the real number 2.3. But, the number 2.3 is not an even power of two and cannot be precisely represented by the IEEE 32-bit floating point format beyond 8 decimal digits.

To ensure the value 2.3 is shown from the application as the ASCII characters 2.3, the number must not exceed

eight digits. If the number exceeds the eight digit maximum, the resulting number may be shown as 2.29999999 or 2.30000001 instead.

Example 2

When two real tags values are compared, the difference of two real tag values should be greater than FLT_EPSILON (value 1.19209290E-07F, in decimal 0.0000001192092896). However is the number exceeds 8 digits the resulting value may be incorrect. To correct for this multiply the value by 1000 or a larger multiple of 10. By doing this the value is be greater than 1e-7. Perform the necessary comparison operations and then divide by 1000, or the number you multiplied by.

Configure and using the InTouch OPC UA server

InTouch HMI supports the OPC UA (Unified Architecture) protocol for machine-to-machine communications.

This OPC UA server functionality allows third party clients to interact with InTouch HMI and leverage industry standards, such as OPC UA. When OPC UA Server functionality is enabled on InTouch HMI, a client can utilize the built-in Gateway Communication driver or a third party client to connect to InTouch HMI, and use Gateway Communication Driver or the client to securely browse the OPC UA namespace and interact with InTouch HMI.

OPC UA configuration checklist

Required tasks for end-to-end configuration of the OPC UA server and OPC UA client

The configuration tasks are shown in the order in which they must be completed.

1. **Configure the System Management Server:** The System Management Server is used for establishing a trust relationship between machines, and must be configured to ensure secure communications between nodes. The System Management Server is normally configured during initial System Platform installation. See *System Platform Installation*, "Configure the System Management Server," for details.

   Note: InTouch HMI must be run in the context of a user with Administrative privileges, which gives InTouch HMI access to the encryption certificates that enable secure communications.

2. **Configure the OPC UA server:** Set the configuration options, test the OPC UA Server connection, and activate the Gateway Communication Driver..
3. **IT compliance/firewall validation:** Firewall configuration and verification must be completed at this point of the configuration. The node to which the OPC UA Server has been deployed must have Inbound Rules for the firewall configured and verified.

   IMPORTANT! A firewall test must be successfully performed before proceeding with the remaining configuration tasks.

4. **Configure the OPC UA Client:** Client configuration may include the following:
   - Define the OPC UA server address (in the format `opc.tcp://<ServerName>:<PortNumber>`).
   - Select the correct OPC UA server security policy (`Basic256Sha256`).
   - Add the users to the InTouchHMIOPCUAWriteUsers user group.
   - Enter the configured OPC UA User Credentials (username and password)
   - Anonymous Connections are supported only for reading InTouch tags. To avoids any security risks, it is

recommended to access the data using authenticated credentials.

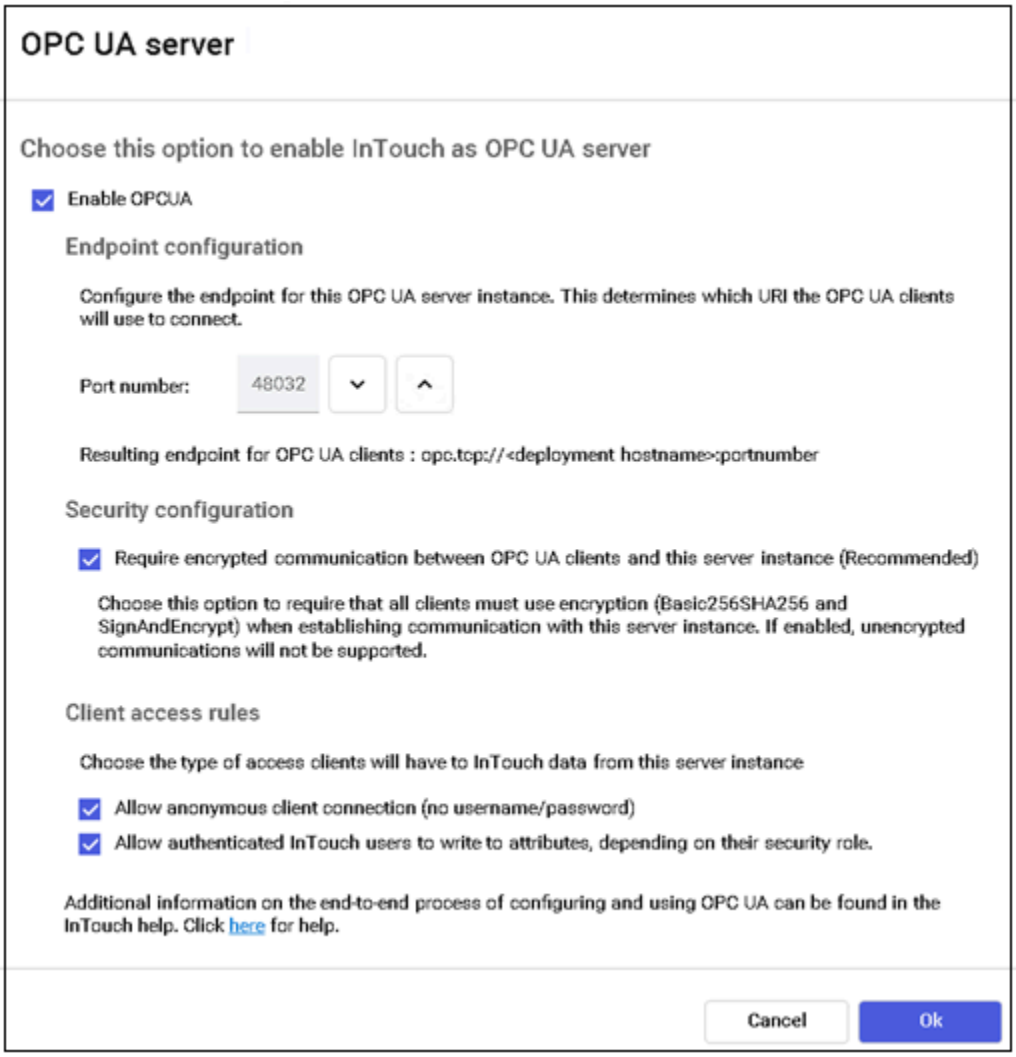
5. **Security Certificate:** Download and configure the OPC UA security certificate on the run-time node.
6. **Validate connectivity:** Open the OPC UA client and verify that you can connect to the OPC UA Server, and can view items in the namespace.

Configure the InTouch OPC UA Server

The InTouch OPC UA Server provides access from an OPC UA client to InTouch HMI data, without the need for a gateway, or other protocol translation mechanism.

1. Launch **AVEVA InTouch HMI Application Manager**.
2. On the Tools tab, select **OPC UA Configuration**.

   The OPC UA Server dialog box appears.
3. The OPC UA server will be disabled by default. To configure the server, select **Enable OPCUA**.



4. Edit the **Port Number**. By default the port will be set to 48032.

Note: The port number is unique for a user and RDS session. Assign alternate port numbers for additional

RDS sessions.

5. **Security Configuration**: It is strongly recommended that you enable this option, as this will encrypt the payloads across the connection. Note that the client must match this configuration.
6. The Client Access Rules options allow you to determine the type of access the client will have to InTouch data.
   a. Select the **Allow anonymous client connection (no username/password)** checkbox to allow anonymous access.
   b. Select the **Allow authenticated InTouch Users to write to attributes, depending on their security role** checkbox to allow only authenticated users.
7. Select **OK**.
8. After configuration, start the WindowViewer to launch the OPC UA Server. OPC UA clients can now access InTouch HMI data.

Configure the firewall for the OPC UA service

OPC UA communications in InTouch HMI require that the run-time node firewall allows a connection with an OPC UA client node. Before changing firewall settings, however, it is recommended that you perform a Firewall test.

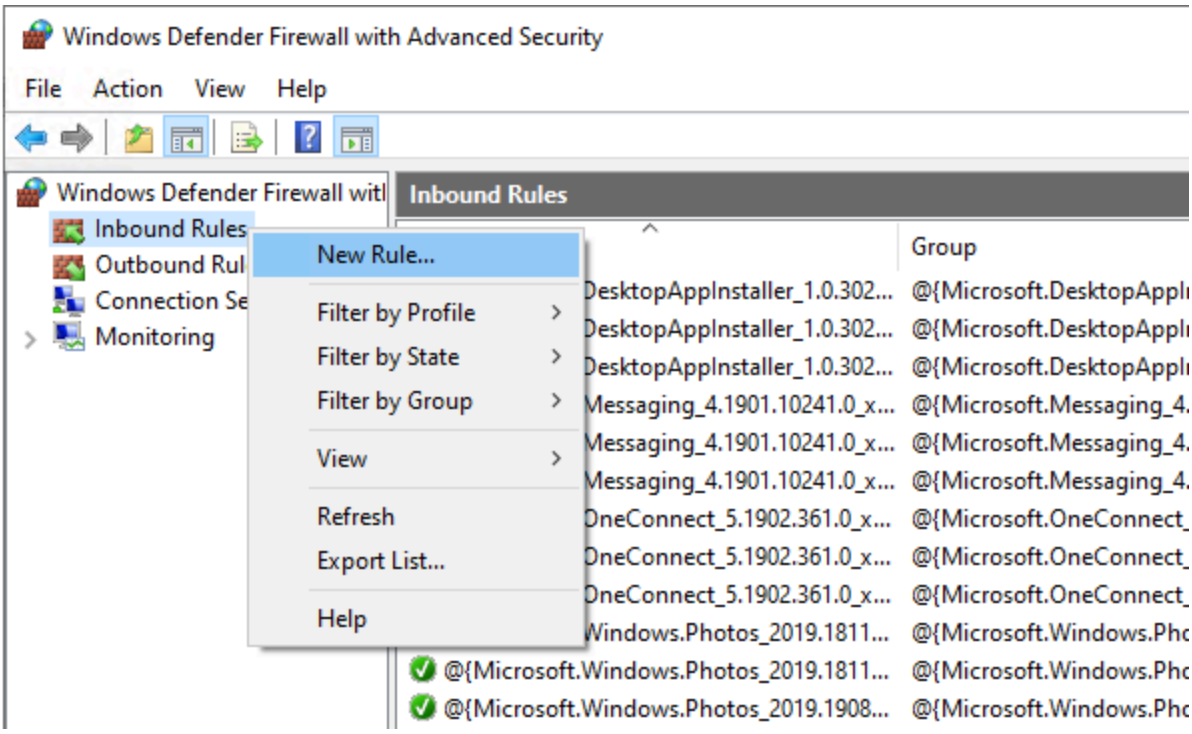
Configure the run-time node firewall

Important: The firewall rules must be added to the node to which the OPC UA Server Service is deployed.

Configure the run-time node firewall

On the run-time node(s) where the OPC UA Server Service is deployed, open the Windows firewall and configure it as follows:

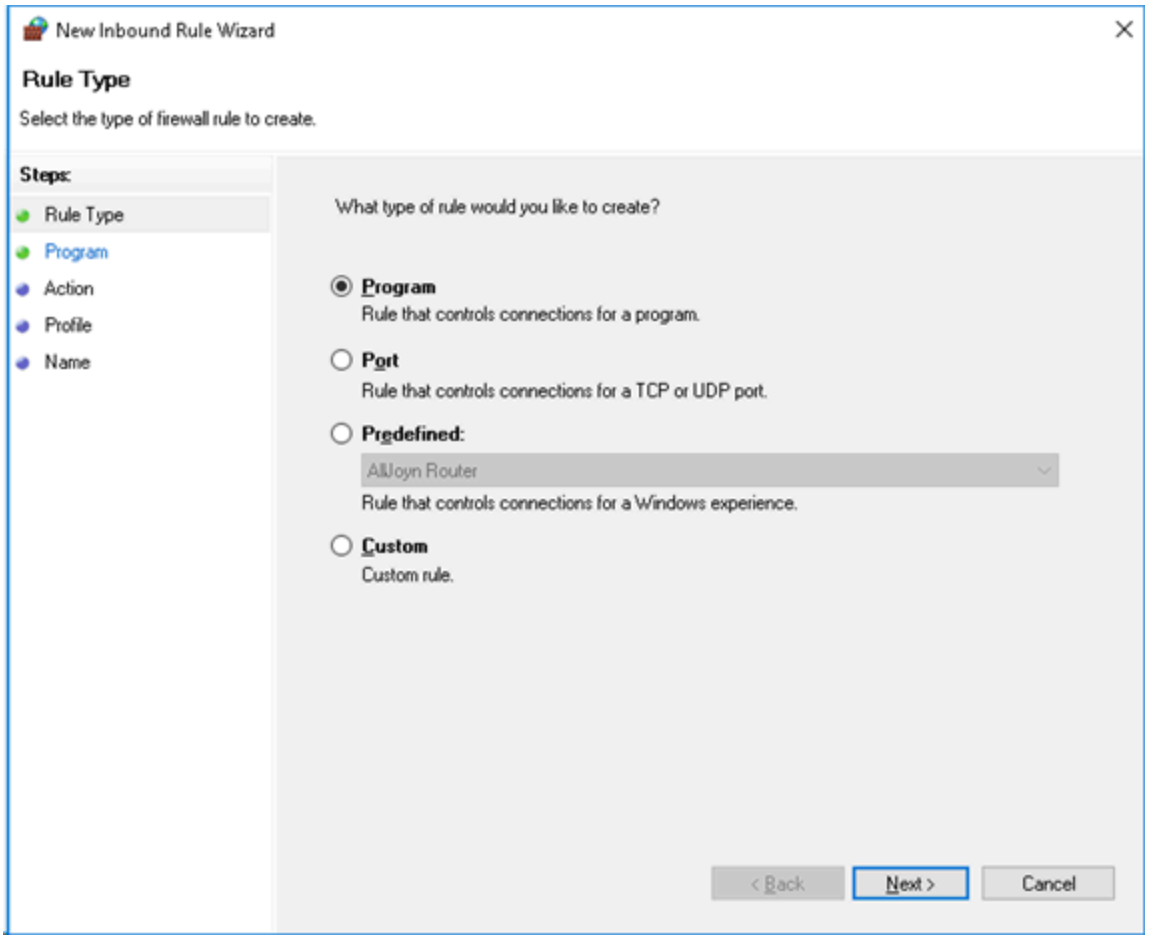
1. In the Windows Search bar, open **Windows Firewall**.
2. Select **Advanced Settings** and create an **Inbound Rule**.



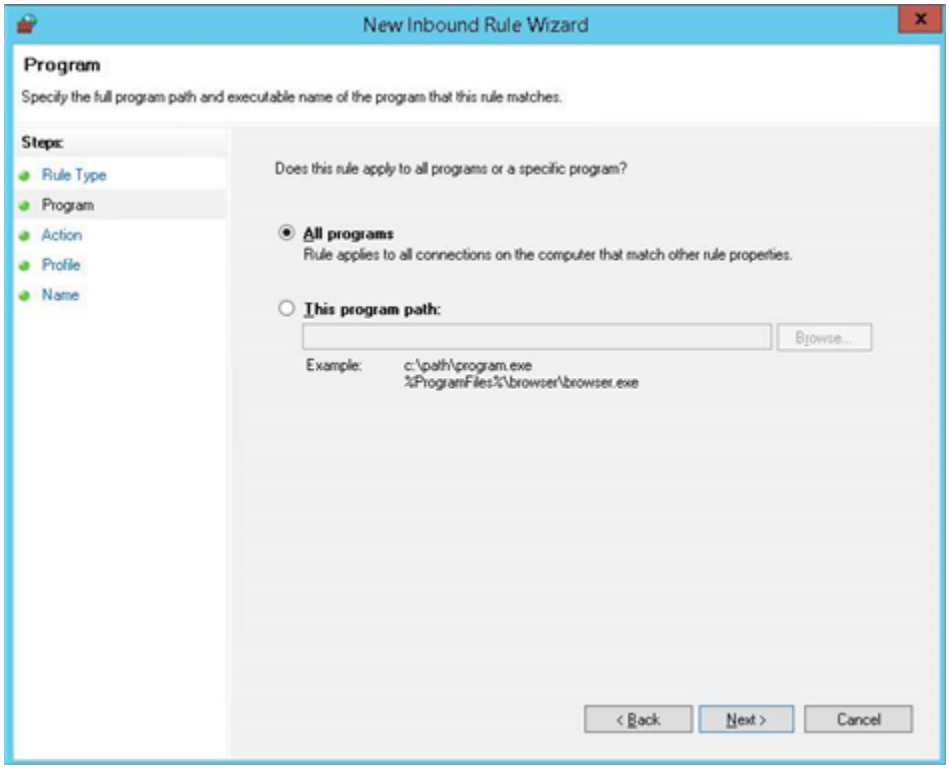
3. Select **New Rule.**

   The **Rule Wizard** opens. .

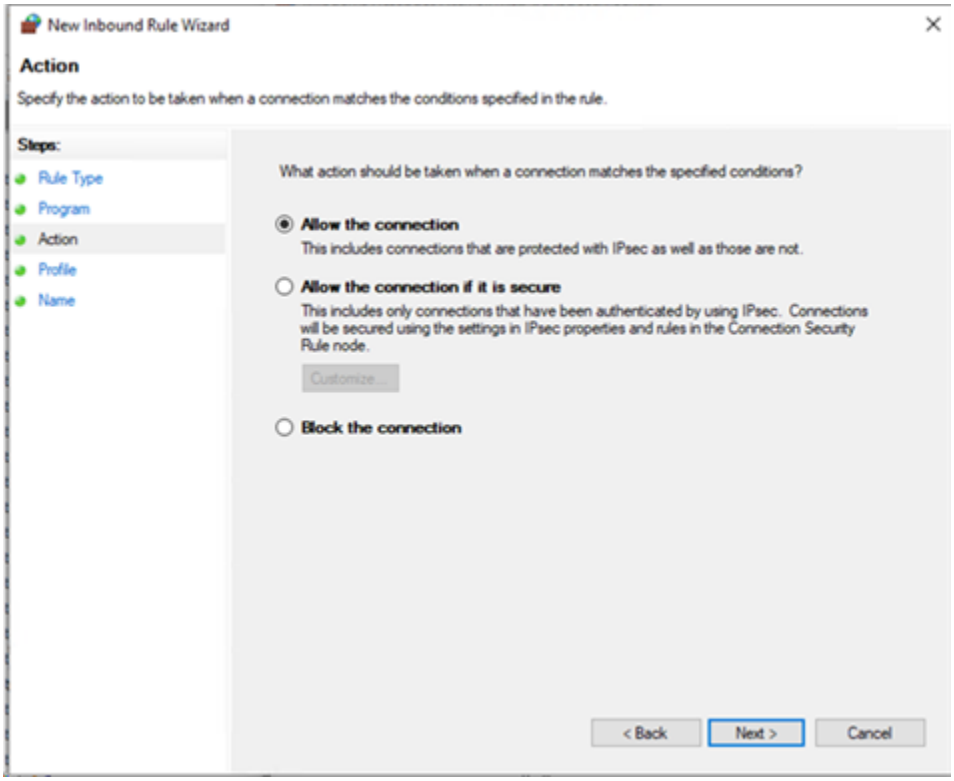
4. Select **Program** for the Rule Type and select **Next**.



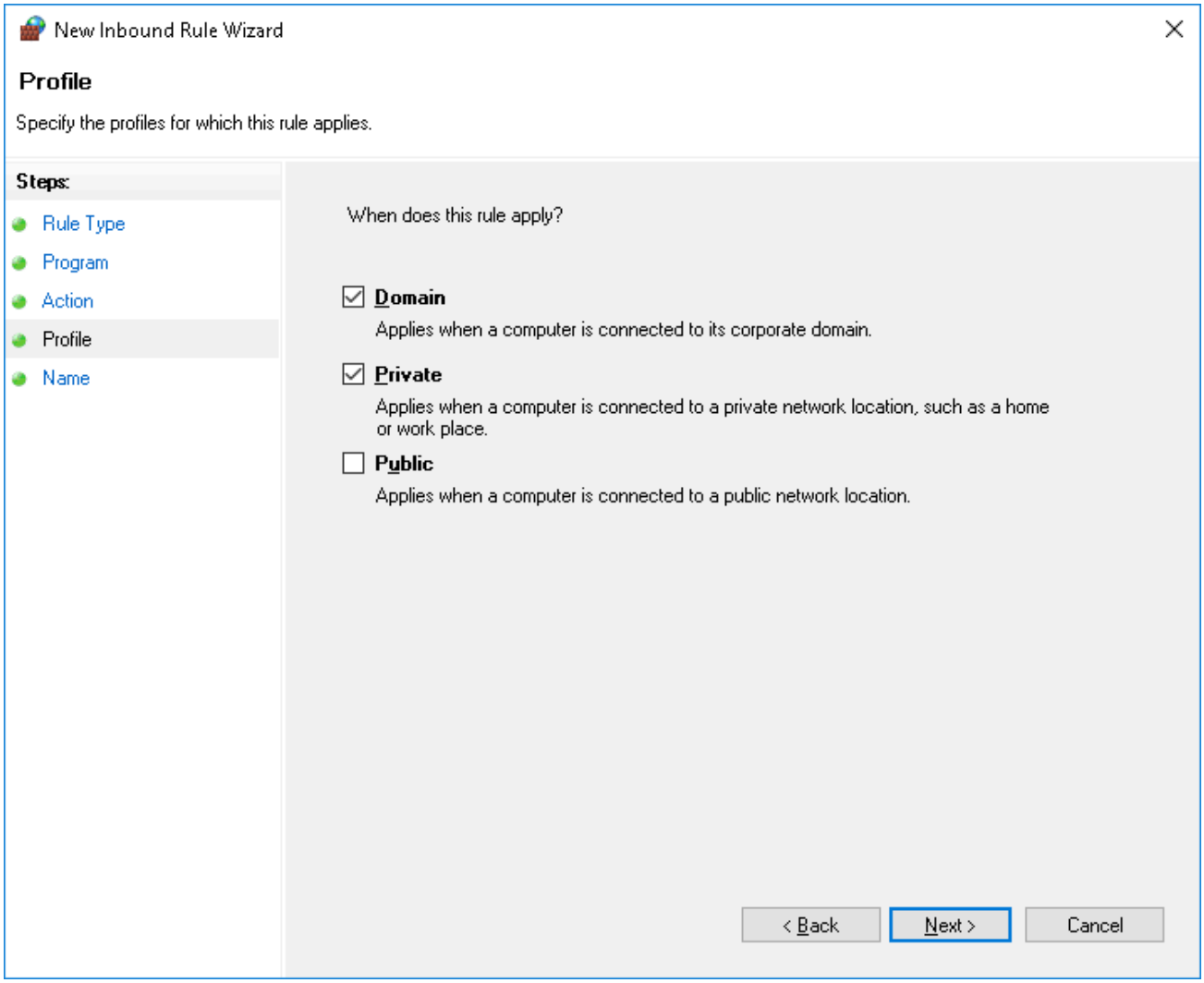
5. Specify the program path for the selected rule.



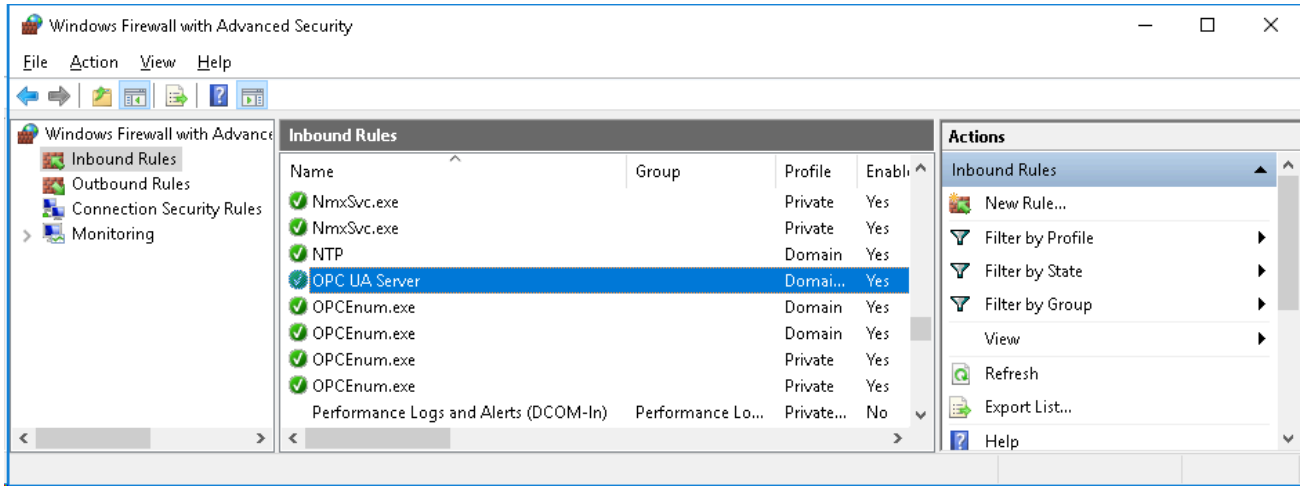
6. On the next screen, select the option "**Allow the connection**." Select **Next**.



7. The wizard will ask when the rule applies.
   - For **Domain** environments: Select **Domain** and **Private**. We recommend that you deselect **Public**.
   - For **Workgroup** environments: Select **Public**. The Domain and Private settings have no affect in a Workgroup environment.



8. Finally, provide a name for this rule (for example, "OPC UA Server"). If you will be configuring multiple OPC UA services, be sure to use names that differentiate each service from the others.
9. Now, check that the new rule has been added to the list of InBound Rules in the Windows Firewall and that it is enabled.



10. Verify that you can connect to the run-time node from the OPC UA client node by repeating the Firewall Test.

Firewall test

Perform a firewall test with Telnet

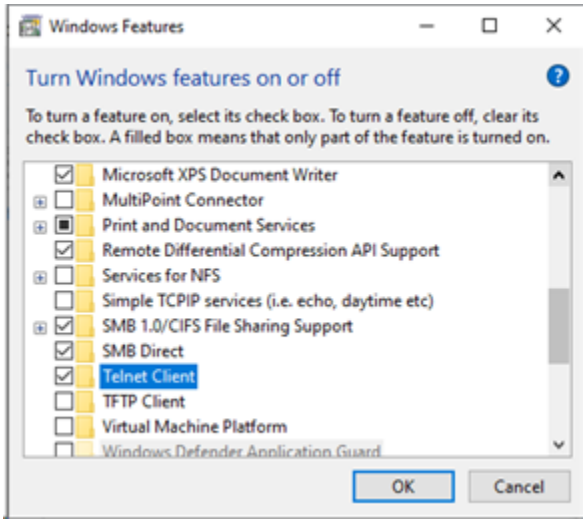
1. From a separate node from where the OPC UA Server service is running, enable **Telnet** by turning on the **Telnet Client** Windows Feature.

   Note: This test is ideally run from the OPC UA client node.

   a. Open the Windows **Control Panel**.

   b. Go to **Programs and Features**, then select **Turn Windows features on or off.**

   c. Scroll down the list of features to **Telnet Client** and enable it. Telnet is disabled by default.



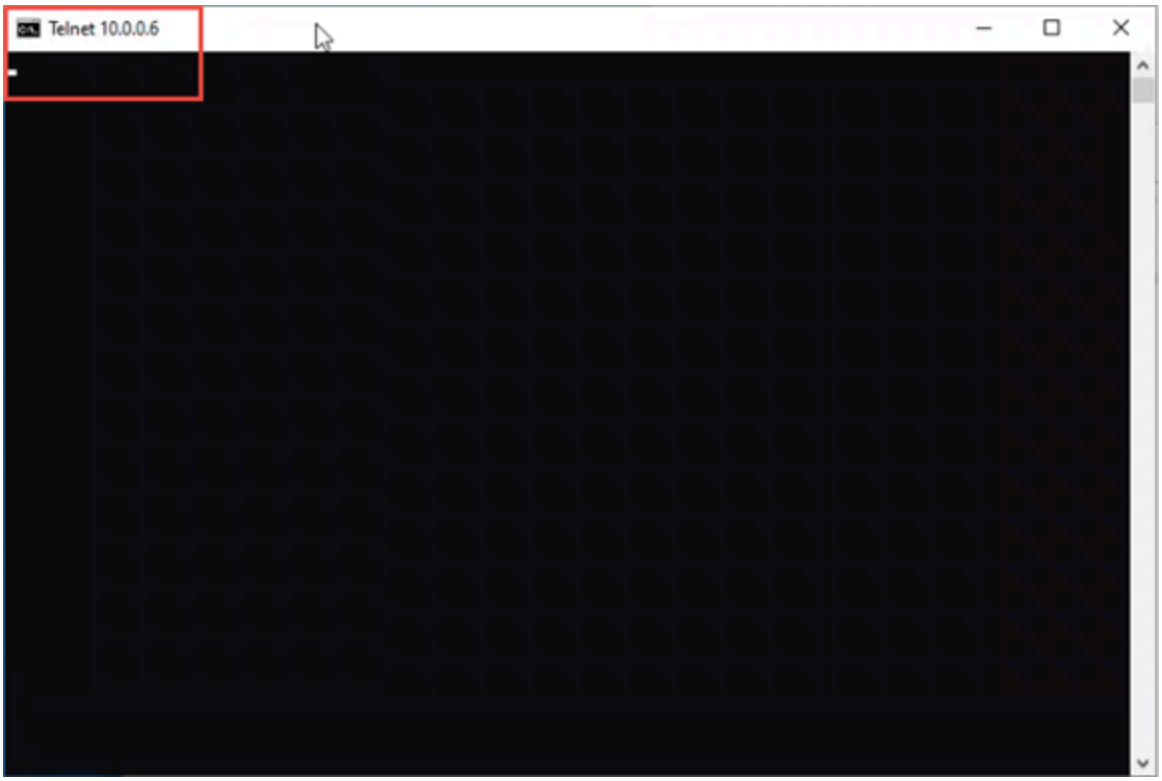
2. Prior to running this test, verify that WindowViewer is running. If the OPC UA Service Host is configured, WindowViewer will start the InTouch OPC UA Service. See Configure the InTouch OPC UA Server for details.
3. Run Telnet in a command window on the OPC UA client node by entering the following:

   telnet <nodeName or ipAddress> <portNumber>

   where

   - **nodeName** is the machine name of the InTouch HMI run-time node. Use nodeName or ipAddress, not both.
   - **ipAddress** is the IP address of the InTouch HMI run-time node. Use nodeName or ipAddress, not both.
   - **portNumber** is the port number you configured in InTouch HMI for the OPC UA Service. The default port number 48032.

     Example: `telnet 10.10.10.06 48031`
   - If the command is not successful, it will time out with a message stating that the connection failed. In this case, go to Configure the run-time node firewall.
   - If the telnet command is successful, the command prompt changes to a Telnet prompt.



- If the Firewall Test is successful, configure the OPC UA client and OPC UA server certificates. The next step in setting up your OPC UA connection depends on if you are using a third-party OPC UA client application, or the OPC UA connection available with Gateway Communication Driver.

Configure server and client certificates for third-party OPC UA client applications

IMPORTANT! These procedures apply ONLY if you are using a third party OPC UA client. If you are using Gateway Communication Driver as the OPC UA client, skip to Use OI Gateway to configure the client security certificate.

To configure encrypted communications between the InTouch OPC UA server and a third-party OPC UA client, both computers will need access the following certificates:

- <Computer Name> ASB OPC UA Server
- The client certificate from your OPC UA client.

To complete this setup, you will need to perform three steps:

1. Copy the certificate from the OPC UA server node to the OPC UA client node. This step includes the following:
   - Exporting the OPC UA server certificate.
   - Installing the certificate on the client node.
2. Copy the certificate from the OPC UA client node to the OPC UA server node.
3. Make sure that the firewall has been configured to allow the InTouch.OPCUA.ServiceHost.exe application.

Export the OPC UA server certificate to the OPC UA client node

Export the OPC UA server certificate

1. Open the Windows Certificate Manager on the OPC UA server node.

   To open the Certificate Manager, either type "`Manage Computer Certificates`" in the Windows search box and select, or open the command prompt and run "`certlm.msc`."
2. In the tree view, expand the **Personal** node, then select on **Certificates**.
3. Locate the certificate "**<computer name> ASB OPC UA Server**".
4. Right-click on the certificate and select 'All Tasks\Export…"

   The **Certificate Export Wizard** opens.
5. Depending on type of certificates that your client application uses, you will need to export the certificate as one of two certificate file types:
   - **.cer file**, if the client uses the Windows Certificate Store
   - **.der file**, if the client uses file-based certificates.

   Even though the file extensions are different, the file formats are the same.
6. In the Certificate Export Wizard, select the following options.
   - **Export Private Key:** select "No, do not export the private key" (default), then select **Next**.
   - **Export File Format:** select either "DER encoded binary X.509 (.CER)" or "Base-64 encoded X.509 (.CER)," depending on the requirements of your client application. Make the selection, then select **Next**.
   - **File to Export:** enter a file name to export the Root CA, for example "`c:\temp\<machine name> OPC UA Server.cer`," then select **Next**.
7. When the **Export Wizard** finishes, you may need to change the file extension of the certificate from ".cer" to ".der," depending on what your client application is expecting. You will need to do this if your OPC UA Client application stores certificates in a specific folder, rather than in the Windows Certificate Store.

Import the OPC UA server certificate on the client computer

To complete the process of copying and installing the OPC UA server certificate to the OPC UA client node, you need to import the OPC UA Server certificate to the OPC UA client computer.

Each OPC UA client application has its own mechanism to manage certificates. Generally, an OPC UA client will use one of two mechanisms to manage certificates:

- Utilizing the Windows Certificate Store
- Storing certificates in a specified folder, defined by the OPC UA Client application.

Refer to the documentation for your OPC UA client applications for more details on how to import a server certificate.

Import a certificate into the OPC UA client windows certificate store

1. Copy the certificate file (<machine name> OPC UA Server.cer) to the OPC UA client node. Where you copy the file is not important.
2. Right click on the certificate file and select **Install Certificate** from the context menu.

This opens the Certificate Import Wizard.

Note: Administrator privileges are required to import the certificate.

3. In the Certificate Import Wizard, select the following options.
   - **Store location:** Select "Local Machine," then select **Next**.
   - **Certificate store:** Browse to "Personal," then select **Next**.
   - **Completing the Certificate Import Wizard:** Review the settings, then select **Finish**.

Copy a certificate to specified location on the OPC UA client

1. Copy the certificate file (<machine name> OPC UA Server.cer) to the folder that is specified by your OPC UA client applications. The following table shows the location of the certificate folder for a number of common OPC UA clients.

OPC UA client	Manufacturer	Folder
Datafeed OPC UA Client	Softing	C:\ProgramData\Softing\OpcClient\pki\trusted\certs
UaExpert	UnifiedAutomation	C:\Users\Admin\AppData\Roaming\unifiedautomation\uaexpert\PKI\trusted\certs
UA Client Getting Started	UnifiedAutomation	C:\ProgramData\unifiedautomation\UaSdkNetBundleEval\pkiclient\trusted\certs
Matrikon	Matrikon	C:\Users\Admin\Appdata\Local\Matrikon\OPCUAExplorer\pki\DefaultApplicationGroup\trusted\certs
KEPServer	Kepware	C:\ProgramData\Kepware\KEPServerEX\V6\UA\Client Driver\cert
Top Server	Software Toolbox	C:\ProgramData\Software Toolbox\TOP Server\V6\UA\Client Driver\cert

Refer to the documentation for the OPC UA client application for additional information about managing certificates.

2. Configure the OPC UA certificate, as described below.

Configure OPC UA client certificates on the OPC UA server

The next step in the of configuring OPC UA server and client certificates is to trust the OPC UA client certificate that has been installed on the OPC UA server node.

The easiest way is to attempt to connect an OPC UA client to the server. Since trust of the client certificate has not yet been established, the connection is expected to fail.

Once the connection fails, any OPC UA client certificates that are not installed on the OPC UA server are placed in the “Rejected Certificate” folder on the OPC UA Server.

By default, the folder location is:

C:\ProgramData\AVEVA\PCS\OPC UA Rejected Client Certificates\certs

Note: Access to this folder requires administrator rights, and the folder is hidden by default.

Import certificates placed in the Rejected Certificate folder

1. To import the OPC UA Client certificates, browse to the rejected certificate folder.
2. Right click on the certificate for the OPC UA Client that you want to trust and select “Install Certificate”. This opens the **Import certificate Wizard**.
3. Select the following options in the wizard:
   - **Store location:** Select "Local Machine," then select **Next**.
   - **Certificate store:** Select “Trusted People,” then select **Next**.
   - Completing the Certificate Import Wizard: Review the settings, the select **Finish**.

Port usage

OPC UA communicates via a single TCP port. This is specified in the **Endpoint Connection** setting in the configuration of the OPC UA Server, and defaults to port 48032. For details, see Configure the InTouch OPC UA Server.

If you will run multiple OPC UA Servers on the same computer using RDS sessions, you must specify a different port number for each subsequent server.

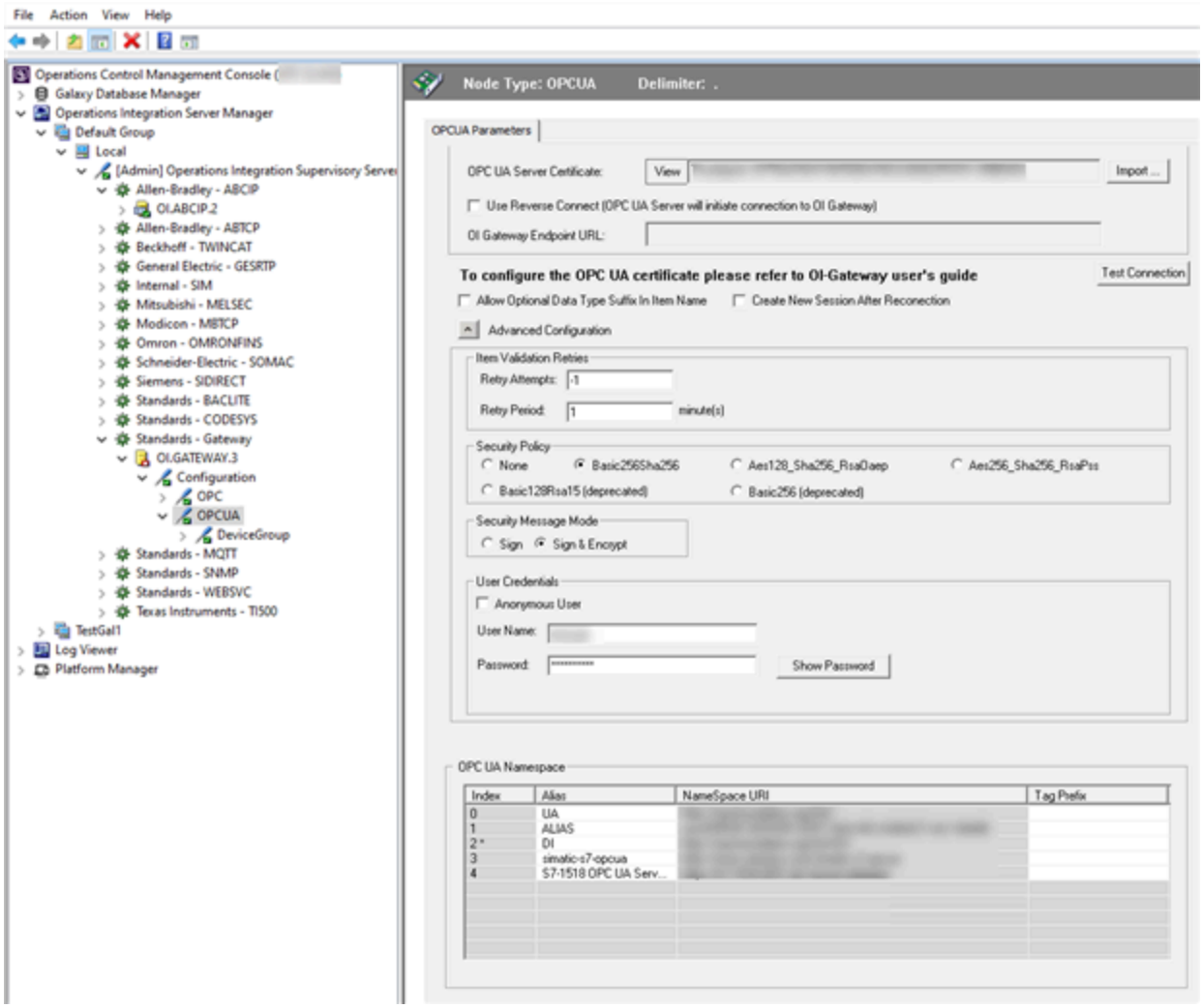
Confirm that this port is not blocked by any firewall software installed on your computer. For information about testing and configuring communications through firewall, see Configure the firewall for the OPC UA service.

Use OI Gateway to configure the client security certificate

Gateway Communication Driver provides a convenient method for configuring security certification on the server and client OPC UA nodes.

Configure the security certificate through Gateway Communication Driver

1. From the Start Menu on the run-time node, open the Operations Control Management Console. (**Start > AVEVA > Operations Control Management Console)**



2. In the console tree, navigate to the **OI.GATEWAY.3** node under Operations Integration Supervisory Servers.
3. Create an OPC UA connection.
4. Configure OPC UA Server Details.
   - **Server Node:** Enter the machine name of the run-time node.
   - **OPC UA Server:** This is the URI (uniform resource identifier) for the OPC UA server (the run-time node). The address must be entered manually because it is not currently discoverable. Enter it in the format opc.tcp://<machine name>:<OPC UA port number>

     Use the OPC UA port number that you entered when configuring the InTouch OPC UA Server in InTouch HMI Application Manager. The default port number is 48032.
5. Enter the authorization and authentication credentials.

   You must match the authorization settings configured in the OPC UA server dialog. If the Require Security Authentication checkbox is checked, then you must select the following settings:
   - **Security Policy:** Basic256Sha256.
   - **Security Message Mode:** Sign and Encrypt.
   - Under User Credentials, select **Anonymous User** to allow anonymous access. You can also provide user credentials of authenticated users if the corresponding option was selected during OPC UA configuration. The user credentials provided must be part of the InTouchHMIOPCUAWriteUsers user

group.

6. Select the **Test** button. The test will fail, but it will download the OPC UA certificate.

   IMPORTANT! The reason for this initial test failure is because the certificates between the client and server applications must be trusted. Installing the certificates will fix this.

7. Go to the next section.

Once the certificates are trusted, the OPC UA client configuration will need to be validated.

Trust the certificate between the OPC UA server and OPC UA client

In this release of the OPC UA Server service, the operation of creating the trust between the OPC UA Server and the OPC UA client must be done manually. The **Test** operation causes the Gateway Communication Driver to submit its own certificate to the OPC UA Server node so it can be trusted. The following steps show how to then trust the client certificate from the OPC UA Server node. If you are not using Gateway Communication Driver, follow the procedures listed in Configure server and client certificates for third-party OPC UA client applications.

1. Access the folder **C:\ProgramData\AVEVA\PCS\OPC UA Rejected Client Certificates**.

   This is the location where the certificate from the client is initially placed by default as an attempt to connect to the OPC UA Server.

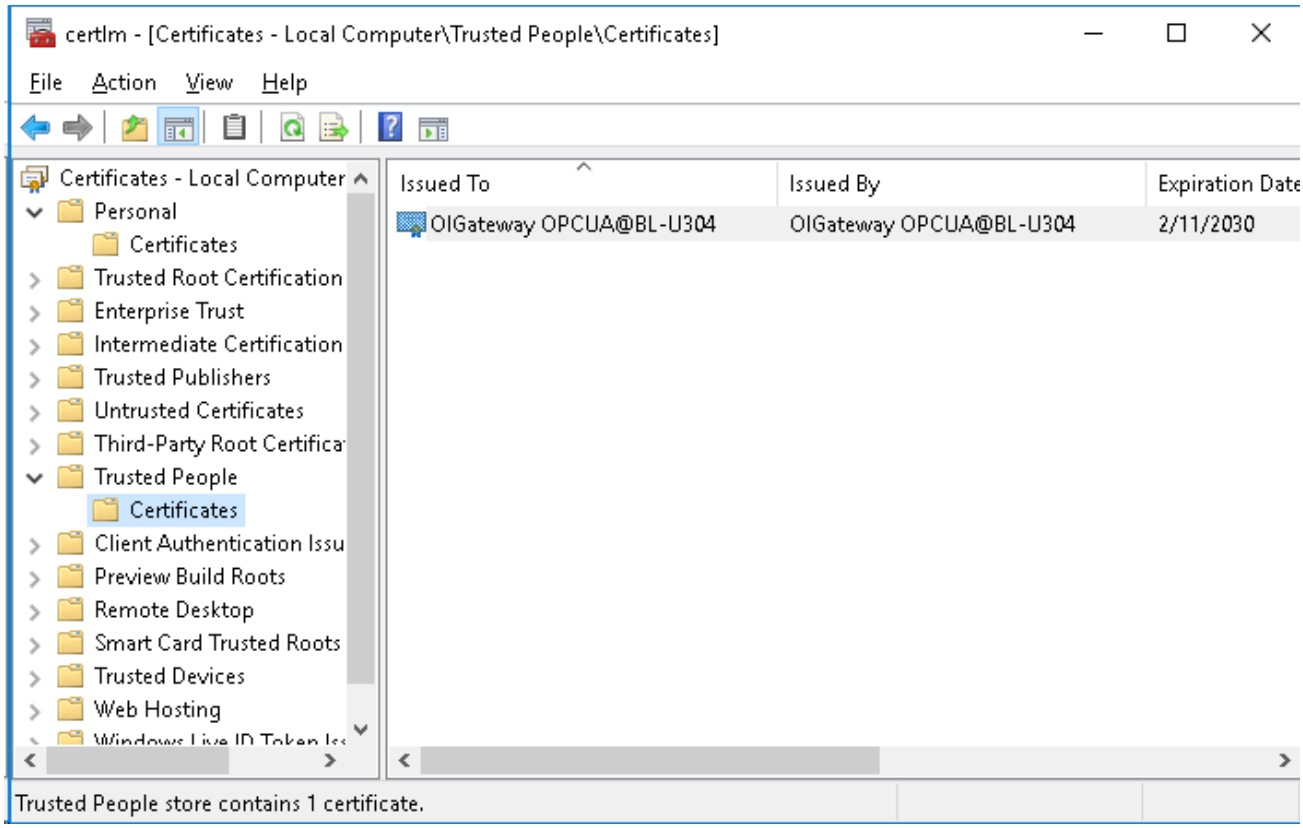
   Note: The ProgramData folder is hidden by default. You may need to enable the hidden items option in Windows Explorer in order to view it.

2. Right-click on the certificate name, for example, **OIGatewayOPC UA@OPCUA client node{long hex ID}.der**.
3. Select **Install Certificate** from the context menu. This opens the **Certificate Import Wizard**.
4. Select **Local Machine** for the Store Location, then select **Next**.
5. From the **Select Certificate Store** list, select **Trusted People** as the certificate store. This is the only choice that will work with the OPC UA certificate.
6. Close the wizard to complete installation.
7. Finally, delete the certificate from the **OPC UA Rejected Client Certificates** folder, since the certificate is now installed.

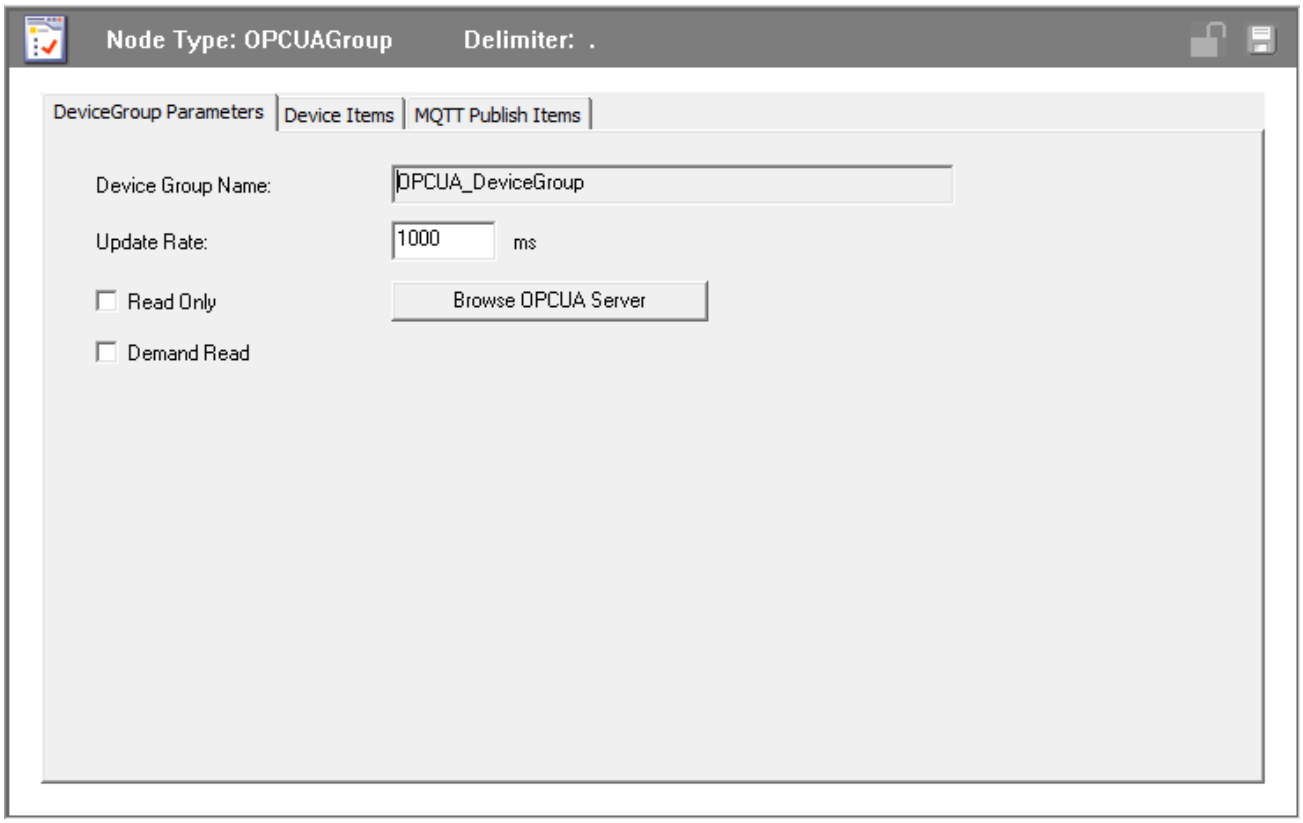
Verify OPC UA certificate installation

Verify certificate installation

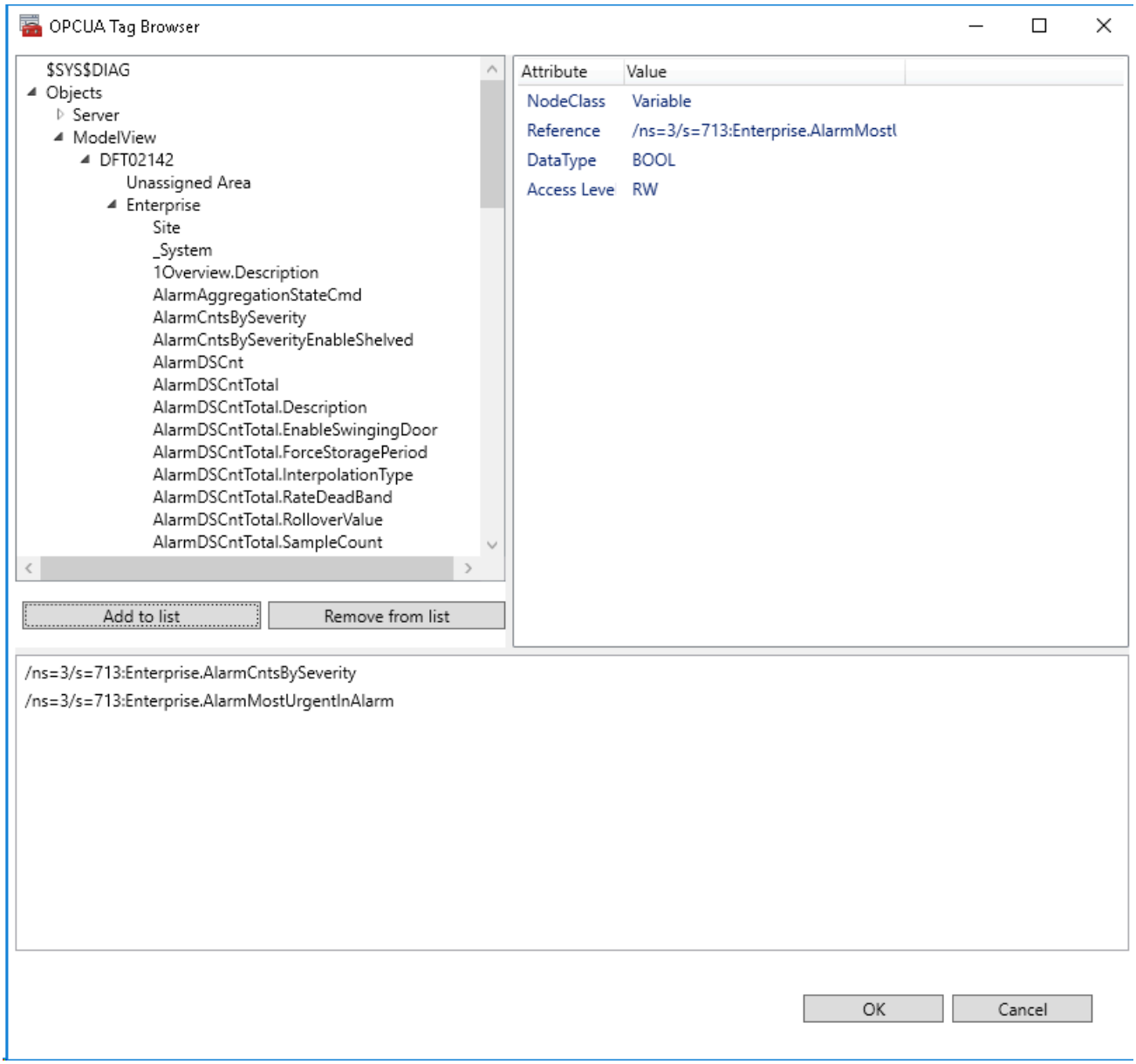
1. Open the certificate manager by typing **certificate** in the Windows search window, then select **Manage Computer Certificates** from the search results. The **Certificate Manager** opens.
2. Navigate the **Trusted People** folder and check that the OPC UA certificate has been installed.



3. From the OPC UA client node, open the **Operations Control Management Console** again, and select the **Test Connection** button in OPC UA parameters window. At the bottom of the window, the **OPC UA Namespace** alias list will be automatically populated, indicating the connection was successful.
4. Add a group under the OPC UA Connection. The device group name prefixes **OPCUA_** to the group name you enter.



5. Select the **Browse OPC UA Server** button to browse the OPC UA hierarchy.
6. Optional: Add OPC UA tags to monitor in the alias list to help ensure that you can browse the namespace.



7. Switch back to OI Gateway configuration in the **Operations Control Management Console**, and notice that the OPC UA tags are now listed in the **Device Items** window.
8. By default, item names in the list duplicate the complete item reference path. Rename the items as needed to be more user-friendly.

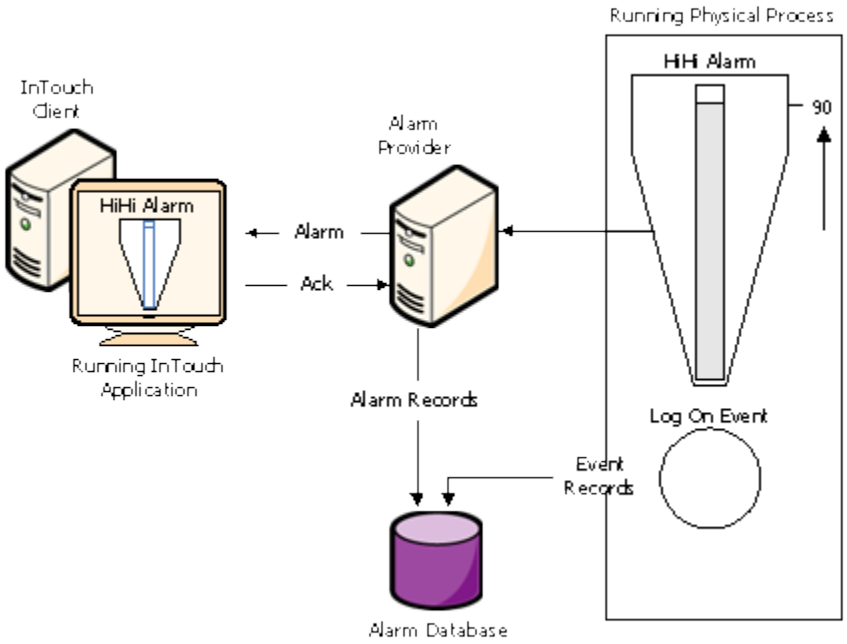
Alarms

You can create InTouch applications that generate alarms and events to notify operators about the status of process activity.

- Alarms warn run-time operators about process conditions that could potentially cause problems. Typically, you set up an alarm to trigger when a process value exceeds a defined limit. An operator must usually acknowledge the alarm.

- Events represent normal system status messages. A typical event is when a system condition occurs, such as an operator logging on to an InTouch application. Operators do not have to acknowledge events.

The following figure shows how the InTouch HMI handles alarms and events while an application is running. Alarm and event data is saved to the alarm database.



You can configure any tag for event monitoring. An event message is logged to the alarm system each time the tag value changes. The event message includes how the value changed and whether the operator, I/O, scripts, or the system initiated the change.

About InTouch alarms

Alarms represent warnings of process conditions that could cause problems and require an operator response. A typical alarm is triggered when a process value exceeds a user-defined limit, such as an analog value exceeding an upper threshold. This triggers an alarm to notify the operator of a problem. After the operator acknowledges the alarm, the InTouch HMI recognizes the alarm has been acknowledged.

You can configure the InTouch HMI to require an alarm to be acknowledged even if the condition causing the alarm has passed. This ensures that an operator is aware of events that caused a temporary alarm state but have returned to normal.

The main alarm states are described in the following table:

Alarm State	Condition
ACK	Alarm was acknowledged.
ALM	Alarm has occurred.
RTN	Tag returned from an alarm state to a normal state.

Alarm State	Condition
LATCHED	Alarm is acknowledged from "UNACK_RTN" state or alarm is returned from "ACK" state.

Alarm priorities

You assign a level of priority, or severity, to an alarm. A boiler temperature limit, for example, would require a high-priority alarm, requiring immediate attention. An end of a shift alarm is a much less severe. The alarm priority usually depends upon the circumstances—the factory application, the nature of the equipment, safety, availability of backup systems, potential costs of damage, or downtime.

You assign an alarm priority when you define a tag. The priority can range from 1 to 999, with 1 being the most severe.

You can designate a range of alarm priorities to represent a classification of alarms. For example, if a process requires four levels of severity, you can create four priority ranges.

Alarm Severity	Priority Range
Critical	1 – 249
Major	250 – 499
Minor	500 – 749
Informational	750 – 999

Ranges are useful for alarm filtering. For example, you can configure an alarm display to filter out all but the critical alarms. You can create animation links, acknowledgment scripts, and filtered viewing and printing, all based on the alarm priority range.

Alarm sub-states

A multi-state alarm includes a range of alarm sub-conditions.
For example, an analog alarm typically has several limits.

- A High and Low threshold set the boundaries for the normal operating range.
- HiHi and LoLo limits mark the extreme deviations from the normal range of values.

A boiler temperature level can be in the alarmed condition for any one of these sub-states. The boiler temperature can also transition between any two sub-states while continuing to remain in the overall alarmed condition.

Alarm acknowledgement

When an alarm occurs, the run time operator (or system) must acknowledge the alarm. Acknowledgement merely indicates that someone is aware of the alarm. This is separate from taking corrective action, which might not happen right away. It is also separate from whether the alarm condition returns to normal—which it might

do on its own, even without any external intervention.

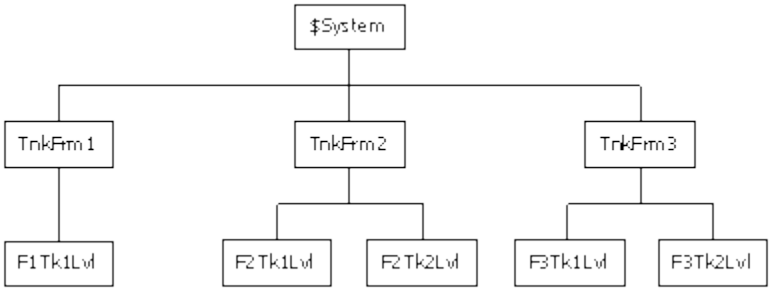
A high or medium priority alarm usually requires immediate acknowledgment, while a very low-priority alarm might not. Although the condition that generated the alarm may go away (for example, a temperature rises too high and then becomes lower again), the alarm itself is not considered resolved until it is acknowledged.

Once the alarm is acknowledged, the operator can dismiss the alarms. With latching enabled, an acknowledged alarm that has returned to normal condition will be placed in the LATCHED alarm state. The operator can dismiss the LATCHED alarms to remove the LATCHED alarms from the current mode of the Alarm Client Control grid, but the alarms would still be visible in the recent mode of the Alarm Client Control.

Alarm groups

You can group alarms to make tracking and management easier. Alarm groups are logical representations of different areas of a factory, pieces of equipment, operator responsibility, or a manufacturing process.

For example, the following figure shows a three-tier alarm group hierarchy for a tank farm application.



Alarm groups are useful for filtering in alarm displays, alarm printers, and acknowledgment scripts.

Every tag is associated with an alarm group. By default, tags are assigned to main $System group. You can create a hierarchy of additional alarm groups under the $System group, up to a maximum of 32 levels.

You create alarm groups and associate tags with them while you are defining your tags in the Tagname Dictionary.

Alarm groups and group variables are not compatible with SmartSymbols. You can't use references to alarm groups or group variables in a SmartSymbol.

About InTouch events

An event is a detectable occurrence of something happening within the system, which may or may not be associated with an alarm. A transition into or out of an alarmed state is one kind of event. An event might also be an operator action, a change to the system configuration, or some kind of system error.

An event is different than a condition. A condition can persist for minutes, hours, days, or weeks. An event is momentary; it takes place and is immediately over. An alarm is a condition; an alarm notification is an event.

Events represent normal system status messages and do not require an operator response.

When you define a tag to do event monitoring, you can choose to have event messages printed or logged to the alarm system each time the tag value changes. The event message includes how the value changed and whether the operator, I/O, scripts or the system initiated the change.

An event can be one of the following types:

Event	Condition
OPR	The operator modified the tag value using the Value input.
LGC	A QuickScript modified the tag value.
DDE	The tag value was poked from a DDE client.
PROT	The tag value change was initiated from an Industrial graphic, either from a custom property or from a pushbutton in the Industrial graphic.
SYS	A system event occurred.
USER	$Operator changed.

The SYS and USER events are generated by the system regardless of whether event logging is enabled for any tags. DDE, OPR, and LGC events relate to tag values and are only generated for tags that have event logging enabled.

Types of InTouch alarms

Within the InTouch HMI, alarms are classified into general categories based on their characteristics. These categories are known as Class and Type. The Distributed Alarm system categorizes all alarms into five general conditions: Discrete, Value, Deviation, Rate-of-Change, and SPC.

Alarm Condition	Distributed Class	Distributed Type
Discrete	DSC	DSC
Value - LoLo	VALUE	LOLO
Value - Low	VALUE	LO
Value - High	VALUE	HI
Value - HiHi	VALUE	HIHI
Deviation - Major	DEV	MAJDEV
Deviation - Minor	DEV	MINDEV
Rate-of-Change	ROC	ROC
SPC	SPC	SPC

You associate each InTouch tag with an alarm condition when you define the tag. Depending upon a tag's type, you can define one or more of the alarm classes or types for it.

Discrete alarms

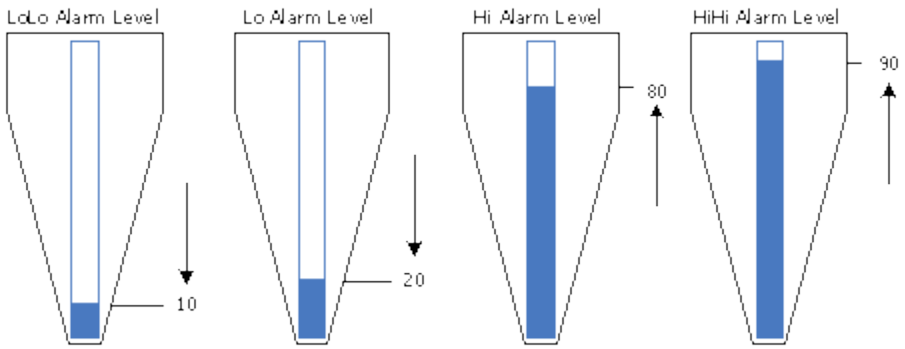
A discrete alarm corresponds to a discrete tag with two possible states. When you create a discrete tag, you configure whether the alarmed state corresponds to the true or false state of the tag.

Analog alarms

An analog alarm corresponds to an analog tag, which is associated with an integer or real number. Within the analog alarm type, there are several sub-types: value, deviation, and rate-of-change.

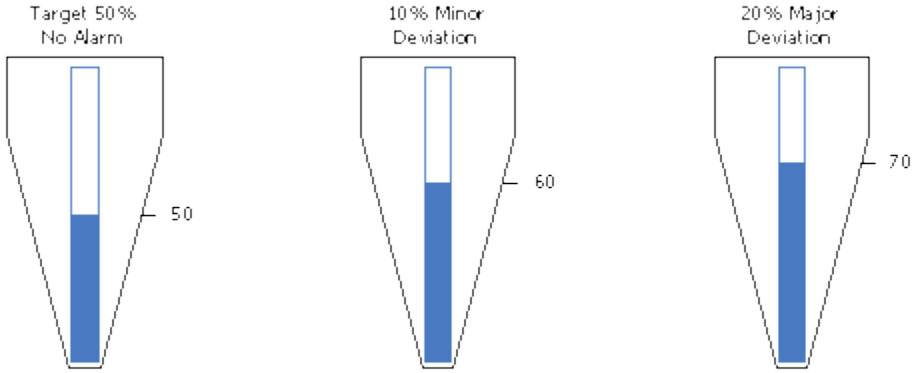
Value alarms

The current tag value is compared to one or more predetermined limits. If the value exceeds the limit, the alarmed state is declared. You can individually configure values and priorities for the "LoLo" limit, "Lo" limit, "Hi" limit, and "HiHi" limit, and indicate whether or not each limit is to be used.



Deviation alarms

The current tag value is compared to a target value, and then the absolute value of the difference is compared to one or more limits, expressed as a percent of the range of the tag value.



You can individually configure values and priorities for the minor deviation limit and the major deviation limit, and indicate whether or not each limit is to be used. You can also configure a value for a deviation deadband, also expressed as a percent of the tag's range. This controls the percentage (of the total range) that the tag value must change before it is evaluated to be in alarm.

For example, you configure limits as follows:

- Range of 0 to 100
- Target of 50
- Minor deviation of 10 percent, which sets the minor deviation thresholds at 40 and 60.
- Major deviation of 20 percent, which sets the major deviation thresholds at 30 and 70.
- Deadband of 10 percent

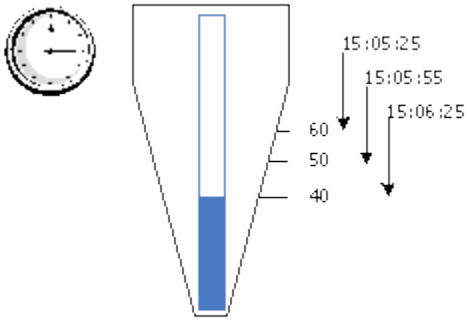
If the tag value is 39, a minor deviation alarm occurs. However, the value must change at least to 50 (40 plus the deadband of 10) before the alarm is evaluated and cleared.

If the tag value is 72, a major deviation alarm occurs. The value drop to at least 61 before the alarm is cleared.

Rate of change alarms

A tag's current and previous values are compared over a measured period. The rate of change of a tag's value is calculated using the previous tag value, the time of the previous change, the current tag value, and the current time.

A tag is tested for a rate-of-change alarm whenever its value changes.The one exception is the initial tag value when an application starts running in WindowViewer. In that case, the initial value is ignored and the first rate of change comparison is made between the second and third changes in a tag's value. Thereafter, rate of change measurements are made between consecutive tag value changes.



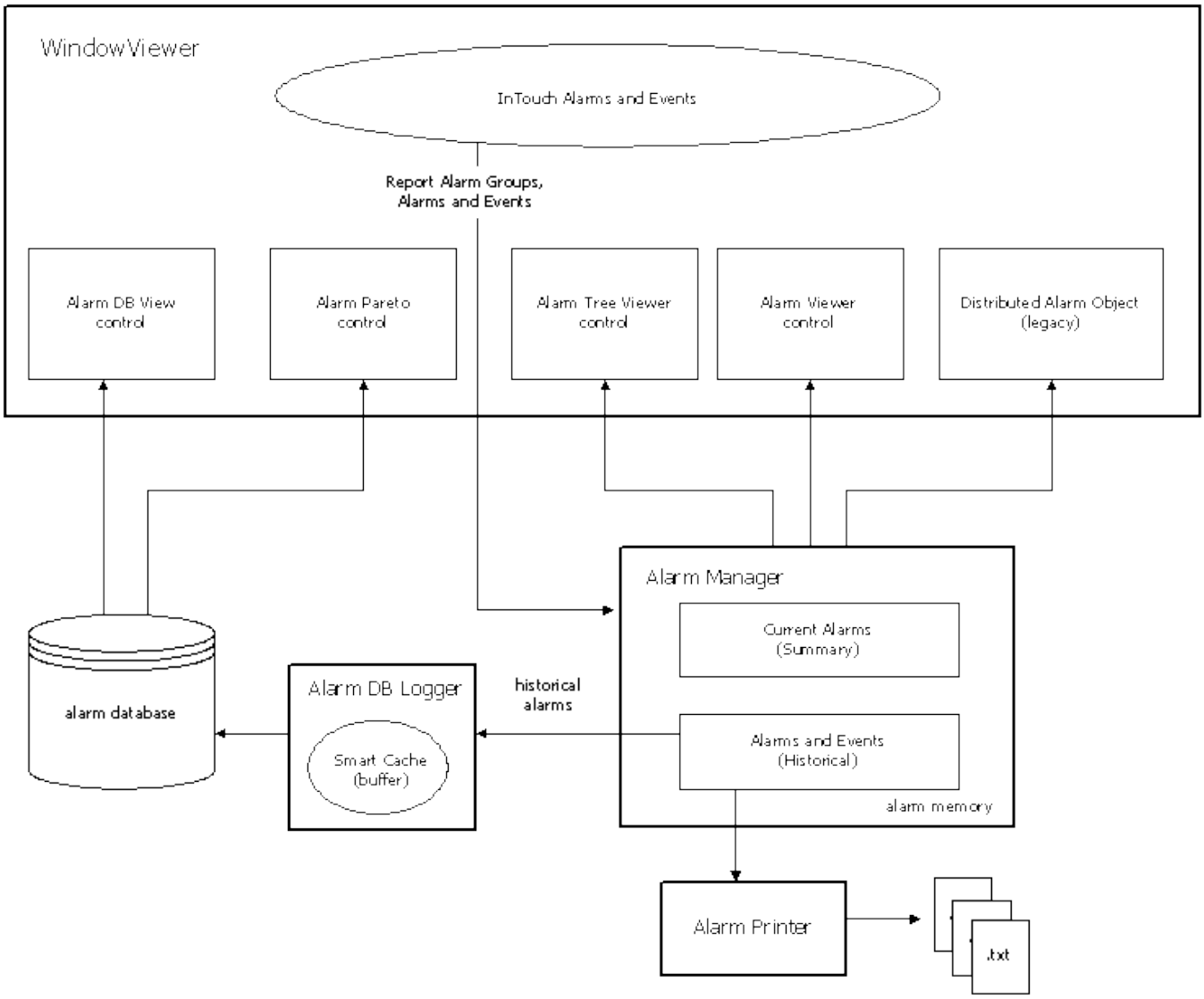
If the absolute difference in values between consecutive tag changes exceeds a specified limit, a rate of change alarm occurs. The rate of change limit is expressed as a percentage of the tag's value over a time interval, which can be per second, per minute, or per hour. You can configure the value and priority of the ROC limit, and whether or not the limit is to be used.

InTouch distributed alarm system

The Distributed Alarm system is made up of:

- An Alarm Manager, which manages currently active alarms (summary alarms) and historical alarms and events. The summary and historical alarms are held in the InTouch internal alarm memory.
- An Alarm DB Logger, which stores historical alarms and events to the alarm database. The alarm database is a SQL Server database.
- An Alarm Printer, which prints historical alarms and events.
- A set of ActiveX controls, which retrieve alarms and events from either the internal alarm memory or the alarm database at run time.

The following figure shows an overview of the system.



Important: The Distributed Alarm system runs as a set of Windows services. To reduce the security exposure of running the Distributed Alarm system with administrator privileges, the user account permissions have been set to non-interactive for these services.

Run time operators can use the Distributed Alarm system to:

- Show, log, and print alarms and events generated by a local InTouch application and by alarm systems of other networked applications.
- Acknowledge alarms locally or from a remote network node.
- Use the alarm ActiveX controls in your InTouch applications to show alarm displays that you pre-configure.
- Provide more feedback about the alarm using a separate alarm comment field.

As an application developer, you can:

- Control the alarms through dotfields.
- Configure your alarms so that they are enabled or disabled directly or indirectly under full control of the

application. You can apply alarm suppression to single alarm classes, tags, or groups to prohibit the showing of alarm information on a specific view node. System-wide disablement can block alarm activity at the source.

- Configure alarm information to be logged to history. The Alarm DB Logger can run as a Windows service or be manually started on demand. Alarm logging uses UTC (GMT) time stamping and provides compatibility with DST and across time zones.
- Set up failover alarm providers. If a primary alarm provider fails, the Distributed Alarm system seamlessly acquires alarm information from the backup system. On reconnection of the primary node, the Distributed Alarm system ensures that alarm acknowledgements are re-synchronized prior to the returning primary system becoming live.

The Distributed Alarm system:

- Sends data through the SuiteLink protocol and uses a minimal amount of CPU and network resources.
- Time stamps the alarm at the time the alarm occurs, not when the consumer receives the alarm. The time stamp includes milliseconds.

Alarm providers and consumers

On any given node there can be a collection of Alarm Providers (Publishers) and Alarm Consumers (Subscribers). The InTouch Distributed Alarm system provides the communication link to pass alarm information between nodes and software components.

Alarm provider

An alarm provider:

- Keeps track of alarmable items—that is, items that can transition into an alarmed condition—and provides the Distributed Alarm system with the list of these items, including information on any hierarchical grouping of the items.
- Notifies the Distributed Alarm system when the status of an alarm item changes. Status changes include whether the item is in or out of the alarmed state and whether the most recent alarm has been acknowledged.
- Keeps track of whether an alarm item is disabled.

The InTouch HMI supports external alarm providers, such as QI Analyst, an Application Server Galaxy, and other software built with the Alarm API toolkit. The date/time stamp for these alarm records is provided by the alarm provider, and is not generated by the Distributed Alarm system.

Alarm consumer

An alarm consumer:

- Provides the Distributed Alarm system with a set of queries identifying alarmable items about which it wishes to receive notifications. A query remains active until changed or removed by the alarm consumer, and specifies an alarm provider or group of alarms - much like a SQL query with "wildcards." Whenever an alarm

provider issues notification of a change, the Distributed Alarm system checks the alarm for matches with all registered queries and passes updates to the corresponding alarm consumers.

- Upon receiving updates, shows or logs information relating to the status of the items or their transitions.
- Acknowledges alarms. The alarm consumer sends an acknowledgement notification to the Distributed Alarm system, identifying the alarm and the alarm provider. The notification is passed to the alarm provider, which then updates the status of the item to acknowledged (if appropriate) and in turn notifies the Distributed Alarm system, thereby ensuring that the update gets distributed to all interested alarm consumers.

Note: The majority of communication in the Distributed Alarm system consists of sending alarm queries and alarm records from one node to another. Within a node, alarm queries and alarm records are tracked and buffered by the internal alarm memory to minimize network traffic.

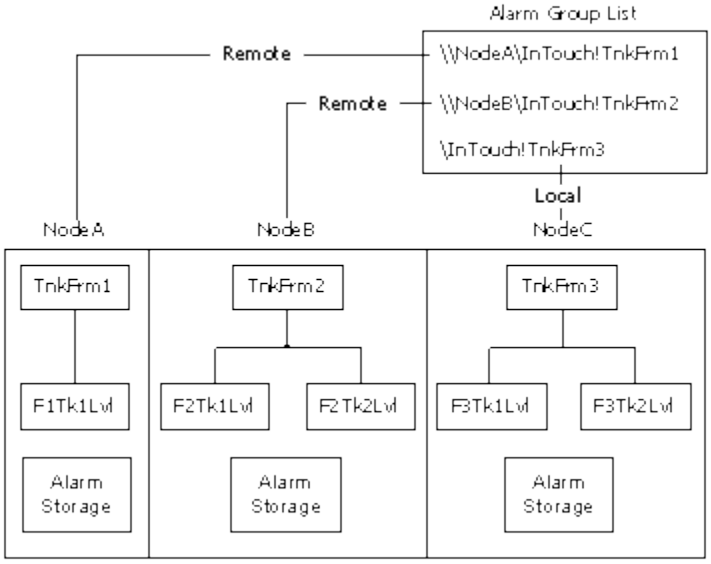
Distributed alarm group lists

The Distributed Alarm system uses alarm groups to organize alarms into a local tree view. The distributed alarm display uses the tree view to filter alarms. You can view these alarm groups from multiple nodes on a network.

The Distributed Alarm system uses an alarm group list to combine alarm groups from local and remote nodes. An alarm group list is a named list consisting of InTouch nodes and the alarm groups defined on each of those nodes. It can also contain other alarm group list names and local alarm groups. An alarm consumer, such as the Alarm Viewer control, uses this list to query for alarms.

Under a query alias, the Alarm Viewer control can show the combined alarms from the groups that belong to the list. Alarms can be acknowledged on the local InTouch node or from a remote node on the network.

The following figure shows an alarm group list that combines alarm groups from three nodes. The alarm group list is defined locally on NodeC. The remaining alarm groups are from remote nodes.



The Distributed Alarm Display shows the alarms resulting from a query across all of the alarm groups that belong to the list. For example, if you were interested in showing all tank farm alarms across several InTouch nodes, you can create a list called TankFarmAlarms. To this list, you add alarm groups from all nodes that run tank farm InTouch applications.

For more information on creating alarm groups, see Create an alarm group.

Summary alarms versus historical alarms

Summary alarms are alarms that are currently active. Historical alarms are alarms that are not currently active and are typically stored to the alarm database.

For example, you might want to see a summary of all current alarms that are awaiting acknowledgment, whereas all other alarm information is of historical interest and of less urgency.

Alarm disablement, inhibition, and suppression

You can turn alarms "off" or ignore them without actually removing the alarm configuration. You can either disable an alarm, inhibit it, or suppress it.

Alarm disablement and inhibition is controlled at the alarm provider. Suppression is controlled at the alarm consumer. For more information on providers and consumers, see Alarm providers and consumers.

- **Disablement**. You disable an alarm at the alarm provider by setting a flag that marks it as disabled. No matter what alarm conditions occur, the item is never put into an alarmed state. For information on dotfields you can use to disable an alarm, see Enabling and Disabling Alarms for a Tag or Alarm Group.

  You can disable or enable all of a tag's alarms at one time. Also, for an alarm that has sub-states, you can disable each sub-state individually.
- **Inhibition**. You inhibit an alarm by:
  1. Adding an "inhibitor" tag to the alarm configuration in WindowMaker. The inhibitor tag is used at run time to mark the alarm as inhibited.
  2. Setting the inhibitor tag to True or False at run time. When the inhibitor tag is False, the alarm is handled normally. When the inhibitor tag is True, the item cannot alarm.

  Each alarm sub-state can be inhibited by a different tag, and you can leave some sub-states with no inhibitor tag assigned.

  Assigning a tag as an inhibitor tag for an alarm increases its cross-reference use count.
- **Suppression**. Suppression causes an alarm consumer to ignore certain alarms. If an alarm matches the exclusion criteria, it is not visible. That is, it is not shown on a display, printed, or logged at that particular alarm consumer.

  The actual alarm generation is completely unaffected by suppression. Alarm records can still be logged into alarm history.

If an alarm becomes disabled or actively inhibited while the item is in an alarmed state, the item is forced to a different (valid) state. What that state should be depends upon which states are available and whether they have also been disabled. This activity is handled by the alarm provider according to the type of alarm, limit values, and so on.

An alarm that is disabled or actively inhibited is not waiting for an acknowledgment. If the alarm has sub-states, it can only be waiting for an acknowledgment on sub-states that are still available.

Terminal services alarm support

By using the Distributed Alarm system with Terminal Services for InTouch, alarm clients running on different terminal sessions can select what alarm data to show and how to present it.

Alarm Providers identify themselves by a name that uniquely identifies their application, and the instance of their application. This information is made available to the Distributed Alarm system when the Alarm Provider or the Alarm Consumer registers with the Distributed Alarm system.

The node on which an Alarm Provider is running is identified by a name that uniquely identifies the computer node in the system. The alarm records that are generated by Terminal Services Edition (TSE) client sessions include an expanded node name in the format Node:IP Address; for example, serverAlarm:192.168.1.23. This information is made available to the Distributed Alarm system when an instance of it starts up on the computer node.

When an alarm event is logged, the node and complete Alarm Provider name identify the source of the alarm.

When an alarm is acknowledged in a Terminal Services environment, the Operator Node that gets recorded will be the name of the client machine that the respective operator established the Terminal Services session from. If the node name can't be retrieved, the node's IP address will be used instead.

A terminal server client session cannot be an InTouch alarm provider.

Distributed alarm system data storage

There are several forms of data storage used in the Distributed Alarm system:

- **Internal alarm memory (buffer)**

  Most information about current and recent alarms is held in memory on various computer nodes. InTouch uses two memory locations: one for summary alarms (current) and one for historical alarms and events. This model is also used in the Distributed Alarm system.

  The memory for summary alarms grows as needed to accommodate all current alarms up to the limit of available memory. The memory for historical alarms can grow only to a pre-determined limit. After the historical memory reaches this limit, the oldest alarm records are discarded as new ones are added. In a multi-node environment, the alarm memory on the various nodes constitute a single collective of alarm memory.

  For information about setting the limit, see Configure the alarm buffer size.

- **Alarm database**

  The Alarm DB Logger creates a database, keeping track of when an alarm occurs, makes a sub-state transition, is acknowledged, and when it returns to normal. Essentially, these records constitute a history of alarms in the system.

Because it is based on the use of queries, the Distributed Alarm system supports using one computer node to log alarms for several other nodes.

Build

Manage InTouch HMI applications

When managing InTouch HMI applications, you:

- Create or delete InTouch applications. See Create an InTouch application and Delete an InTouch application from the Application Manager.
- Open applications in either WindowMaker or WindowViewer. See Open an application in WindowMaker and WindowViewer.
- Search for applications. See Find InTouch applications.
- Move an application to a different computer. See Publish applications to remote nodes.
- Distribute applications among multiple computers. See Distributing Applications in *AVEVA™ InTouch HMI Application Deployment.*
- Manage InTouch services. See About managing InTouch services.
- Import or export tag definitions, windows, and scripts. See Export and import InTouch components.
- Configure security. See Secure InTouch in *AVEVA™ InTouch HMI Management*.

You can extend your application by:

- Translating text strings and alarm comments into different languages. See Switch a language at runtime in *AVEVA™ InTouch HMI Application Run Time*.
- Integrating an application with Application Server. See Managed InTouch Applications and About viewing applications at runtime.
- Displaying applications on multiple monitors. See About Setting Up a Multi-Monitor System.
- Using applications on a Tablet PC. See About Using InTouch on a Tablet PC.

For the Windows Vista, Windows 7, and Windows Server 2008 operating systems, a standard user can use the InTouch Application Manager to find an application and open WindowViewer. Application properties will be read-only for the standard user. All read/write or configuration operations from Application Manager require administrative privileges.

Start the Application Manager

You can start the Application Manager from the **Start** menu or from a shortcut placed on your computer's desktop.

Start the Application Manager for the first time

1. On the taskbar, select **Start**, point to **Programs**, point to **AVEVA InTouch HMI**, and then select **InTouch HMI Application Manager**.

   The **AVEVA Application Manager** opens.

When **Operations Control Connected Experience** mode is enabled in the Configurator, you will be prompted to authenticate with AVEVA Identity Manager when you launch the Application Manager for the first time.

2. Authenticate using AVEVA Identity Manager. For more information, refer to the Authenticate using AVEVA Identity Manager section.

   The window shows InTouch applications on your computer that you created or found using the Application Manager.

Identify the InTouch version that was last used to save the application

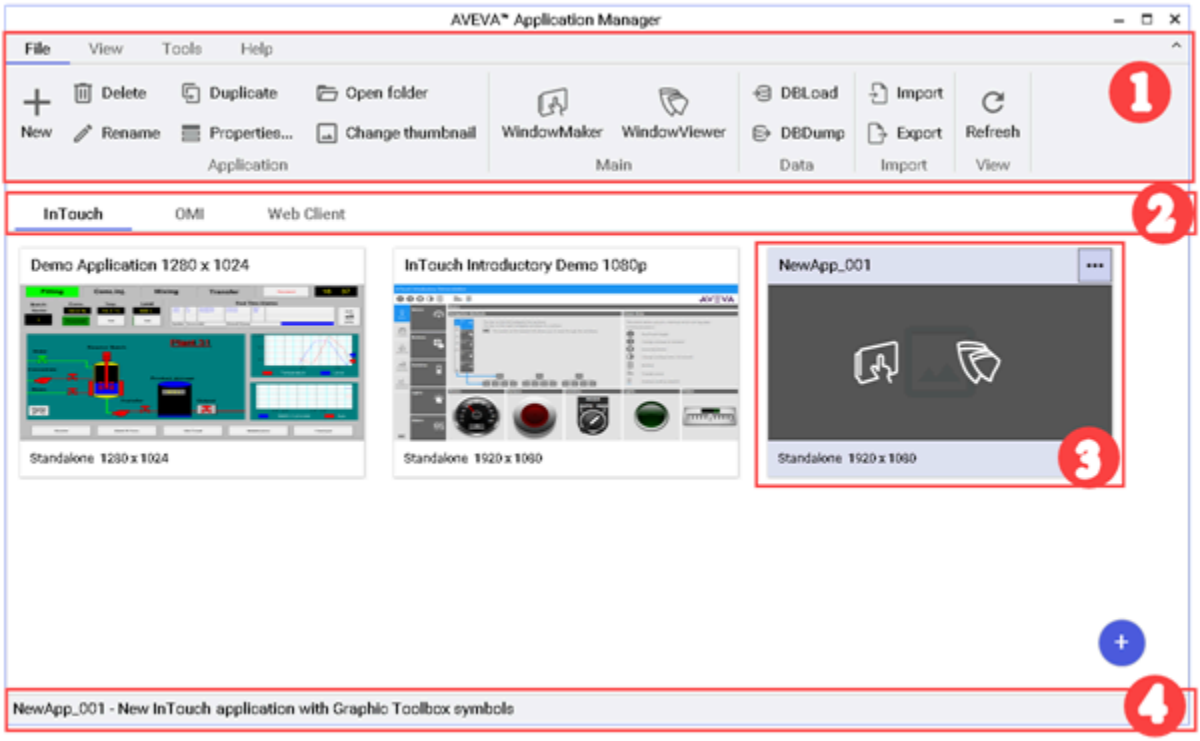
1. Open the Application Manager.
2. On the **View** menu, in the **View** group, select **Details**.

   The application list appears in the Details grid view.
3. See the **Version** column for the application.

Use the Application Manager

The Application Manager is the first step in creating InTouch HMI applications and provides multiple options for application management.

The interface is divided into the Ribbon and the working area/canvas.



Identifier	Element	Description
1	Ribbon bar	Contains menu items across the top, and commands on each menu item, grouped based on their

Identifier	Element	Description
		functionality.
2	Application bar	Displays shows tabs for InTouch, OMI, and Web Client. The command on each menu item in the Ribbon changes with the selected application type.
3	Application tile	Represents each application.
4	Status bar	Shows the description message for the selected item.

The ribbon contains the following menu items:

- **File**: The **File** menu is the default ribbon menu. It has five groups of related commands; **Application**, **Main**, **Data**, **Import**, and **View**. You can modify the application settings by using commands such as create, duplicate, delete, open in WindowMaker or WindowViewer, import, export, and many other common features.
- **View**: The **View** menu provides options to alter the view of the list on the canvas, such as **List**, **Tile**, or **Details**.
- **Tools**: The **Tools** menu contains important Application Manager tools like **Node Properties**, **OPC UA Configuration**, **Find**, and **Insight Publisher**.
- **Help**: The **Help** menu allows to access the InTouch HMI Help document, as well as view the Application Manager version and copyright version.

The working area/canvas is divided into tabs and each tab represents an application type - InTouch, OMI, and Web Client.

Create an InTouch application

You can create a new InTouch application using the Application Manager. The application path cannot exceed 114 characters, including the network drive letter, colon, and all backslashes. If the limit is exceeded, you cannot open the application in WindowMaker. Application name must be 32 characters or less.

The INTOUCH.ini file is created when you create an application. The INTOUCH.ini file contains the default configuration settings for your application. As you configure your application, the new settings are written to the INTOUCH.ini file.

After you create an application, you can copy an existing INTOUCH.ini file to the application folder. This way, you do not need to configure customized InTouch parameters each time you create a new application.

Create a new application

1. You can create a new application using different options on the Application Manager.
   - On the **File** tab, in the **Application** group, select **New**.
   - Select  at the right bottom.

- Press Ctrl+N
- Right-click in the empty space, and select **New…**.

The **Create New Application - Select a Template** page appears.

AppManager-CreateNewApp

2. Highlight a Template for the new application and select **Next**.

   The Blank template will not include the default Graphic toolbox symbols and SAL symbols. In all templates you can use both InTouch Symbols and Industrial graphics in your application. If you have created any application templates and they are available in the correct folder, those templates will also be displayed.

   The **Create New Application - Enter Application Details** page appears.

3. Enter the following details:
   - **Type**:Displays the type of application selected.
   - **Application Name**: Enter the name of the application
   - **Directory Name**: Type the application folder name.
   - **Application Path**: Select the ellipses (...) button to browse a folder other than the default location.
   - **Set Default Directory**: Use the toggle to set the application path as the default directory.
   - **Resolution**: Select the application target resolution from the dropdown list.

     The available resolution options are:
     - Screen Resolution (default)
     - 1024 x 768
     - 1280 X 1024
     - 1366 X 768
     - 1440 X 900

       To edit the width and height, select **Custom Resolution** from the Resolution dropdown list.
   - **Width**: Specify the width of the application.
   - **Height**: Specify the height of the application.
   - **Description**: Type an optional description up to a maximum of 255 characters.
4. Select **Finish**.

   After the application is created, it appears in the Application Manager's list of applications.

Create a new InTouchView application

You identify an InTouchView application by setting an option from Application Manager when you create the application. You can configure an InTouch application to an InTouchView application and vice versa. A full InTouch license is required to run applications converted from InTouchView to InTouch. When an application is changed to an InTouchView application, the license acquired will differ based on the data source selected. For more information, see InTouchView applications.

Change an InTouch application to an InTouchView application

You can change an InTouch application to an InTouchView application if the application only needs to connect to data from an Application Server or InTouch Tag Server. Some WindowMaker functions are unavailable in an InTouchView application.

Change an InTouch application to an InTouchView application

1. Open the InTouch application in WindowMaker.
2. On the **File** menu, highlight **Configure**, and then select **WindowViewer**.

   The **WindowViewer** configuration screen appears.
3. In the **Application type** tab, select the **InTouchView Application** checkbox.

   - To connect to data from a Galaxy, select **Galaxy**.
   - To connect to data from InTouch Tag Servers, select **InTouch Tag Server**.
4. On the **Home** menu, in the **Tags** group, select **Access Names**.
   a. If you select **Galaxy**, you must remove all Access Names other than Galaxy before converting an InTouch application to InTouchView. If they are not removed, a message is shown during the conversion attempt.
   b. If you select **InTouch Tag Server**, you can configure new access names.
   Select **Add**, and provide the Access Name and Node Name.

      The Application Name and Topic Name are grayed out. All Access Names other than Galaxy or referencing to InTouch Tag Server must be removed prior to changing this application to InTouchView. Only I/O references from access name(s) where Application is configured as *view* will be binded, I/O reference from access name *Galaxy* will not be binded.
   c. Select **OK**.

For more information on configuring Access Names, see Set up access names.

5. Select **Close**.
6. Select **OK.** When a message appears, select **OK**.

Open an application in WindowMaker and WindowViewer

You must open a new application in WindowMaker before you can open it in WindowViewer.

Open an application in WindowMaker

1. Select the application in the Application Manager window.
2. On the **File** menu, in the **Main** group, select **WindowMaker**.
   Alternatively, hover over the application tile, and select **WindowMaker**.

The application opens in WindowMaker.

Tip: You can also double-click an application to open it in WindowMaker.

Open an application in WindowViewer

1. Select the application in the Application Manager window.
2. On the **File** menu, in the **Main** group, select **WindowViewer**.
   Alternatively, hover over the application tile, and select **WindowViewer**.

The application opens in WindowMaker.

Customize the Application Manager window

You can hide the toolbar and status bar. You can also change how the applications are listed in the Application Manager window. Applications can be shown as Details, List and Icon.

- **Details** – This view lists the application in a tabular form with all the application related properties like path and resolution.
- **List** – This view displays the application name, type, thumbnail and the ribbon.
- **Icon** – This view displays the application name, type, resolution, description, thumbnail and ribbon.

Show/hide the toolbar

- On the **View** menu, select **Toolbar** to show or hide the status of the selected application.

Show/hide the status bar

- On the **View** menu, select **Status Bar** to show or hide the status of the selected application.

Change the view for the application list

- On the **View** menu, highlight the appropriate command or select the corresponding button on the toolbar.

Rearrange the order of the Toolbar options

1. Select the icon and select **Customize…**
2. Under the **Items** tab, highlight an option (for example WindowMaker), and select the **Move Up** or **Move Down** buttons to change the order the options appear on the toolbar.

Remove a tool from appearing on the Toolbar

- On the **View** menu, select the checkbox against the option.

1. For example, clear the checkbox against the option WindowMaker and select **Close**. The Window Maker option will not appear on the toolbar.

Apply themes to the Application Manager

- On the **View** menu, in the **Theme** group, select **Light Theme**/**Dark theme** to toggle between the two themes.

Use the application tile

In the Icon or List view, each application displays as a tile with application specific options.

Launch WindowMaker or WindowViewer

Hover over the application tile to view the options to launch the application in WindowMaker or WindowViewer.

Additional Settings

Select the ellipses icon to view the additional settings.

- **WindowMaker**: Select to launch WindowMaker.

  Note: You may have to authenticate with AIM when you launch the WindowMaker for the first time. For more information refer to the Authenticate using AVEVA Identity Manager section.

- **WindowViewer**: Select to launch WindowViewer.

  Note: You may have to authenticate with AIM when you launch the WindowViewer for the first time. For more information refer to the Authenticate using AVEVA Identity Manager section.

- **Application folder**: Select to navigate to the application folder.
- **Properties**: Select to launch the application properties window.
- **Change Thumbnail**: Highlight the icon, and select the image from the Browse dialog box. The image will be displayed in the Icon and List view.
- **Rename**: Select and enter the new name of the application.

- **DBLoad**: Select to launch the DBLoad utility.
- **DBDump**: Select to launch the DBDump utility.
- **Export as Template**: Select to export the InTouch application. See Export InTouch applications to use as a template.
- Delete: Select to delete the application.

Lock and unlock InTouch applications

If an application is being edited in another session or was not closed correctly, the application may be locked. You must unlock the application using the Application Manager.

Unlock an InTouch application

- Launch Application Manager.
- Select the application currently locked.
- Select the application tile to unlock and reload the application.

The application is unlocked and available for use. For more information, see Application editing locks.

Modify an InTouch application

You can rename an application and change the application properties.

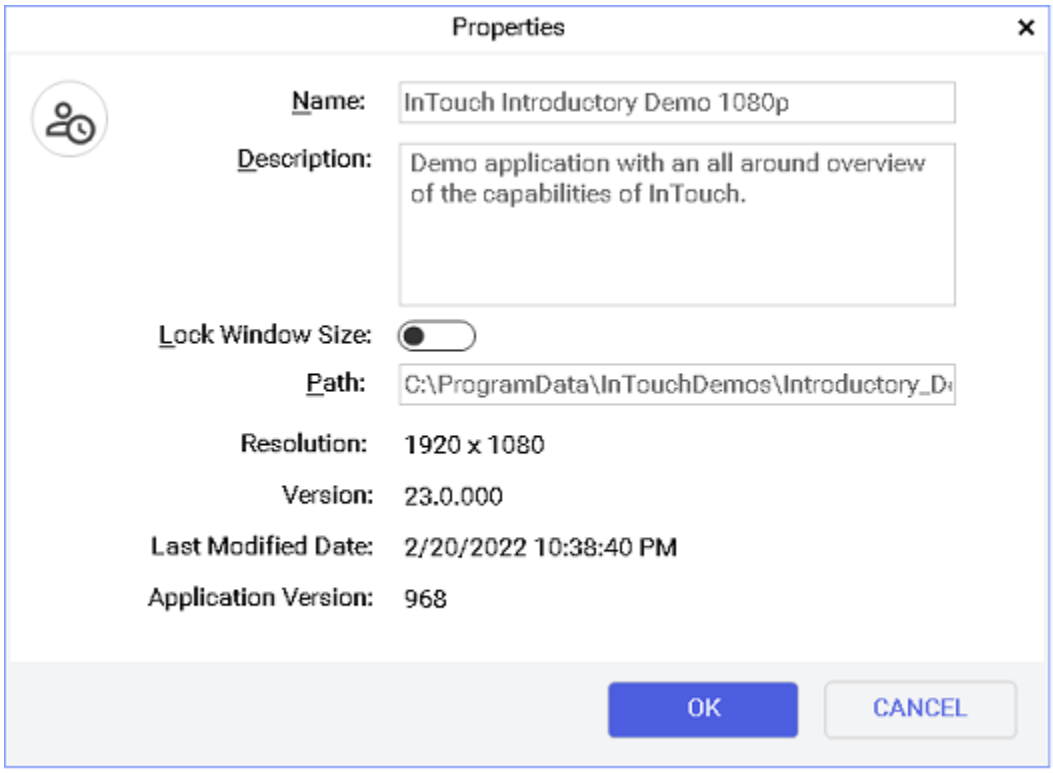
Rename an application

1. Select the application in the list.
2. On the **File** tab, in the **Application** group, select **Rename**.

Modify application properties

1. Select the application in the list.
2. On the **File** tab, in the **Application** group, select **Properties**.

The **Properties** dialog box appears.



3. Configure the application properties.
   - In the **Name** box, type a new name for the application.
   - In the **Description** box, type another description for the application.
4. Select **OK**.

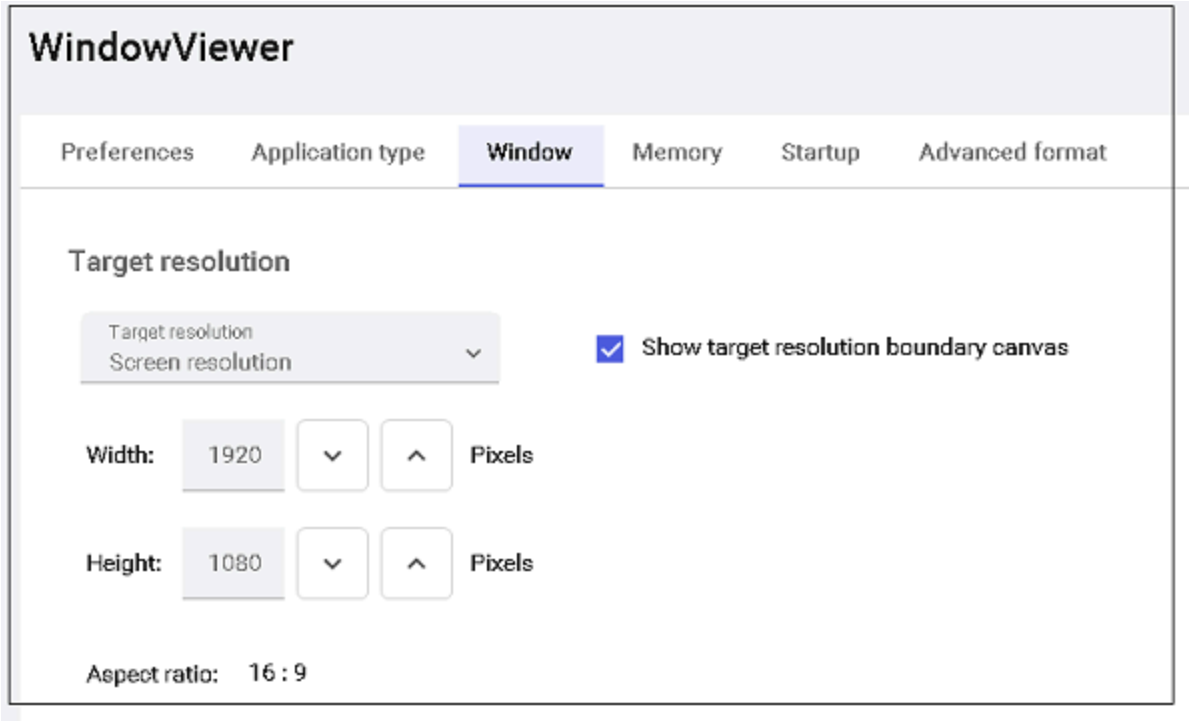
Change target resolution

You can change the specified target resolution while editing the application in WindowMaker. This functionality is not available using command line.

Do the following:

1. Open WindowMaker.
2. On the **File** menu, highlight **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.
3. Select the **Window** tab.
4. In the **Target resolution** section, edit the target resolution as needed.



5. Select **OK**.

   The boundary canvas will be modified to reflect the change. Graphics, window size and window controls will remain the same.

   Note: The **Show target resolution boundary canvas** is checked by default.

Delete an InTouch application from the Application Manager

When you delete an application from the Application Manager, the application files remain on your computer.

Delete an application

1. Select an application in the list.
2. On the **File** tab, in the **Application** group, select **Delete**.
3. In the message that appears, select **Yes**.
4. You can also delete an application in the following ways:
   - Right-click the application and select **Delete** from the context menu.
   - Select the **delete** icon from the application ribbon.
   - Select the application and press the **delete** key on the keyboard.

   You can select and delete multiple applications.

Delete earlier Modern applications

If the node contains older Modern application and the node is upgraded to the current Product version, you can select to delete the older Modern application. This action will permanently delete the Modern application folder and repository.

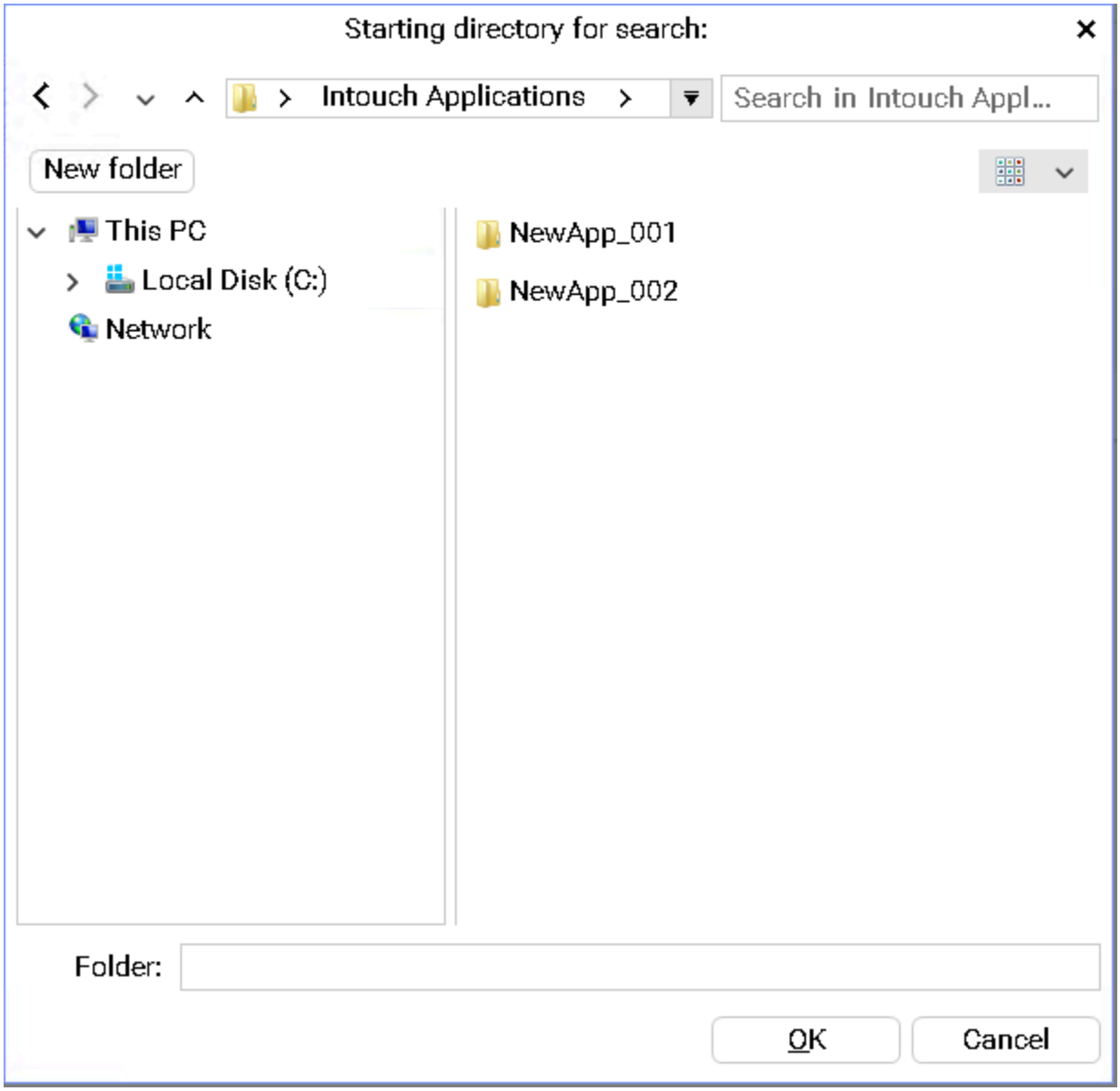
Find InTouch applications

You can search for existing InTouch applications. Application Manager shows any applications that are found.

An application cannot be opened in WindowMaker if the full path exceeds 114 characters (including the network drive letter, colon, and all backslashes). If an application exceeds the character limit, rename the folders in the path or move the application to a different location.

Find applications

1. On the **Tools** tab, select **Find**. The Starting directory for search dialog box appears.



2. Select the folder in which you want to search for applications.
3. Select **OK**.

Work with application templates

You can develop your own application template from a Standalone application. Developing an application template is a three-step process.

1. Create an application and set appropriate windows in the application to template windows.
2. Create an application thumbnail for preview in the Application Template Browser.
3. Export the application as an .aaPKG file to make it available as a template in the Browser.

Set application windows as template windows

1. Create an application.
   a. Develop your application using graphics, scripts and windows.
2. Do the following for all windows in the application:
   a. Right-click each window and select **Properties**.
   b. In the **Window Properties** panel, select the **Template** checkbox.

Upon selecting **OK**, each window is automatically placed in the Template Windows folder in the Windows pane.

You must now create and assign a thumbnail to the application you want to make into a template.

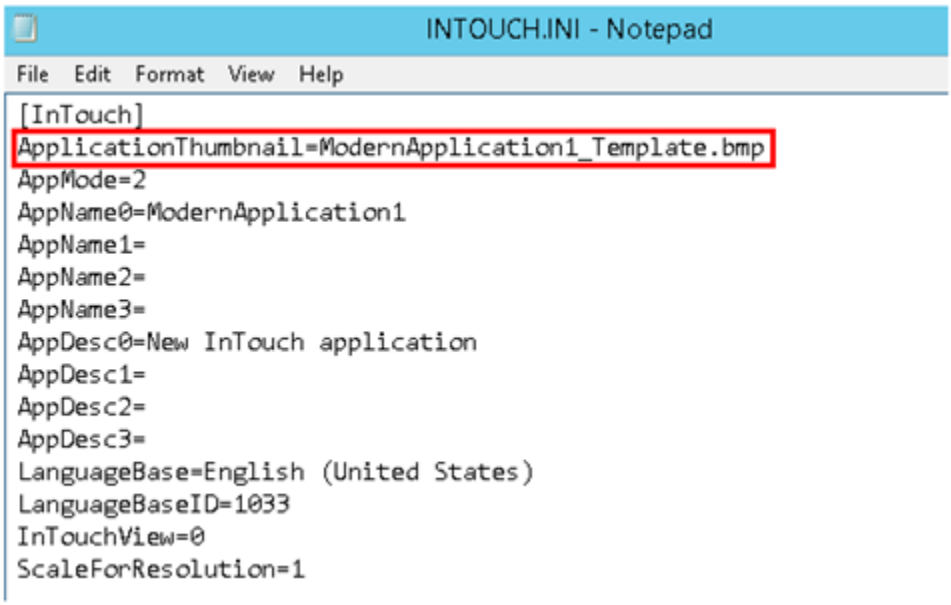
Create and assign an application thumbnail

1. Using any screen capture program, take a screen capture of your application at either configuration or run

time.

2. Save the image to any picture file format, such as a .bmp or .png file, and copy it to your application folder.
3. From your application folder, open the INTOUCH.INI file with a standard text editor such as Notepad.
4. Edit the INTOUCH.INI file to include the file name of the image in the **ApplicationThumbnail** field.

Note: The **ApplicationThumbnail** field is case sensitive and must exactly match the name and extension of the thumbnail image.



5. Save your changes and close.

The application must now be exported as a .aaPKG file that will populate in the **Application Template Browser**.

Export application and create template

1. In the Application Manager, export your application to create an .aaPKG file.

   For details on how to export an Application, see Export InTouch applications to use as a template.
2. Copy the exported .aaPKG file into the following directory:

   ```
   C:\Program Files (x86)\Wonderware\InTouch\ApplicationTemplates
   ```

   Your application is now available as a template in the Application Template Browser.

Note: The application template thumbnail you created in the previous procedure is extracted from the exported .aaPKG file. If your application template appears in the **Application Template Browser** with a blank thumbnail, a valid image could not be extracted. Be sure a valid image file format was used to save the thumbnail and the exact filename is entered into the ITOUCH.INI.

View the InTouch application folder

Using WindowMaker, you can view the files in the InTouch application folder that are deployed and available at run time.

Open the application folder

1. Open the InTouch application in WindowMaker.
2. On the **View** menu, in the **Application** group, select **Open folder**

   The standard Windows Explorer window appears showing the InTouch application folder. For managed applications, the application folder on the Galaxy Repository node is displayed.

Export InTouch applications to use as a template

The Export As Template option allows you to export an .aapkg file that contains all the graphic related contents (symbols, client controls, script library, language, styles) and save it as a template. This template file can then be used to create other Standalone or Managed Application.

Export an application as a template

1. In the Application Manager, on the File tab, in the Main group, select **Export as Template**.

   The **Export InTouch Application** dialog box appears.
2. Provide a location to store the .aapkg file.

   The Export InTouch Application progress window appears.
3. After the export is complete, select **Close**.
4. Select **View Details** to check for any errors.

   The .aapkg file will be available in the location specified, and can now be used as an application template to create new applications.

Update InTouch Web Client settings using the Application Manager

The Web Client tab of the AVEVA Application Manager provides options for the user to configure InTouch Web Client related settings.

Update the InTouch Web Client settings

1. Launch the Application Manager and select the **Web Client** tab.

   The **Web Client settings** screen appears.
2. Configure the following settings:
   - *Current application:* Select an application from the list to modify the web client settings.
   - *Graphic refresh rate (ms)*: Set the rate on how frequently the web browser will query the web server for graphic data. The default is 1 second. For more information, see System Platform Installation.
   - *Alarm refresh rate (ms)*: Set the rate on how frequently the web browser will query the web server for alarm data. The default is 1 second. For more information, see System Platform Installation.
   - *Show header*: Select the checkbox to display the Title Bar.
   - *Enable navbar*: Select the checkbox to display the Navigation Bar.

     This setting will be disabled if the **Show header** setting is not selected.
   - *Allow Anonymous Access*: Select the checkbox to allow users access to web client without authentication.

- *Enable workspaces*: Select the checkbox to enable Workspaces.

3. Under the Advanced settings section, select **Customize** to configure the following settings:
   - *Website name*: Provide a string that will replace the standard URL.
   - *Application title*: Provide a string that will appear as a title for the application in the App Bar.
   - *Website title*: Provide a string that will appear as the title on the title bar of the browser.
   - *Website icon*: Provide an image file that will replace the icon on the title bar of the browser.
     a. Select **Choose file**... to select the icon image.
     b. Select a file.
     c. Select **Open**.
4. Select **Save** to save the advanced settings.
5. After modifying the settings, select **Apply**.
6. To view the InTouch Web Client, select **Launch**.

Note: If the web site name or address is changed, ensure to configure the AVEVA Identity Manager to register the new web site. For more information, see Register with the AVEVA Identity Manager.

For more information on these settings, see Viewing *Application Graphics in a Web Browser*.

Launch an InTouch OMI ViewApp

Use the OMI tab in the Application Manager to launch the Viewer for InTouch OMI ViewApps.

1. Develop and deploy an InTouch OMI app.
   Once the ViewApp is deployed it will appear in the Application Manager OMI tab.
2. Select the ViewApp.
3. In the File menu for OMI, select **OMI Viewer**.
   The Viewer is launched and the ViewApp is displayed.

Register with the AVEVA Identity Manager

Using the AVEVA Identity Manager (AIM), you can configure the InTouch Web Client to use Single Sign On, instead of the default Windows OS-based authentication.

Note: InTouch Web Client supports ArchestrA-based security with AIM configuration only.

Configure Identity Server

1. In the System Platform Configurator, configure the **Common Platform** > **System Management Server**.
2. In AVEVA Application Manager, register the AIM server with user credentials.

The AIM Registration dialog box can also be used to configure the reverse proxy server:

1. Setup the reverse proxy server.
2. In the System Platform Configurator, configure the **Common Platform > System Management Server**.

3. Provide the Secure Gateway address in the AIM registration dialog box.

You can select a remote or local System Management Server. For more information on configuring the System Management Server, see *System Platform Installation*.

Register with the Identity Server

1. In Application Manager, select the **Web Client** tab.
2. Select **Tools** and then **Identity manager**.

   The **Identity server settings** screen appears.
3. Select the **Use AIM Server as the authentication server** checkbox to enable the use of the AVEVA Identity Manager.

   The **Identity Server** field displays the Identity Server configured in the Configurator.
4. Update the following settings:
   a. **User Name**: Provide the user name to connect to the Identity Server. The user must be part of the Administrators user group.
   b. **Password**: Provide the password for the corresponding user name.
5. To complete the reverse proxy setup, provide the URL of reverse proxy or DMZ server in the **Secure Gateway** field.

   This is an optional setting and only needs to be set if the Web Client is hosted behind a reverse proxy server.
6. Optionally you can also select the **Allow Industrial Graphics to be embedded in any website** checkbox to view the web client within an HTML iframe in runtime.
7. Select **OK**.
8. Select **Apply**.

Note: If the web site name or address is changed, you must configure the AVEVA Identity Manager to register the new web site.

Work with Credential Manager

The Credential Manager allows you to store your credentials in a secure location. You can use the stored credentials for authenticating access to other components in WindowMaker, WindowViewer, and Alarm utilities.

Launch Credential Manager

1. Launch Application Manager as an administrator.
2. On the **Tools** menu, in the **Tools** group, select **Credential Manager**.

   The Credential Manager screen appears.

   Note: You must open the Application Manager as an administrator to access Credential Manager. If a standard user attempts to open the Credential Manager, the system prompts to launch in Admin mode.

The Credential Manager consists of two sections:

- Named Credentials
- Details for Credential

The **Named Credentials** section displays the list of credentials in a grid format, with the following columns.

- Name: Indicates the name of the credential.
- Type: Indicates the type of credential - username and password or domain username and password.
- Group: Indicates the Windows user groups to which the credentials belong. Select the ellipses icon (...) in the Group column to select one or more groups. For more information, see Manage named credentials.

The **Details for Credentials** section displays the following fields for the selected credentials

- Domain
- Username
- Password.

Manage named credentials

Use the Credential Manager screen to add/delete credentials. You must assign groups to each of the credentials. The users in those groups will be able to access the credentials from various InTouch components such as WindowMaker, WindowViewer, and Alarm utilities.

Add a credential

1. In the **Named credentials** section of the **Credential Manager**, do the following:
   a. Select **Add (+)**.
      A new row is added in the Named Credentials grid.
   b. In the **Name** column, enter the name of the credential.
   c. In the **Type** column, select the type of credentials from the drop-down list.
      a. **Username and password**: Allows you to assign SQL Server accounts to the credentials.
      b. **Domain username and password**: Allows you to assign Windows accounts to the credentials.
   d. In the **Group** column, select the ellipses at the end of the column.
      The **Select Group** screen appears.
   e. From the **Select from** list drop-down list, select the domain.
   f. Scroll through the **Available OS group** list to locate the group, and select the + icon.
      The selected OS group appears in the **Selected group** section.
      By default, all named credentials are added to the Administrators group. In addition, we recommend you to add groups that the users running the application belong. This will allow the users to access the credentials even when running the application in a non-administrator mode, such as WindowViewer.
      Note: To remove a group, highlight the group in the **Selected group** section, and select **Delete (-)**.
   g. Select **OK**.
2. In the **Details** section, do the following:
   a. In the **Domain** field, enter the user domain.
      Note: The Domain field is enabled for Domain username and password type only. If you select the Credential Type as Username and Password, the Domain field is disabled.
   b. In the **User name** field, enter the user name for the selected credential.

c. In the **Password** field, enter the password for the given user name.

d. In the **Confirm password** field, enter the password again.

3. Select **Save**.

Delete a credential

1. Select a credential from the grid.
2. Select **Delete (-)**.

   The selected credential gets deleted.

Export and import named credentials

You can export the credentials for the selected applications, and import it on the node where the application is to be imported.

Export Credentials

You can export credentials by one of the following ways:

- Export credentials only
- Export credentials when exporting applications.

When exporting credentials, you must configure a Security Key. The Security key encrypts the credentials and is required for authentication when importing the credentials.

Export credentials using Credential Manager

1. In the **Credential Managers** screen, select **Export**.
2. The **Export credentials** window appears.
3. Select the credentials from the grid to be exported.
4. In the next section, set up a **Security Key** .

   Note: The Security Key must be at least 12 characters long, and must include at least one upper case, lower case, and numerical character and one special character.
5. Select **Export**.

   The **Export credential details** file browser appears.
6. Highlight folder to save the credentials and select **Save**.

   The credentials are exported to .bin file format.

Export credentials when exporting applications

1. In the Application Manager, right-click the application to be exported, and select **Export as template**.
2. Select the **Export Credentials** checkbox.

   The application along with the credentials list are exported.

Import Credentials

You can import credentials to use on a particular node where the application has been imported.

Import credentials

1. In the **Credential Managers** screen, select **Import**.
2. Use the file browser to select the .bin file and select **Open**.
3. Enter the Security Key.

   Tip: You must use the same security key configured when exporting the application.

   The credentials are imported to the Credential Manager.

Import credentials for NAD

At the time of this release, named credentials on the NAD master will not be propagated to NAD clients automatically. That is, selecting an application to run with NAD (using the **Find application** action) does not import the named credentials.

To import the named credentials on the NAD master, import the named credentials through application manager on every NAD client.

Import Credentials Scenarios

Scenario 1: The imported credentials and credentials existing on the machine are different.

The credentials are imported to the Credential Manager and new entries are added to the Credential Manager grid.

Scenario 2: The imported credentials and existing credentials are same.

The credentials are imported to the Credential Manager and new entries are added to the Credential Manager grid. The newly imported credentials are renamed with.

Scenario 3: The names of the imported credentials and existing credential are same, but the details are different.

The credentials are imported to the Credential Manager and new entries are added to the Credential Manager grid. The newly imported credentials are renamed with an increment of the existing ones.

Manage InTouch applications with the IDE

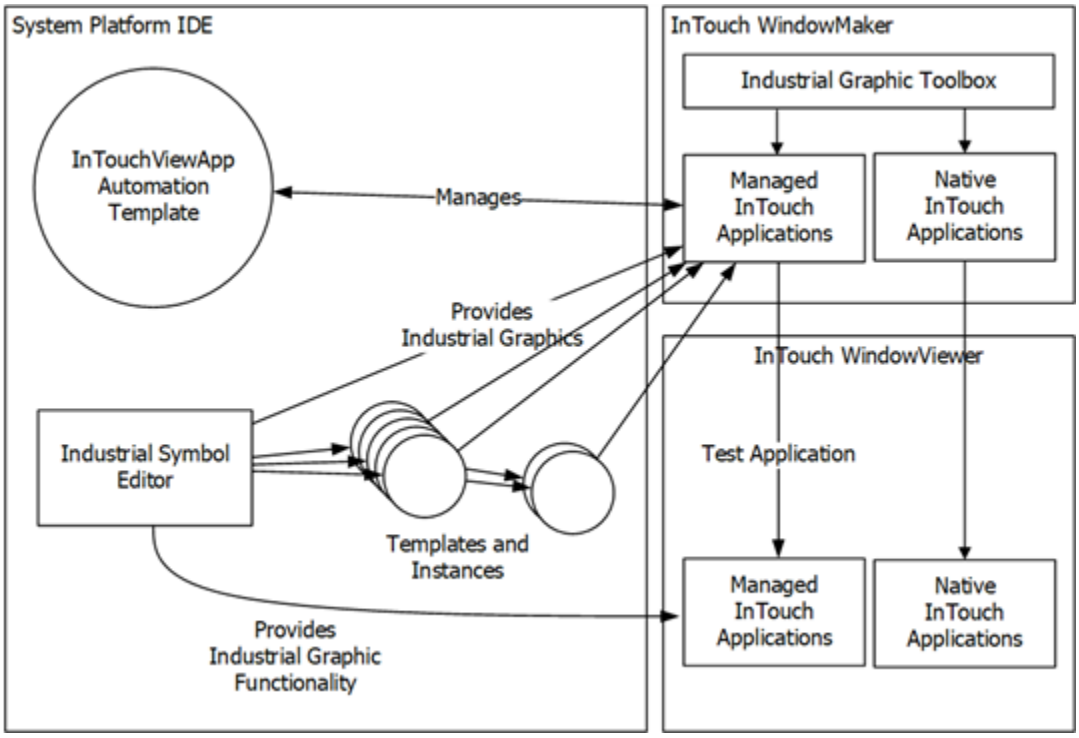
You can use the System Platform IDE to:

- Create a new managed InTouch application.
- Import an existing InTouch application to use as a managed InTouch application.
- Start WindowMaker.
- Submit the changes you make in WindowMaker to a managed InTouch application.
- Export and import a managed InTouch application together with its InTouchViewApp object.
- Publish a managed InTouch application.
- Delete a managed InTouch application.
- Import and export tag data used in a managed InTouch application.

- Export tag data to and import tag data from a .csv file.
- Switch languages for a Managed InTouch application.
- Add files to a Managed InTouch application.
- Associate all Galaxy graphics with an InTouchViewApp.
- Use an application template to create a Managed InTouch application.

You can start the System Platform IDE from the InTouch HMI Application Manager.

The following graphic shows how applications are imported, exported, managed, and published.



Create a managed InTouch application

You create a managed InTouch application by creating and configuring an InTouchViewApp object.

You can also view application version, resolution, and description information directly from the InTouchViewApp object.

The InTouch application directory is created as a share:

\\GRNodeName\GalaxyName-$InTouchViewAppObjectName

This directory is managed by the IDE, not by the InTouch HMI Application Manager.

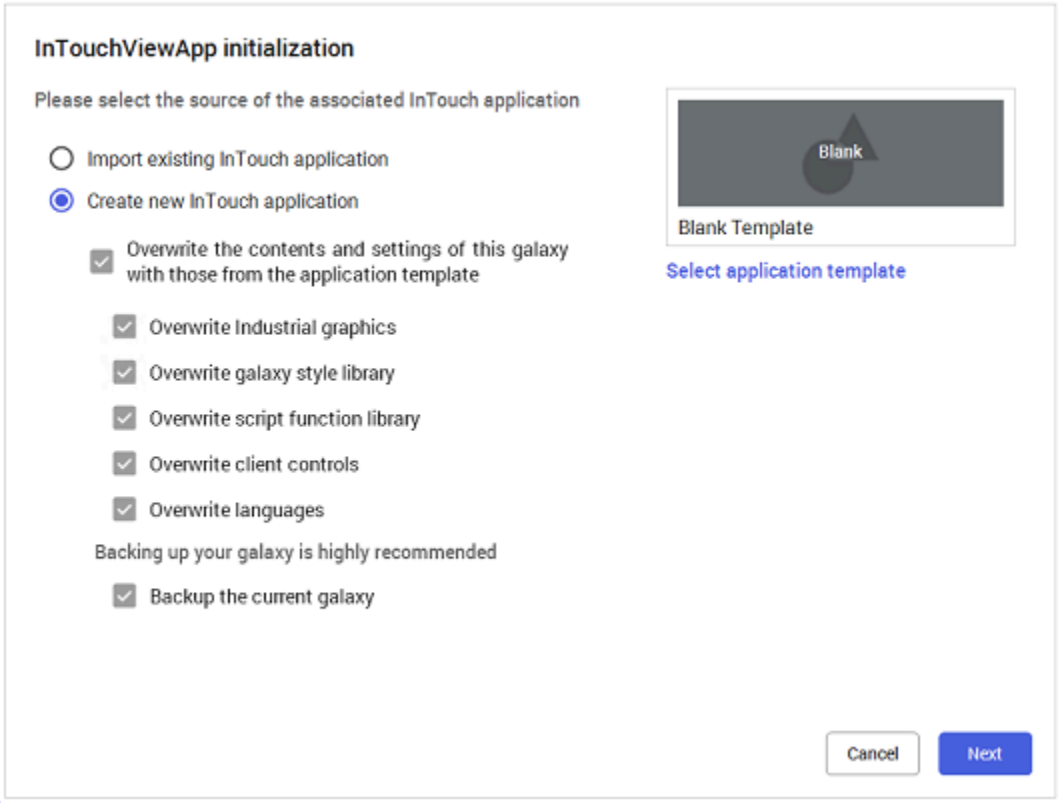
Create a managed InTouch application

1. Open the System Platform IDE.
2. In the **Template Toolbox**, expand the **System** toolset.
3. Derive a template from the **$InTouchViewApp** base template. Do the following:
   a. Right-click the **$InTouchViewApp** base template, point to **New**, and then select **Derived Template**. A new derived template appears with a default name.

b. Rename the new template if needed.

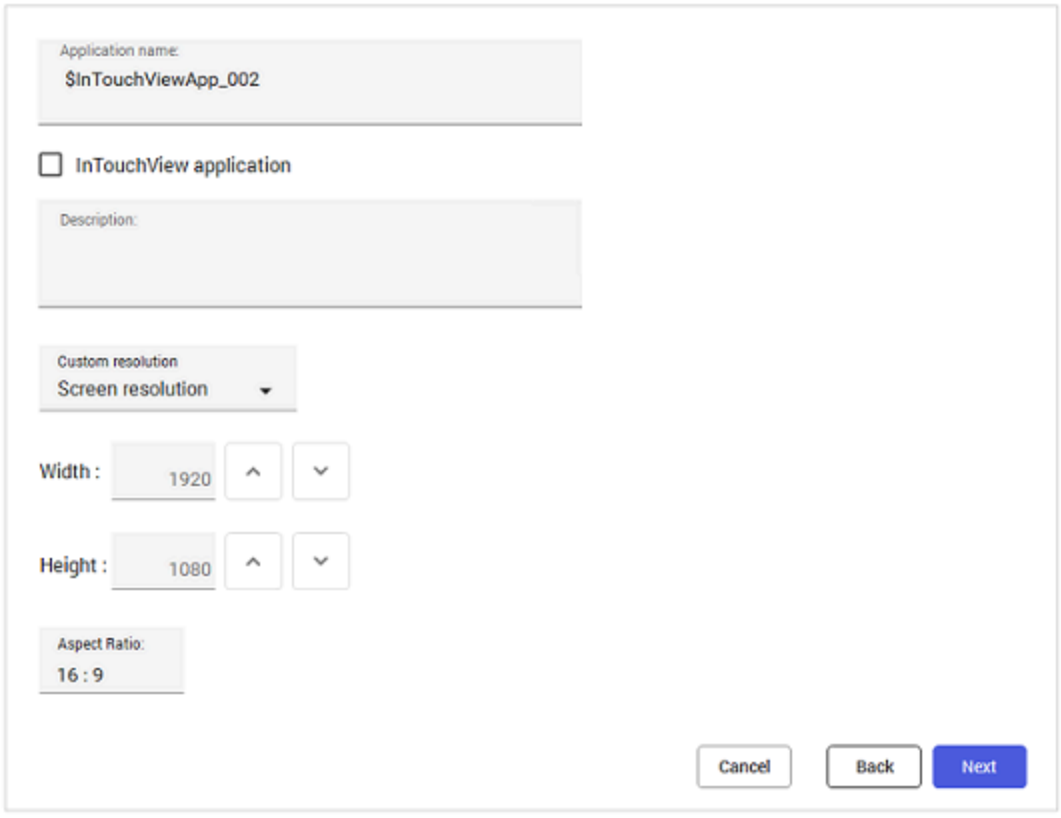
4. Double-click the derived template.

   The **InTouchViewApp Initialization** dialog box appears.



5. Highlight **Create New InTouch Application** and select **Next**.

   The next panel appears.



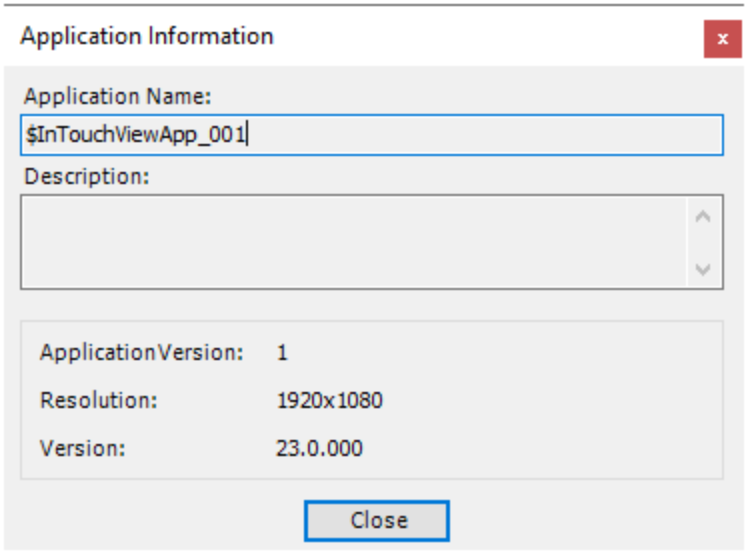
6. Type a name for the InTouch application and a description.
7. Select **InTouchView Application** to create an InTouch application that uses only Application Server references as an external data source.
8. Select the application target resolution if different than the default screen resolution option. Options for this field are as follows:
   a. Select the **Select target resolution** dropdown menu to view a list of predefined target resolutions.
   b. Highlight the **Select target resolution** dropdown menu and select **Custom**. The Pixel width and height fields becomes editable. The boundary limits are 150x150 and 10000x10000.
9. Select **Next**.

   WindowMaker starts.

   Note: If you have selected a target resolution that is different from the screen resolution, you will not be prompted to convert to screen resolution upon editing the application in WindowMaker.

View application version, resolution, and description

1. Open the Template Toolbox.
2. Right-click the InTouchViewApp template, and then select **Application Information**. The **Application Information** dialog box appears.



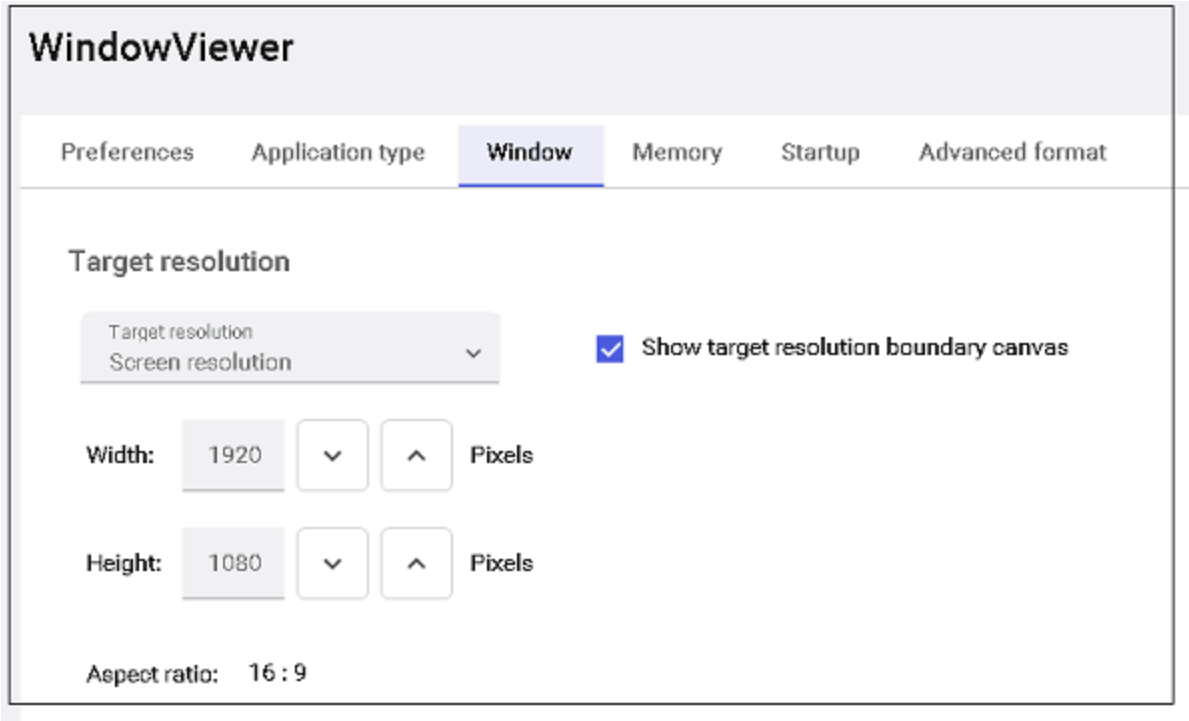
Change the target resolution

You can change the specified target resolution while editing the application in WindowMaker. This functionality is not available for command line.

Do the following:

1. Open WindowMaker.
2. On the **File** menu, highlight **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.
3. Select the **Window** tab.
4. In the **Target resolution** section, edit the target resolution as needed



5. Select **OK**.

   The boundary canvas updates to reflect the change. Windows, window size and window controls will remain the same.

Note: The **Show target resolution boundary canvas** is checked by default.

Create a managed application from an application template

Creating a Managed InTouch application based on an application template enables you to significantly reduce design time and base your new application on existing standards.

You select which template you want to base your Managed InTouch application on using the Application Template Browser. The templates available in the Application Templates Browser are exported .aaPKG files that are loaded from an Application Templates folder in the InTouch installation directory. For example:

```
C:\Program Files(x86)\Wonderware\InTouch\ApplicationTemplates
```

The folder hierarchy in the **Application Templates Browser** will mimic the hierarchy in this directory.

Create a managed application based on an application template

1. Open the IDE and create a derived template from the $InTouchViewApp object.

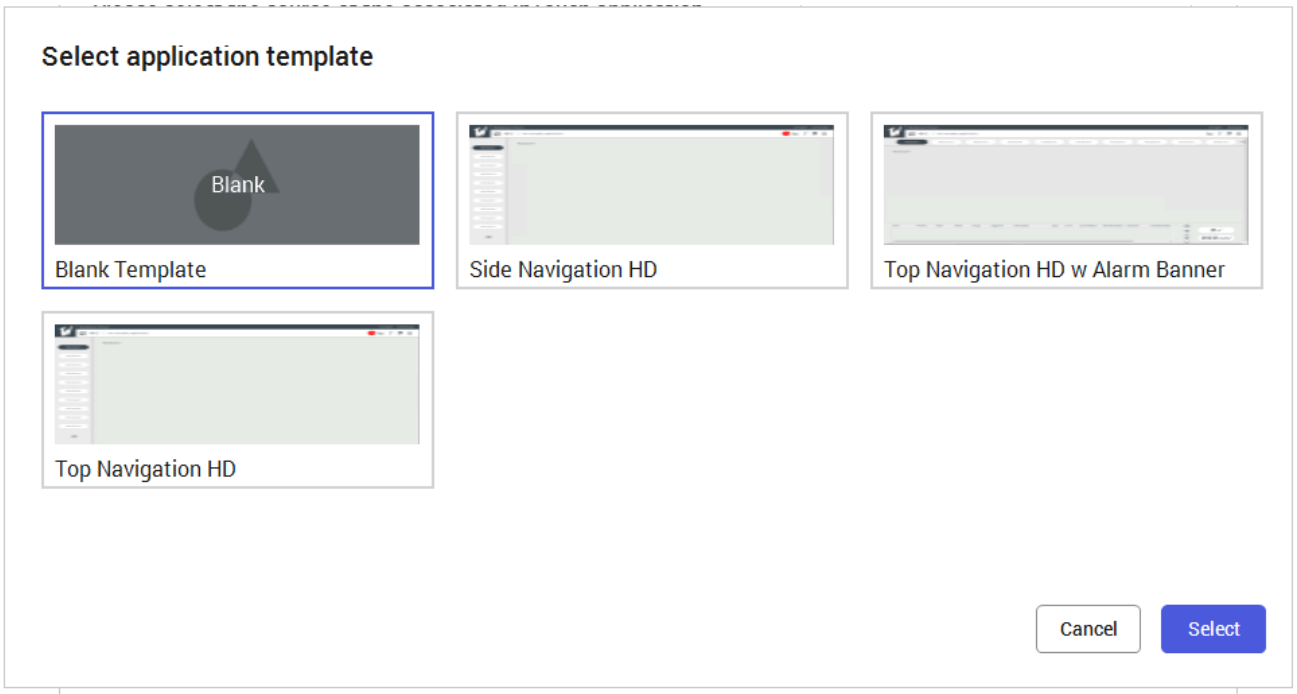
   You can optionally rename the derived template.

2. Double-click the derived template.

   The **InTouchViewApp Initialization** dialog box appears. The **Create new InTouch application** option is selected by default.

3. Choose **Select application template**.

   The **Select application template** dialog appears.



Each thumbnail in the browser maintains its aspect ratio.

4. Highlight your preferred application template and select **OK**.

   Your selection will be populated in the **InTouchViewApp Initialization** dialog box.

5. Carefully review and select your preferred galaxy settings overwrite options.

   Note: The **Overwrite the contents and settings of this galaxy with those from the application template** option is selected by default. The specific galaxy settings that will be overwritten are broken down in the child options. Any or all of these options can be unchecked at this time.

   The overwrite options are as follows:

   - **Overwrite Industrial graphics**

     If selected, galaxy symbols will be replaced by application template symbols in case of conflict.

   - **Overwrite galaxy style library**

     If selected, the application template style library will replace the entire galaxy style library. If unselected, this step will be skipped. There is no merge of style libraries.

   - **Overwrite script function library**

     If selected, the galaxy script function library will be replaced by the application template script library in case of conflict.

   - **Overwrite client controls**

     If selected, a galaxy client control will be replaced by the application template client control in case of conflict.

   - **Overwrite languages**

     If selected, the language settings defined in the application template will replace those defined in the galaxy.

     If unselected, the galaxy language settings will remain in case of conflict. If unselected and no conflicts occur, the application template language settings will append to the galaxy settings.

6. Select **Next**.

   If you selected the **Backup current galaxy** option, you will be prompted for the location of the file name of the backup .cab file.

   Important: It is highly recommended to back up your galaxy.

7. Confirm your application name and select **InTouchView Application**.

   The target resolution is set to the resolution of the template by default. The **Pixel** fields are disabled. This can be adjusted during application development.

8. Select **Next**.

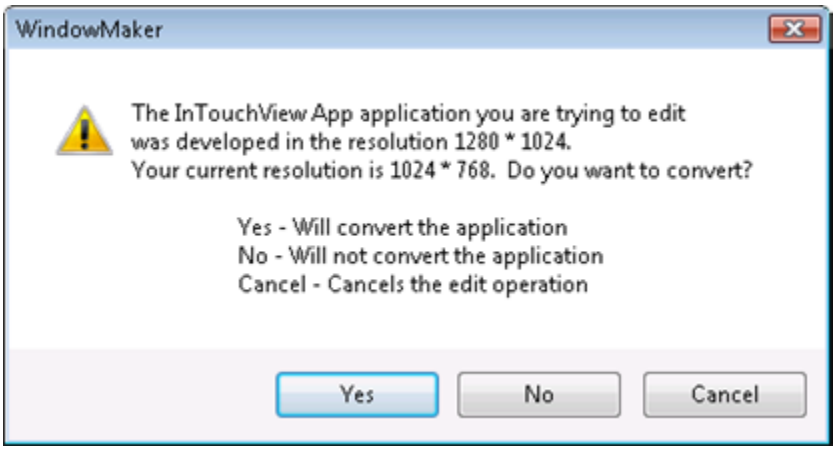
   When initialization completes, you can deploy your application.

Start WindowMaker from the IDE

You can edit a managed InTouch application by starting WindowMaker from within the IDE.

You can start WindowMaker from an InTouchViewApp template or an instance.

When opening a managed InTouch application created with a different screen resolution than your current system resolution, a message appears.



- Select **Yes** to convert the InTouch application to the current system resolution and open it.
- Select **No** to open and edit the InTouch application in its original resolution.

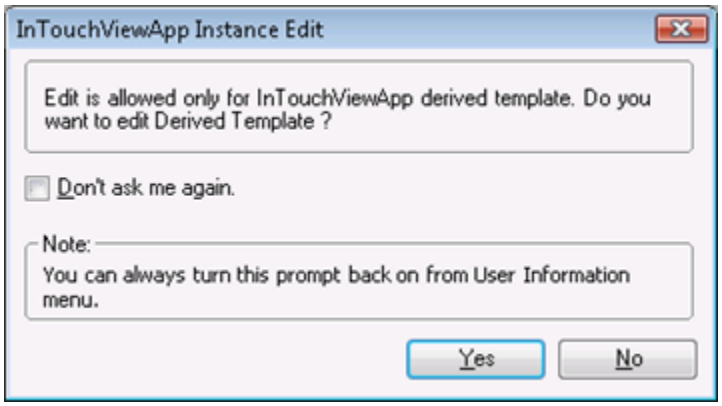
Start WindowMaker from an InTouchViewApp template

1. Open the IDE.
2. Locate the InTouchViewApp template that includes the managed InTouch application you want to modify.
3. Double-click the InTouchViewApp template. WindowMaker starts as the object's default editor and opens the InTouch application. You are ready to edit the managed application.

Start WindowMaker from an InTouchViewApp instance

1. Open the IDE.
2. Locate the InTouchViewApp object whose parent hosts the managed InTouch application you want to modify.

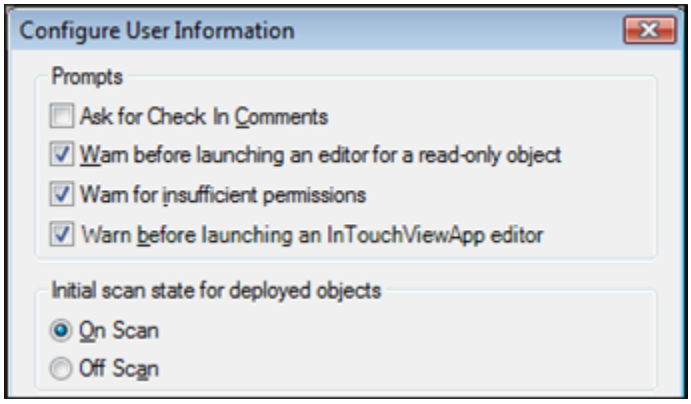
3. Double-click the InTouchViewApp object. The **InTouchViewApp Instance Edit** dialog box appears.



4. Do any of the following:
   - Select **No** to not open the template.
   - Select **Yes** to open the associated InTouchViewApp template in WindowMaker.

     If you enable the **Don't ask me again** checkbox and select **Yes**, the next time you open an InTouchViewApp instance, the managed InTouch application is automatically opened from the associated InTouchViewApp template.

     You can change this setting using the **Configure User Information** dialog box, which you open from the **Edit** menu.



Language switching for a managed InTouch application

For managed InTouch applications, certain language switching functionality is handled by the IDE instead of the InTouch HMI.

Use the IDE to:

- Export and import graphic text for translation.
- Configure the language used by the managed application.

For more information, see the Application Server User documentation.

Submit changes for an InTouch application

After you modify your managed InTouch application, you can submit the changes for deployment to the target nodes.

After you make changes to a managed InTouch Application, any derived InTouchViewApp objects appear with the Pending Changes icon.

This indicates that you must redeploy the changes to the target nodes, for WindowViewer on those nodes to reflect the changes.

For more information on how to deploy the changes to the operator nodes, see Deploy a managed InTouch application and Accept new application versions at the operator node.

Submit the changes for an InTouch application

1. Modify your managed InTouch application in WindowMaker as you would with a standalone InTouch application.
2. Save your InTouch windows.
3. On the **File** menu**,** select **Exit**. WindowMaker closes and the focus returns to IDE. The **Check In** dialog box appears.
4. Type a comment for check in, if necessary and select **OK**. The **Check In** progress dialog box appears.
5. Select **Close**.

Import an InTouch application

You can import an existing InTouch application to use as a managed InTouch application. You do this in two steps:

- Create an InTouchViewApp object to associate with the imported InTouch application.
- Import the InTouch application.

The imported InTouch application becomes a managed InTouch application.

The IDE creates a copy of the existing InTouch application in a folder that it manages. The IDE leaves the existing InTouch application and location unchanged.

If the version of the InTouch application you are importing is 6.0 or above, a conversion message appears. The application is converted before it is opened in WindowMaker. InTouch applications versions before 6.0 cannot be converted.

Import an InTouch application into a Galaxy

1. Open the IDE.
2. In the **Template Toolbox**, derive a template from the $InTouchViewApp base template.
3. Open the derived template. The **InTouchViewApp Initialization** dialog box appears.
4. Highlight **Import Existing Application** and select **Next**.
5. Locate the InTouch application. Do any of the following:
   - Select the ellipsis button to browse for the folder that contains the managed InTouch application.
   - To search for the application, select the **Find applications** checkbox. Specify a search root for the search

to begin and select **Find**. Select the InTouch application from the list.

6. Select **Next**.
7. If needed, type a new name in the **Application Name** box and a description in the **Description** box. The name and description appear in the Application Manager when the managed InTouch application is deployed.
8. Select the application target resolution different than the default screen resolution option. Options for this field are viewable as follows:
   a. Select the **Select target resolution** dropdown menu to view a list of predefined target resolutions.
   b. Select the **Select target resolution dropdown** menu and select **Custom**. The Pixel width and height fields will become editable.

   Note: If a different target resolution is selected when importing an existing application, then there will be no application resolution conversion from the source application resolution to the target resolution.
9. Select **Next**. The next panel appears and shows you the import progress.
10. Select **Close**. InTouch WindowMaker is started and you can edit the InTouch application as a managed InTouch application.

Importing Standalone InTouch Applications with Graphics

If the Standalone InTouch application being converted to a Managed application contains graphics, the application is imported, but the graphics are not imported.

Import the graphics contained within the Standalone application

1. Export the Standalone application as a template.
2. Save the template in the application template location.
3. In the IDE, create a new $InTouchViewApp derived template by selecting this created application template.

Import and export an InTouchViewApp object

You can import and export the InTouchViewApp object in the IDE. The InTouchViewApp object contains all information for hosting a managed InTouch application and can be used to exchange a managed InTouch application between Galaxies.

For more information about importing and exporting AutomationObjects, see the Application Server documentation.

Import an InTouchViewApp object

1. On the **Galaxy** menu, point to **Import**, and then select **Object(s)**. The **Import Automation Object(s)** dialog box appears.
2. Highlight the .aaPKG file that contains the InTouchViewApp object you want to import and select **Open**. The object and its managed InTouch application are imported.
3. Select **Close**.

Export an InTouchViewApp object

1. On the **Galaxy** menu, point to **Export**, and then select **Object(s)**. The **Export Automation Object(s)** dialog

box appears.

2. Highlight a folder to export to, specify an .aaPKG file name and select **Save**. The object and its managed InTouch application are exported.
3. Select **Close**.

Export and import tag data

You can export the tag data of a managed InTouch application to a .csv file. You can import this data into another managed InTouch application or a stand-alone InTouch application.

Export tag data from a managed InTouch application

1. Open the IDE.
2. Select the derived InTouchViewApp template that contains the managed InTouch application from which you want to export the tag data.
3. On the **Galaxy** menu, point to **Export**, and then select **DB Dump**.

   The **CSV File to Dump To** dialog box appears.
4. Specify a location and file name for the .csv file and select **Save**.

   A confirmation dialog box appears.
5. Select **Group output by types** if you want the tag data to be grouped by data types in the .csv file.
6. Select **OK**. When a success message appears, select **OK**.

Import tag data from a .csv file to a managed InTouch application

1. Open the IDE.
2. Select the derived InTouchViewApp template that contains the managed InTouch application to which you want to import the tag data.
3. On the **Galaxy** menu, point to **Import**, and then select **DB Load**.

   The **CSV File to Load From** dialog box appears.
4. Highlight the .csv file and select **Open**.

   During the import, one or more messages may appear warning you of duplicate names. Select the appropriate option for each duplicate tag.
5. When a success message appears, select **OK**.

Retain tag value and parameters

When you use multiple managed InTouch applications on one node, all applications use the same directory. If you configure tag data or parameters to be retained and you switch applications, the run-time data is lost.

Configure retention in a managed InTouch application

1. Open the managed InTouch application with WindowMaker from the IDE.
2. On the **File** menu, point to **Configure**, and then select **WindowViewer**.

   The **WindowViewer Properties** screen appears.

3. Select the **Managed Application** tab.
4. In the **Local Working Directory** box, type the name of a unique existing directory. This is the name of the directory on the target run-time node. You can also use special characters to specify the local working directory path. The following table illustrates the special characters that you can use:

Special Character	Local Working Directory Path
%SystemDrive%	System drive. For example, "C:"
%USER%	Name of the logged-on user. If WindowViewer is running as a service, the application is copied to "All Users"
%APPDATA%	Logged-on user's Application Data. If WindowViewer is running as a service, the application is copied to "All Users" Application Data
%ITUSER%	Logged-on user. If WindowViewer is running as a service or in a console session, the application is copied to "All Users"
%ITAPPDATA%	Logged-on user's Application Data. If WindowViewer is running as a service or in a console session, the application is copied to "All Users" Application Data

Delete a managed InTouch application

You can delete an InTouch application in the IDE by deleting the associated InTouchViewApp template.

When you do this, the template and the InTouch application directory associated with that template are deleted completely.

You can only delete an InTouch application if the associated InTouchViewApp object does not have any derived instances.

Deleting an InTouchViewApp instance does not delete the associated InTouch application.

Delete the InTouchViewApp template

1. Open the IDE.
2. Select the InTouchViewApp template that contains the managed InTouch application you want to delete.
3. On the **Home** menu, in the **Edit** menu, select **Delete**.

   The **Delete** dialog box appears.
4. Select **Yes**.

   The InTouchViewApp template and the associated InTouch application folder are deleted.

Associate all Galaxy graphics in an InTouchViewApp

Associating all Galaxy graphics with an InTouchViewApp template enables deployed and published InTouch applications to execute "show graphic" requests made of any graphic in the Galaxy without having to embed them in the application.

- The ShowGraphic() function uses the graphic as a parameter. Associating all Galaxy graphics ensures that the graphic is deployed and available at run time whether or not it is referenced by InTouchViewApp.
- Associating all Galaxy graphics ensures that template symbols referenced by the ShowGraphic() function are deployed and available at run time.

Note: The term "graphic" includes any symbol or client control present in the Graphic Toolbox, and any symbols owned or inherited by templates and instances.

For more information about associating all Galaxy graphics with an InTouchViewApp, see Application Server documentation.

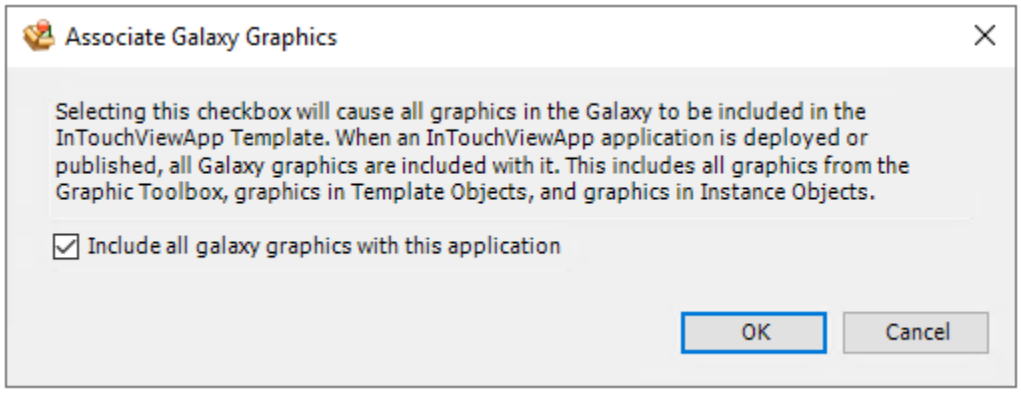
Access the "Include all Galaxy graphics" option through the InTouchViewApp right-click context menu.

Include all Galaxy graphics with your InTouchViewApp template

1. Right-click the InTouchViewApp template you wish to configure.

   The InTouchViewApp context menu appears.
2. Select the **Associate Galaxy Graphics** menu item.

   The **Associate Galaxy Graphics** dialog box appears.



   When the **Associate Galaxy Graphics** dialog appears, the $InTouchViewApp template is checked out.

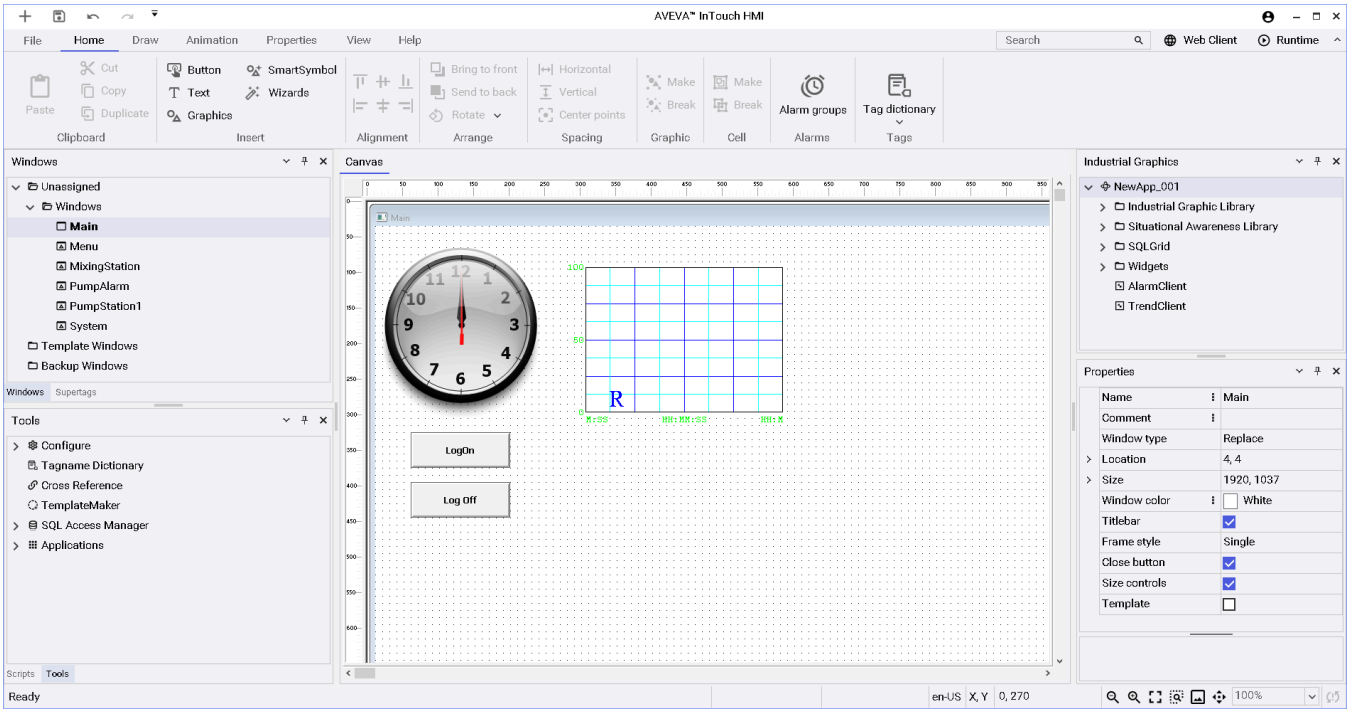
   If the $InTouchViewApp template is not yet initialized or checked out, the **Include all Galaxy graphics with this application** checkbox is not available.
3. Enable the checkbox and select **OK**.

   A status box displays check in progress.

   If an InTouchViewApp instance is deployed when you attempt to modify the **Include all Galaxy graphics** setting, you will see an information message to undeploy all deployed instances.

Windows

An application window is a container for one or more graphics that model your production processes. For example, you can have a window that shows equipment in a unit. Another window can show a grid of alarm information related to that unit.



You can create any number of windows and you can define window properties such as background color, screen position, window title, and so on.

Create application windows

When you create a new application window, the default settings reflect those of the previously created or of the currently active InTouch window. The following settings will reflect those of the previously created window:

- Window names: can have up to 32 characters and cannot include any special characters except dollar ($), pound (#) and underscore(_). When you create a window, you are only required to provide a window name. All other items are optional.

  Note: If the window name contains unsupported characters, an underscore (_) replaces each unsupported character including the space character. If an application is migrated, editing the window or modifying any of the window properties will replace any special characters in the window name to underscore (_). This is applicable for all types of windows including application, frame and template.

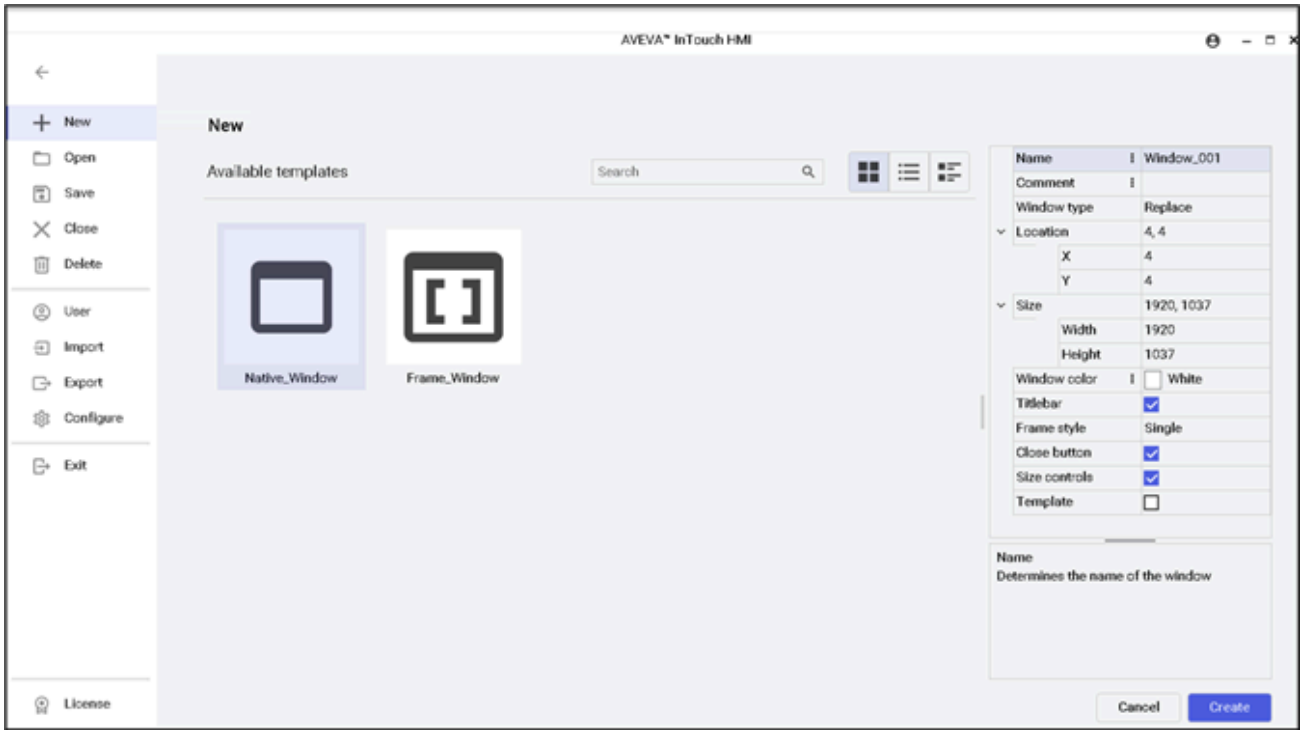
- Comments: can be included for a window, but is for information purposes only. The comment appears in file listings but is not used by the application.
- Window dimension values: by default, the window dimension values are set to the dimensions of the previously created window. These values are also automatically modified if you manually change the window size by dragging the window border.

Note: If you have specified an application target resolution that is different than your screen resolution, your application windows can exceed the canvas boundary. However, only windows within the target resolution canvas boundary will display at Run Time.

Create a new window

1. On the **File** menu, select **New**.

   The **New** window displays the available templates to create a new Native Window and a Frame Window.



2. Select the type of window you want to create from the available templates.

   Alternately, In the Search box, enter the template name based on which you want to create the window, and press Enter

3. To configure the view layout of the templates, select **Icon**, **List**, or **Detail**.
4. Use the Windows Properties panel to configure the basic window properties.

   The Windows Properties is classified into three categories: Appearance, Layout, and Window Style.

5. To configure the Appearance of the new window:
   - In the **Name** textbox, type a unique name that identifies the window.
   - In the **Comment** textbox, type any comments you want to associate with the window. The comment must be 60 characters or fewer.
   - Select the **Window Color** button to select the background color for the window.
   - Select **FrameStyle** for the window to display a border around the window.
     - Select **Single** for a single, one-dimensional border.
     - Select **Double** for a three-dimensional bordered window.
     - Select **None** for a window with no border.

       Select the ellipses next to the **Name, Comment, or Window Color** fields to do one of the following:

- Reset - Returns the field value to its default value.
- Edit - Allows to modify the details of the field.
- Sort - Sort the Windows Properties fields in Alphabetical, Categorized, or Categorized Alphabetical order.
- Show Description - Select to show/hide description
- Show Toolbar - Select to show/hide toolbar.

6. To configure the layout of the new window:
   - From the **Window Type** dropdown, select the options to see how the window opens at run time.
     - Select the Window Type as **Replace** for the window to automatically close any windows it intersects with when it appears on the screen.
     - Select the Window Type as **Overlay** for the window to appear on top of currently open windows. It can be larger than the window(s) it is overlaying. When an overlay window closes, any windows behind it reappear. Selecting on any visible portion of a window behind an overlay window brings that window to the foreground as the active window.
     - Select the Window Type as **Popup** for the window to always stay on top of all other windows. Popup windows usually require a response from the user to be removed.
   - Expand the **Location** field to specify the window location and dimensions.
     - In the **X** box, type the number of pixels between the left edge of the design area and the left edge of the window being defined.
     - In the **Y** box, type the number of pixels between the top edge of the design area and the top edge of the window being defined.
   - Expand the **Size** field to specify the window Height and Width in pixels.
7. To configure the Window Style of the new window:
   - Select the **Close Button** checkbox to include a button on the title bar to close a window at design time or run time. Windows retain the close window button even after they are exported to other applications.
   - Select the **Maximize Button** checkbox to include a button on the title bar to maximize a window at design time or run time.
   - Select the **Minimize Button** checkbox to include a button on the title bar to minimize a window at design time or run time.
   - Select the **Size Controls** checkbox to enable users to resize a window in WindowMaker.
   - Select the **Title Bar** checkbox to include a title bar.
   - Select the **Template** checkbox to save the settings as a template for later use.
   - From the **WindowState** dropdown, select the default state of the window - Normal, Minimized, or Maximized.
8. Select **OK**.

Set an application window as a template window

You can set an application window to be a template window using the Window Properties. This will allow you to reuse windows and reduce configuration time.

Set an application window as a template window

1. Open the **Window Properties** pane by doing one of the following:
   a. To set a new window as a template window, on the **File** menu, select **New.**
   b. To set an existing window as a template window, on the Windows pane, select the existing window.
   The Properties pane for the window displays on the right side of the screen.
2. Select the **Template** checkbox in the **Window Properties** pane.

3. Select **OK**.
   The window now appears in the Template Windows folder under the Windows pane. It is also available in the Template Windows Browser.

Note: You can change a template window back to an application window by unchecking the Template checkbox in the Window Properties pane. You can also drag and drop windows between the Template Windows folders to change this property.

Create an application window from a template window

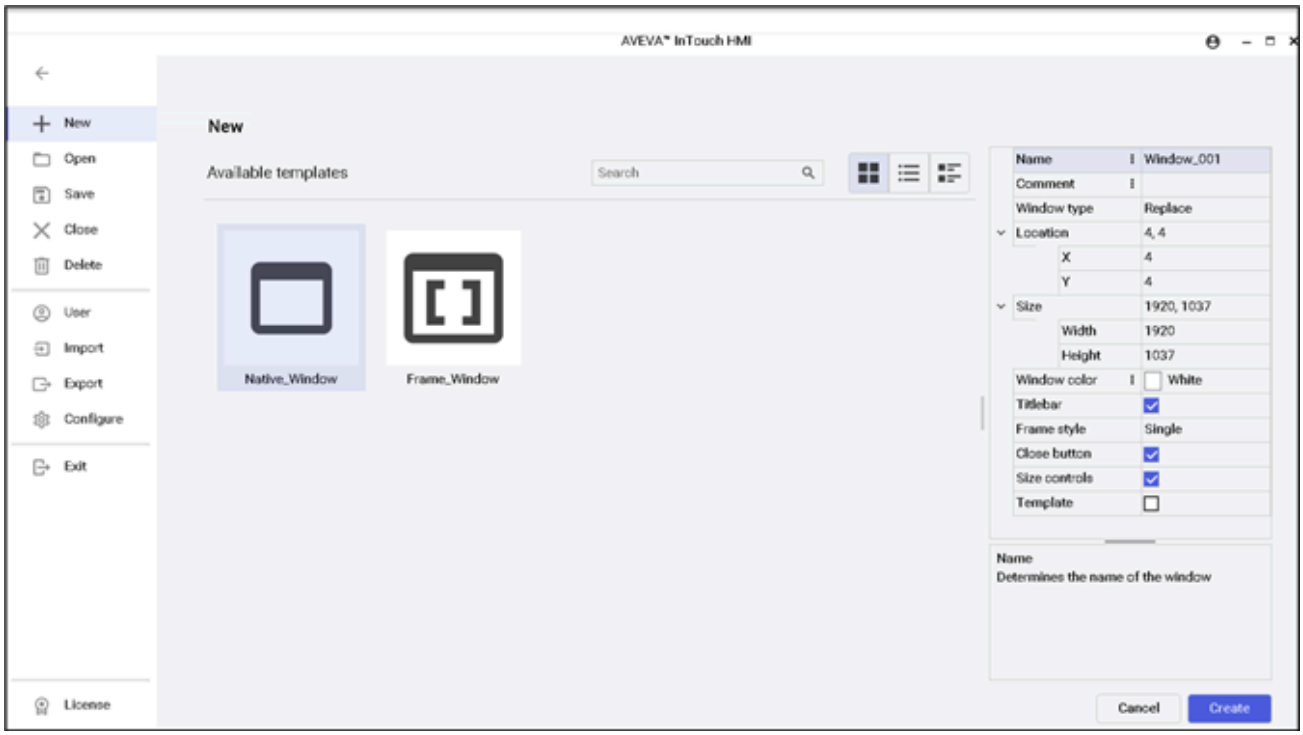
You can create an InTouch application window from any of the template windows available in the Template Windows Browser.

Note: Only windows with the Template property selected will be available in the Template Windows Browser. See Set an application window as a template window for the detailed procedure.

Create an application window from a template window:

1. On the **File** menu, select **New**.

The **Available templates** are displayed.



2. Browse the template window thumbnails and select your window.

   Note: Each template window is shown in thumbnail view by default. The window thumbnails are shown proportional to the actual window location in design time with respect to the whole application.

3. Select **OK**.

   Note: If the window name of a template window from a migrated application contains unsupported special characters, an underscore (_) replaces each unsupported character including the space character in the new application window.

Work with frame windows

If you are working in a Standalone or Managed InTouch application, you can create and develop frame windows in addition to or in place of application windows. Frame windows enable you to host Industrial Graphics that can support pan, zoom and touch capabilities.

Frame windows support most of the same properties as application windows. However, several limitations apply.

Frame windows do not:

- support multiple Industrial Graphics
- host native InTouch controls
- host ActiveX controls
- host SmartSymbols
- support undo/redo actions for window components
- flip or rotate actions for an embedded symbol
- support cut, copy, paste or duplicate operations

- provide cross-reference support for Industrial Graphics embedded in frame windows
- support window conversion to Industrial graphics
- support creating new instances, editing instances or selecting alternate instances of embedded automation object graphics

Create a frame

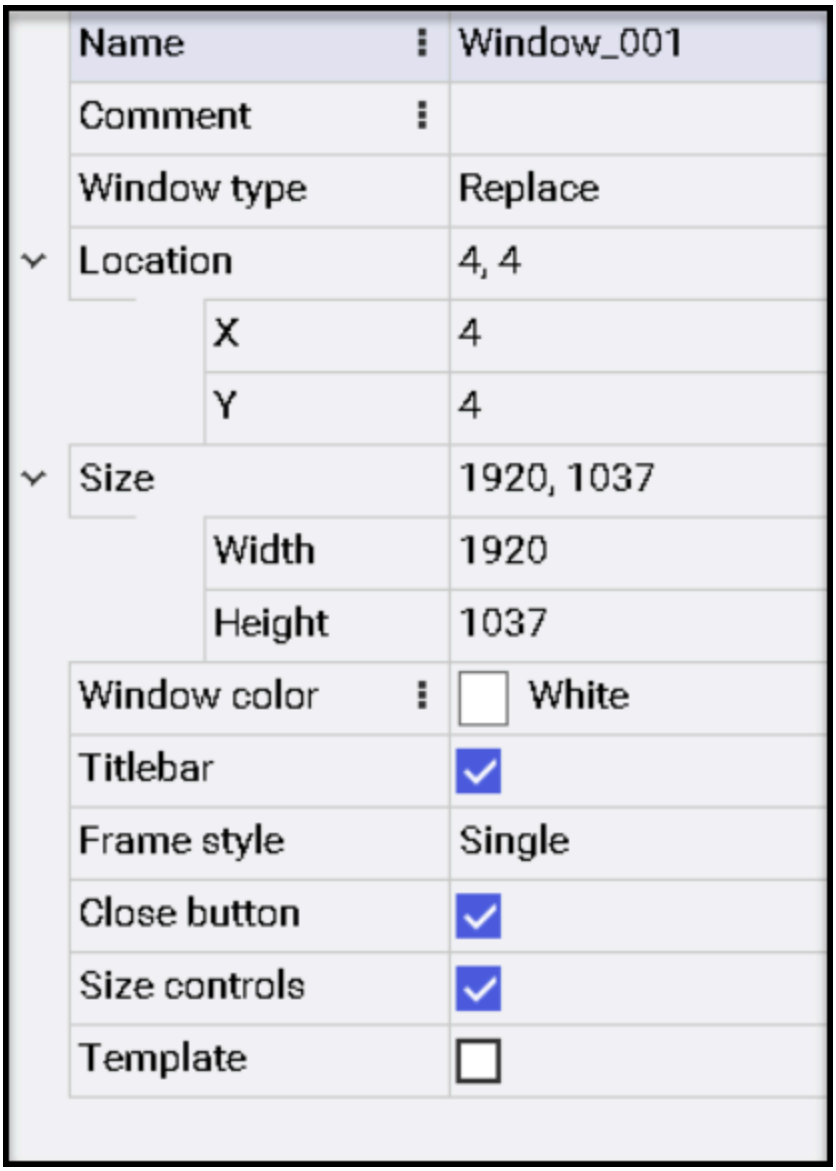
1. On the **File** menu, select **New**.

   The **New** window with the list of **Available templates** appears.

2. Select the **Frame** window.
3. Configure the window properties and select **OK**.

Use the Properties panel

You can edit frame properties directly from the Properties panel on the WindowMaker canvas. The frame properties will populate in the Properties Grid upon creating the window:



The Industrial Graphic Toolbox and the Properties Grid appear to the right of the canvas by default. You can dock the Graphic Toolbox or the Properties Grid to the left side of the canvas. To restore WindowMaker to the default, select **View**, **Restore Layout**.

Note: Application window properties will not populate in the **Properties Grid**.

Frame windows support the same properties as application windows, with three additional properties. They are described below.

See Create application windows for a full description of each window property.

Frame Property	Functionality
MaximizeButton	Enables the Maximize button in the upper right of the frame. Default is **False**.
MinimizeButton	Enables the Minimize button in the upper right of the

	frame. Default is **False**.
WindowState	Displays the initial state of the window: **Normal**, **Minimized** or **Maximized**. Default is **Normal**.

Changes to the frame properties will immediately reflect in the frame, with the exception of the **FrameStyle** property. To configure this property, the **SizeControl** and **TitleBar** properties must first be set to **False**.

The **FrameStyle** dropdown menu will become enabled.

Work with graphics embedded in frame windows

Industrial Graphics can be embedded in frame windows the same way as application windows. The Industrial Graphic Toolbox is integrated into both the System Platform IDE and InTouch WindowMaker. The Graphic Toolbox includes separate folders containing the Industrial Graphic Library and Situational Awareness Library of predefined symbols. The Industrial Graphic Library contains realistic symbols of standard industrial objects.

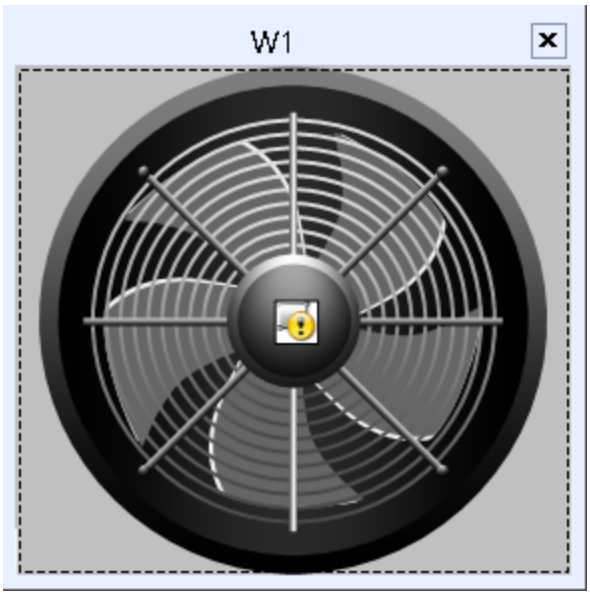
Situational Awareness Library symbols have a simplified look and provide minimum visual detail to efficiently convey their functional purpose and status without showing irrelevant information to operators. Most Situational Awareness Library symbols are designed as Symbol Wizards that incorporate multiple visual and functional configurations in each symbol. Selecting a functional configuration of a symbol is a simple matter of selecting options from the symbol's Wizard Options.

Do either of the following to embed a symbol in a frame:

- after creating a Frame Window, select the Embed Graphic button  present on the canvas and browse for a graphic.
- after creating a Frame Window, select the Create Graphic button  present on the canvas to create a graphic in the Industrial Graphic Editor and save the graphic. The graphic created will be embedded in the window.
- drag an Industrial Graphic from the Graphic Toolbox and drop it onto the frame
- select Embed Industrial Graphic toolbar button  and browse for a graphic
- embed graphics using the context menu from the frame window

You can automatically fit a frame to an embedded Industrial Graphic. To fit the frame to the embedded symbol, right-click the window select **Fit to Symbol**.

The frame will resize to fit the symbol:



Note: Undo is not supported for **Fit to Symbol**. If you manually change the dimensions of the frame window or changes the Size property, you need to perform the **Fit to Symbol** action again.

When you embed an Industrial Graphic in the frame, the symbol will populate in the **Properties Grid** dropdown menu as shown below:

You can toggle between the frame window and symbol properties. Do any of the following:

- Select the symbol or frame window name from the dropdown menu
- Right-click the window frame and select **Window Properties**.

- Select the title bar or frame border to display frame window properties in the grid.

  Select the embedded symbol to display the symbol properties.

You must configure the symbol properties to enable pan and zoom functionality using keyboard, mouse and touch gestures. The configurable symbol properties are listed below.

Symbol Property	Functionality
MaintainAspectRatio	Maintains the symbol's aspect ratio when the modern frame is resized. Default is **True**.
SymbolName	Sets the symbol name being hosted by the modern frame
InteractionMode	Sets the interaction mode. Options are: • **None**: disables Pan and Zoom • **PanZoom**: default, enables Pan and Zoom
ShowZoomControl	Shows the symbol and Zoom control. Options are: • **Auto**: control is shown as required • **Visible**: control is always shown

For details on using pan and zoom capabilities at run time, see Pan and Zoom in *AVEVA™ InTouch HMI Creating Standards for InTouch HMI Components*.

Modify application windows

When developing your application, you can modify the properties of windows at any time.

Modify properties of an application window

1. Right-click the window name, and select **Properties.**
2. In the **Window Properties** panel, make your changes.
3. Select **OK**.

For more information about the window options, see Create application windows.

Open, save, and close windows

Opening a Window

While developing your application, you can open as many windows as your computer memory supports.

Open a window

1. On the **File** menu, select **Open**.

   The **Select windows to open** dialog box appears listing the names of all windows in your application.
2. Do either of the following:
   - To open a single window, double-click the window name.
   - To open multiple windows, select the checkboxes for the windows to open and then select OK.

Saving a Window

When you save a window, all graphics, QuickScripts, properties, and so on associated with the window are also saved.

Save a window

1. On the **File** menu, select **Save**.

   The **Select windows to save** dialog box appears, listing the names of all windows.
2. Select the windows that need to be saved.
3. Select **OK**.

Closing a Window

When you close a window that has been modified, you are prompted to save your changes.

Close a window

1. On the **File** menu, select **Close**.

   The **Select windows to close screen** dialog box appears listing the names of all currently open windows.
2. Select the checkbox next to the window name.
3. Select **OK**.

View a thumbnail preview of a window

You can view the thumbnail preview of a window in WindowMaker. This is especially useful if you have an application with many windows. You can then view the thumbnail preview of a window to check if it is the right one before opening it.

You can view the thumbnail preview of a window. The thumbnail preview is available and updated only after you have saved a window. If you migrate an application or import a window from an XML file, you can view the thumbnail preview without having to explicitly save the window. If you import a window from another application, you can view its thumbnail preview without having to explicitly save the window. However, if you have modified the window after importing it, you must save it to view the changes in the thumbnail.

You can also update the thumbnail of all windows in an InTouch application. You must close all windows before you can update the thumbnail.

Update the thumbnail of all windows

- On the **View** menu, in the **Update** group, select **Window Thumbnails**.

Update the thumbnail of a particular window

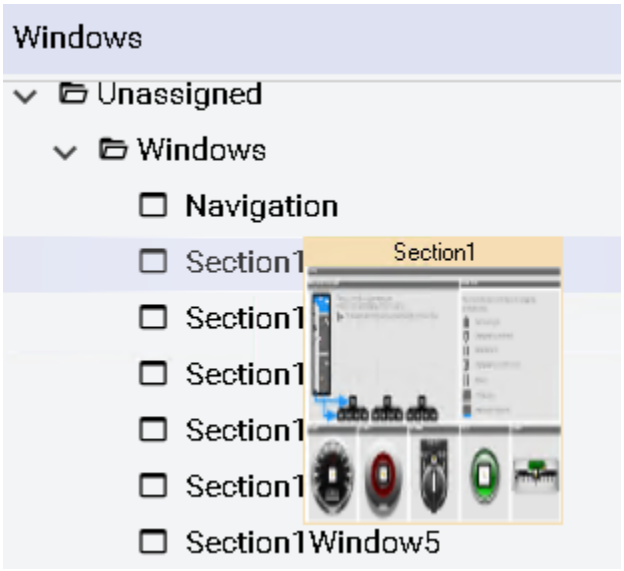
1. Right-click the window, and then select **Update Thumbnail**.

   A confirmation message appears.
2. Select **Continue**. The thumbnail of all windows in the InTouch application is updated.

   The system will open and close all windows to update the thumbnail.

View a thumbnail preview of a window

1. Open WindowMaker.
2. Open an application and save a window.
3. Move the pointer over the rectangle area that covers the window name and the associated icon. The thumbnail preview appears.



Duplicate windows

When you have very similar processes to simulate and control, you may want to duplicate a window and then customize it for a secondary process or unit.

You can duplicate windows with all graphics, QuickScripts, properties, and so on, associated with the window.

Before you start, the window that you want to duplicate must be open and saved at least one time. You can only duplicate one window at a time.

Duplicate a window

1. In the Windows pane, right-click the window and select **Save As**.

   The **Save Window** dialog box appears.

2. In the **New Name** box, type a name for the new window.
3. Select **OK**.

Note: If the original window name contained unsupported characters, an underscore (_) character will replace each unsupported character in the duplicate window name.

Delete windows

To conserve computer storage space, or if the list of windows in the Application Explorer becomes too long to manage, you can delete unused windows.

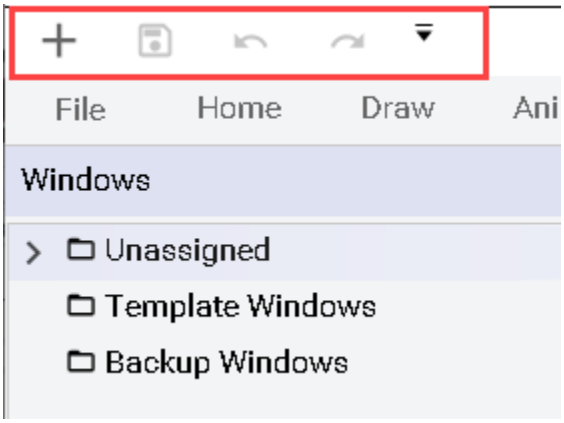
Caution: Make sure you delete the correct window. You cannot restore a deleted window with **Undo**.

Delete a window

1. On the **File** menu, select **Delete**.
   The **Select windows to delete** screen appears.
2. Select the window name you want to delete and select **OK**. When the message appears, select **Yes**.
3. Select **OK**.

Use the Quick Access Toolbar

The Quick Access Toolbar is a customizable toolbar that contains a set of predefined or commonly used commands for quick access from any screen. By default, the Quick Access Toolbar is located in the upper left-hand corner of the WindowMaker screen.



Executing commands from the Quick Access Toolbar

You can execute the following commands from the quick access toolbar.

Note: While most of the commands on the quick access toolbar are also available in other ribbon groups, a few commands are only available from the quick access toolbar.

Quick Access Toolbar Commands	Description
New	Create a new window.

Open	Open an existing window
Save	Save a window
Save As	Create and save a copy of a window by specifying a different name and location
Save All	Save all open windows
Close	Close an open window
Delete	Delete a window
Import	Import a window
Export	Export a window
Print	Print a window
WindowViewer	View a window in WindowViewer
Convert to Industrial Graphics	Convert an existing graphic to Industrial Graphics
Undo	Cancels the last action
Redo	Repeat the last action
Exit	Exit the WindowMaker

Customizing the Quick Access Toolbar

By default, the Quick Access Toolbar displays the following commands: New, Save, Undo, Redo

You can choose to add or remove a command from the list to be shown on the Quick Access Toolbar.

Add a command to the Quick Access Toolbar:

1. Select the down arrow on the Quick Access Toolbar.
2. Select the command to add.

Remove a command from the Quick Access Toolbar:

1. Select the down arrow on the Quick Access Toolbar.
2. Highlight the selected command.

Positioning the Quick Access Toolbar

By default, the Quick Access Toolbar is positioned above the ribbon.

- To position the Quick Access Toolbar below the ribbon, choose the down arrow on the Quick Access Toolbar and select **Show Below the Ribbon**.

- To position the Quick Access Toolbar above the ribbon, choose the down arrow on the Quick Access Toolbar and select **Show Above the Ribbon**.

Using the Quick Access Toolbar to resize the ribbon

You can choose to minimize or maximize the ribbon from the Quick Access Toolbar.

- Minimize the Ribbon: This view shows only the menu items and not the groups. To minimize the ribbon, choose the down arrow on the Quick Access Toolbar, and select **Minimize the Ribbon**.
- Maximize the Ribbon: This view shows the menu items and the groups. To maximize the ribbon, choose the down arrow on the Quick Access Toolbar, and select **Maximize the Ribbon**.

Print information about InTouch windows

You can print the following information for InTouch windows:

- Details for the graphical objects placed in a window. For example, the window location for all types of objects, the font used for a text object, the custom properties defined for an Industrial graphic, and so on.
- Details about the animation links used in a window.
- Scripts associated with a window.
- Tags used in a window.

You can print details for graphical objects to either a printer or an .html file. An .html file is created for each window. The .html file contains:

- A picture of the window as a referenced .png file.
- A listing of details about each graphical object in the window.

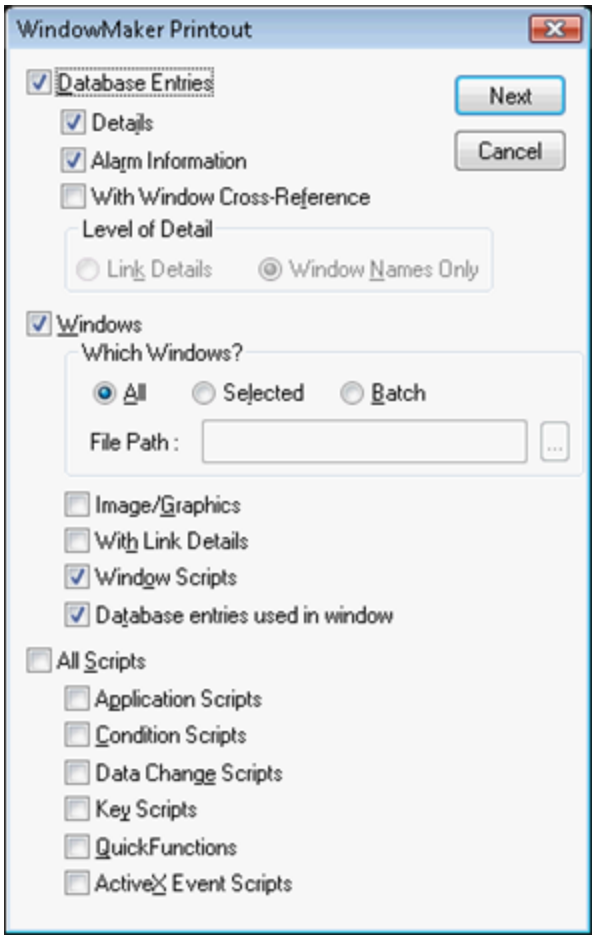
If you print graphical object details for multiple windows, a summary .html file is created that has links to the window-specific .html files.

For animation link, script, or tag details, you can print to either a printer or a .txt file.

Print information about InTouch windows

1. Open an InTouch application in WindowMaker.
2. On the quick access toolbar, select **Print**.

   The **WindowMaker Printout** dialog box appears.



3. Select **Windows**.
4. Select the windows to print:
   - **All** prints the information for all windows in the application, including names of any embedded graphics.
   - **Selected** prints only the information for specific windows, including names of any embedded graphics on the selected windows. The **Windows to Print** dialog box appears. Select the windows in your application you want to print and select **OK**.
   - **Batch** prints only the information for windows specified in a .csv file, including names of any embedded graphics.

     For details on the .csv format, see.CSV format for printing windows.
5. Select what you want printed for the selected windows:
   - **Image/Graphics** prints the information about all of the graphic objects placed in the windows.
   - **With Link Details** prints the link details for the windows.
   - **Window Scripts** prints the scripts associated with the windows.
   - **Database entries used in window** prints the tags used in the windows.
6. Select **Next**. The **Select Output Destination** dialog box appears.
7. Do one of the following:
   - Select **Send output to Printer** to print the information.
   - Select **Send output to Text File** to create a single .txt file.

- Select **Send output to HTML File** to create a summary .html file and a single .html file and .png file for each window you specified. If .html and .png files exist, they are automatically overwritten.

8. Select **Print**.

Print windows information from a command prompt

You can print window information from a command prompt. To do this, you create a .csv file that contains the window names and then reference the .csv file in the print command.

When the print command runs, WindowMaker automatically opens the default InTouch application, performs the print operation, and then closes. Printing from a command prompt only works for stand-alone InTouch applications.

Print window information from a command prompt

1. Create the .csv file that contains the names of the windows to print.

   For details on the .csv format, see .CSV format for printing windows.
2. Exit WindowMaker and close the .csv file.
3. Open a command prompt.
4. Type the command to print the information.

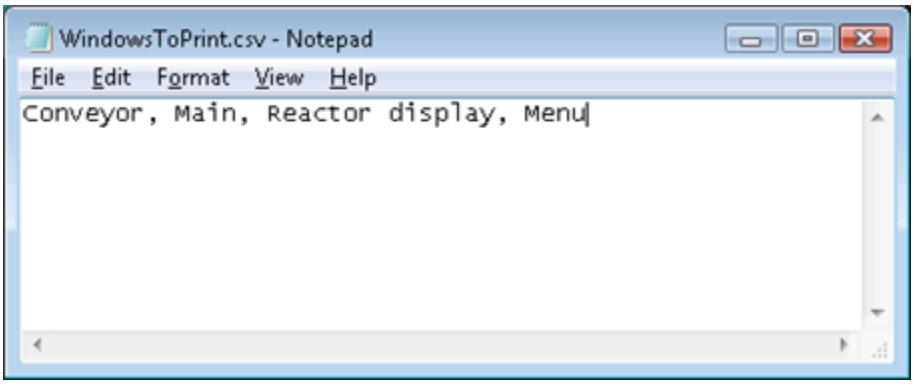
   For details on the command syntax, see Syntax to print from a command prompt.
5. Press the **Enter** key. WindowMaker starts and prints the information.

.CSV format for printing windows

You can use a .csv file to use your window information to multiple formats, such as Microsoft Excel. Create a single row in a .csv file and include each window name separated by a comma:

Window1,Window2,Window3,WindowN

For example:



If you use Microsoft Excel to create the file, there should be one row, with the name of each window in each column cell of the row.

You cannot use the backslash character (\) in a window name.

You must use a comma and not any other separator.

Syntax to print from a command prompt

The command syntax is:

`"<WindowMaker path name>" COMMANDFILE="<CSV file>" ALL OUTPUTTARGET=<target name>`

where:

`<WindowMaker path name>` is the path to the WindowMaker application (WM.exe).

- `<CSV file>` is the name of the .csv file that specifies the windows to print.
- `<target name>` is output location, either to a printer or .html file.
- `ALL` is the command to print all link, database entry, and script information. If you do not include the ALL command, only graphical object information prints.

In this example, all link, database entry, and script information is sent to the default printer:

```
"C:\Program Files\Wonderware\InTouch\wm.exe" COMMANDFILE="D:\print.csv" ALL OUTPUTTARGET = PRINTER
```

In this example, graphical object information is sent to an .html file:

```
"C:\Program Files\Wonderware\InTouch\wm.exe" COMMANDFILE="D:\print.csv" OUTPUTTARGET = HTML <DEMOAPP.html>
```

Configure the depth of the context menu

The depth of the context menu, also known as the Windows shadows or the shadow of the context menu, creates borders for each floating window on the screen. This helps to distinguish one window from the other overlapping windows.

You can configure the depth of the context menu for the InTouch applications in two ways:

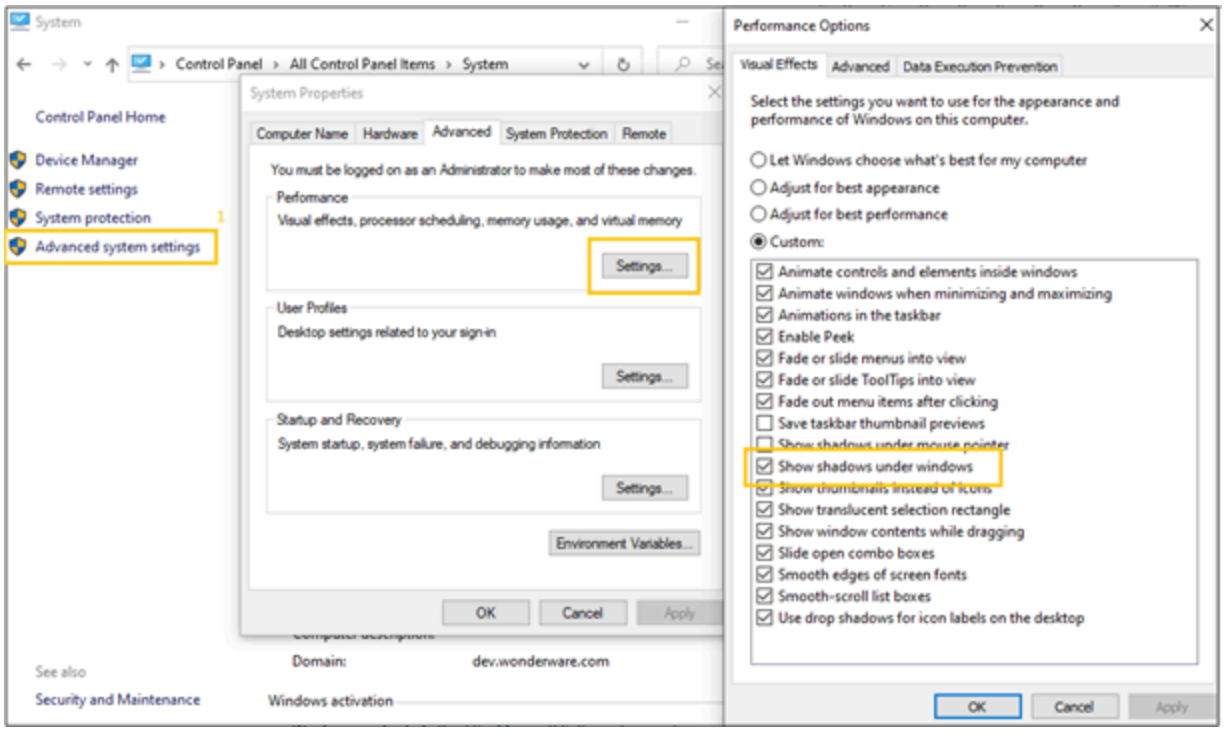
- Modifying the System Properties in the Control Panel
- Modifying the Registry entry in the Registry Editor

Configure the Depth of the Context Menu using Control Panel

1. Open the Control Panel.
2. Highlight **System**, and select **Advanced Settings**.

   The **System Properties** dialog box appears.
3. In the **Advanced** tab, under the **Performance** section, select **Settings**.

   The **Performance Options** dialog box appears.
4. Select the **Show shadows under windows** checkbox.



Configure the Depth of the Context Menu Using Registry Editor

1. Open the Registry Editor.
2. Navigate to HKEY_CURRENT_USER\Software\Microsoft\Windows\DWM.
3. Set the value ColorPrevalence" -Type DWORD to 1.
4. Restart the WindowMaker.

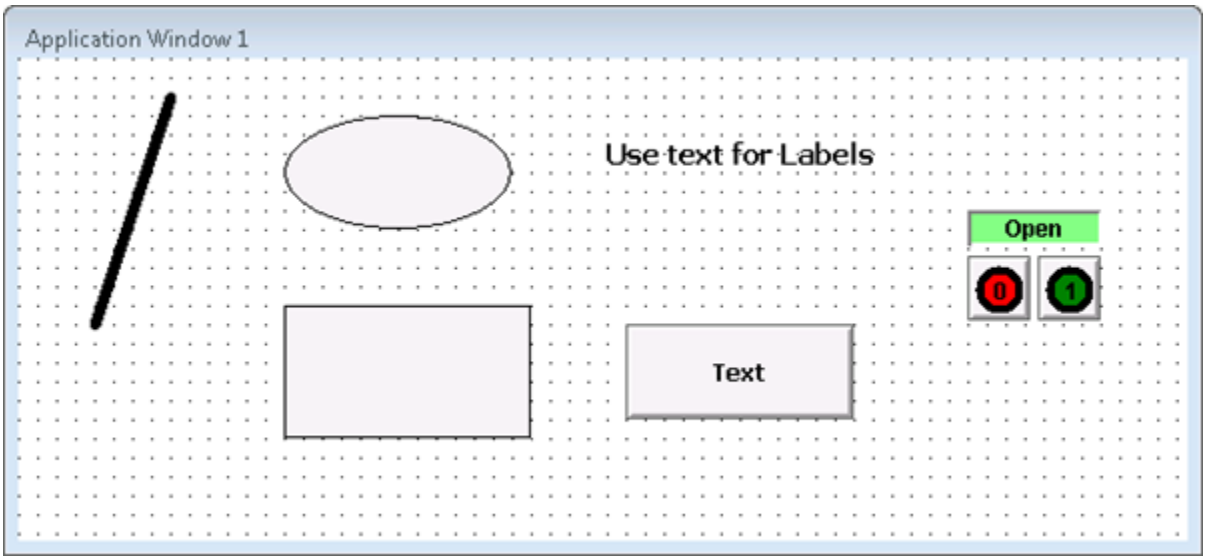
Graphic elements

This section includes:

WindowMaker objects

Graphical objects are the key parts of the human machine interface (HMI) applications you build.

As you build your applications, you create simple objects, combine simple objects to make more complex objects, and use some pre-defined complex objects.



The individual elements of a complex object are usually grouped together to:

- Prevent a complex object from breaking apart while editing.
- Duplicate the entire object.
- Assign a common set of properties to its individual elements.

You can use Industrial graphics created with the Industrial Graphic Editor in your managed or Advanced InTouch applications. You can also add Industrial Graphics directly from WindowMaker's Industrial Graphic Toolbox. For more information about working with Industrial and Situational Awareness Library symbols, see *Industrial Graphic Editor User* documentation or WindowMaker help.

Simple objects

You can create the following types of simple objects:

- Lines
- Shapes
- Text
- Buttons

Each simple object has attributes that control how it appears:

- Line color and weight
- Fill color
- Height
- Width
- Orientation

Create lines and shapes

The following table describes how to do basic drawing tasks. The drawing buttons are located on the **Draw** menu.

To draw this	Select this	Button
Line	Line button	
Horizontal or vertical line	H/V Line button	
Rectangle	Rectangle button	
Rectangle with rounded corners	Rounded Rectangle button **Note:** To adjust the radius of the rounded rectangle corners, see Change the radius of a rounded rectangle.	
Circle or an ellipse	Ellipse button. Press and hold the SHIFT key to draw a circle.	

Create buttons

You can use buttons to create points of interaction with your application. The process is very similar to creating simple drawing objects.

For information about creating polygons, see Create polylines and polygons.

Create a button

1. On the **Draw** menu, in the **Insert** group, select **Button**.
2. Select and drag to place and size the button.
3. Edit the default button text. Do the following:
   a. Right-click the button and select **Substitute Strings**.
   b. In the **New String** box, type the text for the button.
   c. Select **OK**.

Create polylines and polygons

Drawing polylines is slightly different than drawing lines.

Create a polyline or polygon

1. On the **Draw** menu, in the **Shapes** group, select **Polyline** or **Polygon**.

2. Select the application window to set the first point.
3. Select the application window again to set more points to define your polyline or polygon.
4. Double-click the last point.

Create text

You can use text to label visual items in your application.

When you create text, the text formatting settings match those set in the WindowMaker configuration screen. You can change the appearance of selected text. For more information, see Change text appearance.

When you type multiple lines of text, they become objects which can be moved and edited independently. You can also combine text objects into a symbol and edit them as a group.

Create text

1. On the **Draw** menu, in the **Insert** group, select **Text**.
2. Select the place for the text to start.
3. Type your text and press ENTER. A new line of text appears.

Complex objects

Complex objects provide more functionality than simple objects. The following table describes the different types of complex objects.

Complex Object	Description
Cell	A group of two or more objects, including symbols or other cells, that are joined together to form a single unit. You can use cells to create virtual devices such as slide controllers. Cells are useful for creating multiple devices to be associated with different tags.
Symbol	A group of simple objects, such as lines, shapes, and text, that joined together and treated as a single object. Any attribute change applied to a symbol affects all the component objects of a symbol. Symbols cannot contain bitmaps, buttons, cells, wizards, or trends.
SmartSymbol	An InTouch cell that has been converted into a reusable graphic template. You can place one or more instances of a SmartSymbol template in your application windows. Any change to a template propagates to the instances. For more information, see About SmartSymbols.

Complex Object	Description
Industrial Graphic	A highly versatile graphic created using the Industrial Graphic Editor in the Integrated Development Environment (IDE).
Bitmap Container	Objects that enable you to import images such as photographs, drawings, and screen shots. You can rotate a bitmap and you can give it a transparent background. For more information, see Work with bitmap containers.
Trend Objects	Charts of real-time or historical data value changes of multiple tags over time. For more information, see Trend objects.
Wizard	Pre-built object that you only need to select, place, and configure for your application. For more information, see Wizards.
ActiveX Control	Software component that runs within your application. WindowMaker supports both AVEVA and third-party ActiveX controls. For more information, see Use ActiveX controls.

Cells and symbols

You can combine multiple objects into two different types of single units: cells and symbols. A cell can contain any object. A symbol can only contain simple objects. Symbols cannot contain cells.

To find out if a specific object is a cell or a symbol, double-click the object.

- A cell opens, either the **Substitute Tagnames** dialog box, or if the cell doesn't contain tagnames, the Substitute Names warning message appears.
- A symbol or simple graphic object opens the **Animation Links Selection** dialog box.

About cells

Use a cell to combine and maintain a fixed spatial relationship among multiple elements. You can also use a cell to move multiple elements around and align with other graphical elements.

To change the elements of a cell, you must break the cell, change the elements, and then combine the elements into a cell again.

You can animate elements of a cell but you cannot animate a cell. You also cannot resize a cell.

About symbols

You can animate a symbol and simple objects. You can also use a symbol to animate parts of a complex graphic.

You cannot make a symbol if more than one of the selected objects has links.

If you combine two symbols into a new symbol, the original symbol structure is lost. If you break the new symbol, it is broken into the individual components of each original symbol. The two original symbols are lost.

Group objects to cells

You can combine symbols, bitmaps, trends, buttons, wizards and other cells into a cell. If you include a symbol in a cell, all animation links associated with that symbol remain intact.

After you use a cell in a SmartSymbol, if you break the SmartSymbol, you cannot resize the cell.

Create a cell

1. Select the objects that you want to include.
2. On the **Animation** menu, in the **Cell** group, select **Make Cell**.

Break a cell

1. Select the cell.
2. On the **Animation** menu, in the **Cell** group, select **Break**.

Group objects to symbols

Symbols cannot contain bitmaps, buttons, cells, wizards or trends. If one of the selected objects has animation links attached to it, the links are attached to the new symbol.

Create a symbol

1. Select the objects that you want to include.
2. On the **Animation** menu, in the **Graphic** group, select **Make Graphic**.

Break a symbol

1. Select the symbol.
2. On the **Animation** menu, in the **Graphic** group, select **Break**.

Common manipulations

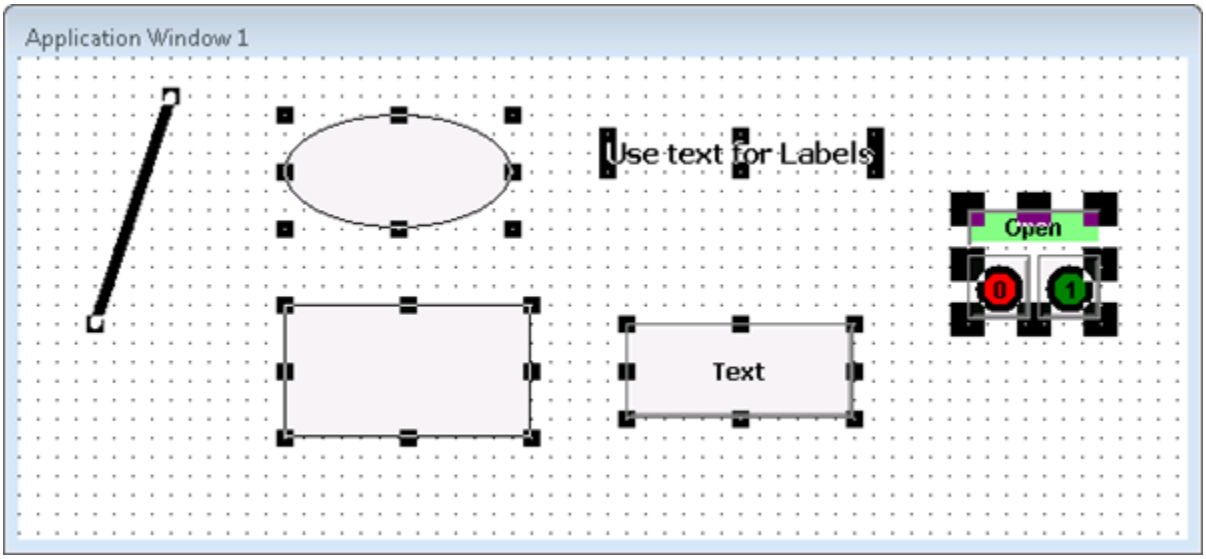
Right-click an object to see a menu showing the valid commands or actions you can apply to that object. You can:

- Select objects
- Move objects
- Align objects
- Arrange objects
- Layer objects
- Undo changes

- Flip objects
- Flip symbols
- Resize objects
- Rotate objects
- Change font
- Change line or outline
- Change fill
- Delete objects
- Control horizontal and vertical spacing

Select objects

You must select an object before you can modify it. When you select an object, handles appear on the perimeter of the object. You can use these handles to resize and/or reshape the object.



Select all objects in the active window

- On the **Animation** menu, in the **Object** group, choose **Select All.**
  Alternatively, press **F2**.

Select an object

1. On the **Animation** menu, in the **Mode** group, choose **Select**.
   The Select Mode is enabled.
2. Highlight the object you want to select.

Un-select an object

- Select a blank area of the window.

Select multiple objects

1. On the **Animation** menu, in the **Mode** group, choose **Select**.
   The Select Mode is enabled.
2. Select the first object, then SHIFT+click the other objects.

Select a group of objects

1. On the **Animation** menu, in the **Mode** group, choose **Select**.
   The Select Mode is enabled.
2. Drag a box around the objects. All the objects that are completely within the rectangle are selected.

Un-select a specific object or objects from a group of objects

- SHIFT+click the object.

Move objects

You can move objects by:

- Dragging them.
- Using the arrow keys on your keyboard.
- Typing window coordinates in boxes in the status bar.

When you are moving an object, notice how the coordinates in the status bar change.

Move an object by dragging

- Select the object and drag it.

When moving objects with the arrow keys, how far an object moves depends upon whether or not the grid is showing.

When the grid is showing, how many pixels an object moves depends upon the grid spacing, which is set on the WindowMaker configuration screen. The default setting is ten pixels between grid points.

When the grid is showing,

- Pressing an arrow key moves the object one grid point.
- Pressing SHIFT + an arrow key moves the object two grid points.
- Pressing CTRL + an arrow key moves the object four grid points.

When the grid is not showing,

- Pressing an arrow key moves the object one pixel.

- Pressing SHIFT + an arrow key moves the object ten pixels.
- Pressing CTRL + an arrow key moves the object 50 pixels.

Move an object with the Arrow keys

- Select an object and
  - press an arrow key.
  - SHIFT + press an arrow key.
  - CTRL + press an arrow key.

Align objects

You can align objects by their left or right edges, centers, center points, tops, middles, or bottoms.

Using the menu commands or the buttons, you can align in several ways.

Select	Or select	To do this
Align Left		Align left edges of the objects with the left edge of the object farthest to the left.
Align Center		Align objects with the vertical center line of the group.
Align Right		Align right edges of objects with the right edge of the object that is farthest to the right.
Align Tops		Align top edges with the top edge of the highest object.
Align Middle		Align horizontal centers with the middle of the group.
Align Bottom		Align bottom edges with the bottom edge of the lowest object.
Align Center points		Align center points with the center point of the group.

Align objects

1. Select multiple objects.
2. On the **Home** menu, in the **Alignment** group, select the appropriate align command.

Layer objects

You can position objects in front of or behind others.

Position an object behind another object

1. Select the objects(s).
2. On the **Home** menu, in the **Arrange** group, select **Send to Back**.

Position an object in front of another object

1. Select the objects(s).
2. On the **Home** menu, in the **Arrange** group, select **Bring to Front**.
   Alternatively, press SHIFT+F9.

Control horizontal and vertical spacing

You can space objects horizontally between the left most selected object and the right most selected object.

You can also control the vertical spacing between the top most selected object and the bottom most selected object.

You can also space the object to align at the center point.

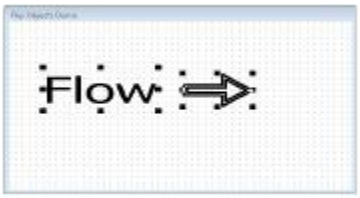
Space objects horizontally or vertically

1. Select the objects.
2. On the **Draw** menu, in the **Spacing** group, select **Horizontal, Vertical**, or **Center Points.**

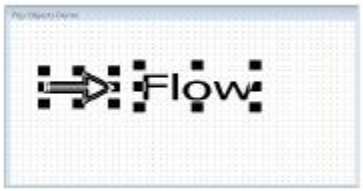
Flip objects and cells

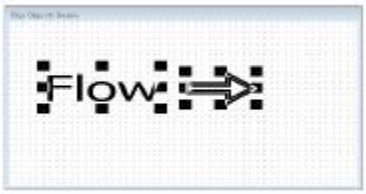
You can flip most objects horizontally or vertically. You can flip objects singly or in groups.

When you flip an object, you transform it to its mirror image. You cannot flip text.



When you flip cells, they are not mirrored. Only the position of the cell in the group of objects is mirrored.





Compare the location of the cells before and after they are flipped. The position is flipped, not the contents.

Flip an object or a cell

1. Select the object(s).

2. On the **Draw** menu, in the **Arrange** group, select the down arrow next to Rotate .
3. Select **Flip Horizontal** or select **Flip Vertical**.

Resize objects

You can resize an object using two methods. You can drag it or specify an exact width and height.

If snap to grid is turned on, the object snaps to the grid during the proportional resizing operation. The result is a slight deviation in the ratio between vertical to horizontal size. To avoid this deviation, turn off the snap to grid.

Resize an object by dragging

1. Select the object and then position the point of the arrow cursor in the center of a handle.
2. Drag the handle to resize the object.

Resize an object by proportional resizing

- Select the object and SHIFT+drag.

Resize an object by dimensions

1. Select the object.
2. In the Properties panel, type the width and height dimensions in the **W, H** boxes.

Rotate objects

You can rotate most objects including symbols, text, and bitmaps. You cannot rotate cells.

You can rotate objects clockwise or counter clockwise 360 degrees in 90 degree increments.

Rotating objects in WindowMaker has nothing to do with dynamically rotating objects at run time or in WindowViewer. You rotate objects in WindowViewer by linking them to an orientation animation.

Rotate an object

1. Select the object(s).

2. On the **Draw** menu, in the **Arrange** group, select the down arrow next to **Rotate**.
3. Select **Clockwise** or select **Counter Clockwise**.

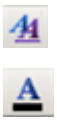
Change text appearance

You can set the appearance of fonts before you create text by setting defaults in advance or you can change the appearance of fonts after you create them. For information about setting font defaults, see Set font defaults.

The text justification attribute settings are important for text objects showing dynamic values. The justification determines how fields of varying length appear at run time.

For example, if you are showing a numeric value at the end of a text string that is centered or is right justified, the entire text string, including the value, is centered or justified each time there is a change in the number of digits shown.

Using the menu commands or the buttons, you can configure text in several ways.

To	Select	Button
Change the font, style, color, or text size	**Font**	
Make the text bold	**Bold**	
Make the text italic	**Italic**	
Make the text underlined	**Underline**	
Reduce or enlarge font size	**Reduce Font** or **Enlarge Font**	
Change the justification	**Left Justified**, **Centered**, or **Right Justified**	

To configure the text appearance

1. Select the text object.
2. On the **Draw** menu, in the **Format** group, select the appropriate text command.

Change lines and outlines

You can change the color and either the pattern or width of lines or outlines around outlined objects. Outlined objects include filled shapes such as ellipses, rectangles, and polygons, as well as bitmaps and other imported images.

Wider lines take longer to draw in run time. Dashed and dotted lines can only be one pixel wide.

Set the line appearance default settings

1. Select a blank area of the window.
2. On the **Draw** menu, in the **Format** group, select the line width or pattern.
3. Highlight the **Line Color** tool, and select a color.

Change the color of a line

1. Select a line, a group of lines, or an object with an outline.
2. On the **Draw** menu, in the **Format** group, select the **Line Color** tool.
3. Select a color.

Change the style or width of a line or outline

1. Select the object.
2. On the **Draw** menu, in the **Format** group, select the line style or width.

Remove an outline

1. Select the object.
2. On the **Draw** menu, in the **Format** group, select **No Line**.

Change fill

Filled shapes include shape surrounded by a line. Examples of filled shapes are rectangles, rounded rectangles, circles, ellipses, and polygons.

Change the fill color of an object

1. Select the object.
2. On the **Draw** menu, in the **Format** group, select the **Fill Color** tool.
3. Select a color.

Set the default color for filled shapes

1. Select a blank space on the window.
2. On the **Draw** menu, in the **Format** group, select the **Fill Color** tool.
3. Select a color.

Delete objects

You can delete one or more objects.

Delete an object

- Do either of the following:
  - Right-click the object and select **Erase**.
  - Select the object and then press Delete.

Undo changes

WindowMaker records your editing and formatting changes for each window. By default, WindowMaker supports 10 levels of undo/redo, where one level represents one action. You can set WindowMaker to retain up to 25 levels of actions. You can also turn off undo/redo by setting the undo/redo level to zero.

If you close the window, all recorded actions are cleared.

Undo a command

- On the quick access toolbar, select **Undo**.

Redo a command

- On the quick access toolbar, select **Redo**.

Set the number of undo/redo levels

1. On the **File** menu, select **Configure**, and then select **WindowMaker**.
   The WindowMaker configuration screen appears.
2. In the **Levels of Undo** box, type the number of levels.

Special manipulations for all objects

You can modify and manipulate simple objects. You can:

- Cut, copy, and paste objects.
- Cut, copy, and paste object links.
- Duplicate objects.

Cut, copy, and paste objects

Cut, copy, and paste operations in WindowMaker are much the same as in other Windows based applications but there are some significant differences you need to be aware of.

When you cut, copy, or paste an object, attributes and animation links are also cut, copied, or pasted with it.

All pasted objects remain selected after being pasted and you can move them to adjust their location.

Cut an object

- Right-click the object and then select **Cut**.

Copy an object

- Right-click the object and then select **Copy**.

Paste an object

1. Right-click a blank space in the window and select **Paste**. The cursor changes to a corner symbol.
2. Hold down the left mouse button. The cursor changes to a dotted rectangle the size of the copied object.
3. Drag the rectangle to locate the object.
4. Release the mouse button.

Cut, copy, and paste object links

The Clipboard is a temporary storage area for links you cut or copy.

- The Clipboard only stores the links for your most recent cut or copy action.
- You can paste the links to any object or symbol that supports the links in the Clipboard.
- If a pasted link is not supported by the object, for example, a line color link on a text object, the link is not pasted.
- If you select multiple objects for pasting, the links are pasted to all objects.

Cut, copy, paste and clear links

- Right-click the object, point to **Links** and select the appropriate command.

Duplicate objects

Duplicating objects is similar to copying objects and their animation links, but has the advantage of also duplicating the offset distance and direction when the objects are duplicated more than once.

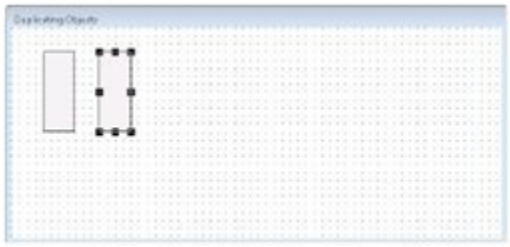
When you move a duplicated object without un-selecting it, then duplicate it again, the third iteration is offset the same distance and direction between the first two iterations.

You can repeat this procedure as many times as necessary.

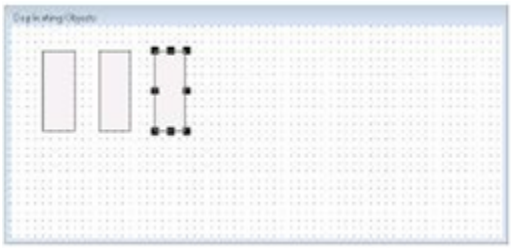
Duplicate an object

1. Right-click the object and select **Duplicate**. The object is copied and pasted to an offset position from the original object.

2. Keeping the duplicated object selected, drag it to a different position.



3. Again, without un-selecting the duplicated object, right-click the object and select **Duplicate** again. The third iteration of the object appears in the same relative position as the first two.



Special manipulations for special objects

The following object types have unique attributes that you can edit:

- Polylines and polygons

- Bitmap containers
- Bitmap transparencies
- Rounded rectangles
- Object text

Reshape polyline and polygon objects

You can adjust the shapes of polylines and polygons.

Reshape a polyline or polygon

1. Select the object.
2. Do one of the following.
   - On the **Animation** menu, in the **Object** group, select **Reshape** to view all reshape options.
   - Right-click the object and click **Reshape Object**.

     Each shape definition point becomes a handle.
3. Drag a handle to reshape.

Add a point to a polygon

1. Select the object.
2. Do one of the following.
   - On the **Animation** menu, in the **Object** group, select the arrow to display the hidden commands, and then select **Add Point**.
   - Right-click the object and select **Add Point**.
3. Select an edge of the polygon, then drag the point to change the shape of the polygon.

Delete a point from a polygon

1. Select the object.
2. Do one of the following.
   - On the **Animation** menu, in the **Object** group, select the arrow to display the hidden commands, and then select **Delete Point**.
   - Right-click the object and **Delete Point**.
3. Select a point of the polygon, the point is deleted and the shape of the polygon changes.

Work with bitmap containers

Bitmap containers are objects that enable you to import graphic objects such as pictures, screen captures, and drawings into your application.

Supported bitmap container file types are .bmp, .jpeg, .jpg, .pcx and .tga.

When you import a bitmap, it automatically fills the bitmap container, but you can resize it to its original size and proportions.

You can rotate bitmaps in 90 degree increments.

You can include bitmaps in a cell but not in a symbol.

Using WindowMaker, you are able to place more bitmaps into a window than can load in WindowViewer. If you need to place a large number of bitmaps in a window, be sure to test the window in WindowViewer before releasing the application.

Import a bitmap image

1. On the **Draw** menu, in the **Shapes** group, select **Bitmap**.
   The cursor turns into a cross-hair.
2. Drag the cursor to draw a bitmap container.
3. On the **Animation** menu, in the **Bitmap** group, select **Import**.
   The **Select Image File** dialog box appears.
4. Select the image filename and select **OK**.

Make the bitmap its original size

1. Select the image.
2. On the **Animation** menu, in the **Bitmap** group, select **Original Size**.

Paste a bitmap image

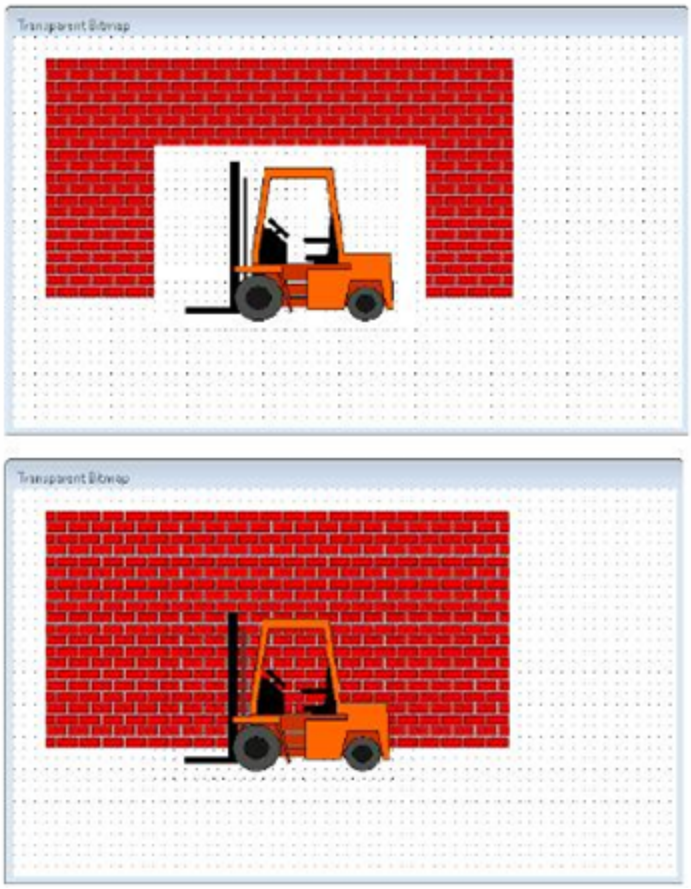
1. Copy the graphic to the Windows Clipboard.
2. Select the **Bitmap** tool and draw a bitmap container in your window.
3. On the **Animation** menu, in the **Bitmap** group, select **Paste**
   Alternatively, right-click the bitmap container and select **Paste Bitmap**.

Edit a bitmap

1. Select the bitmap.
2. On the **Animation** tab, in the **Bitmap** group, select **Edit**.
   Microsoft Paint opens showing the bitmap.
3. Edit the bitmap in MS Paint.
4. Save and close MS Paint.

Define bitmap transparency

When you define a transparent color in a bitmap, the window background or any objects behind the bitmap shows through it everywhere the transparent color is used. You can define only one transparent color per image.



Create a transparent bitmap

1. With the bitmap selected, select the **Transparent Color** button On the **Draw** menu, in the **Format** group to open the transparent color palette.
2. Right-click a color square in the **Custom Palette**. The **Edit Custom Color** dialog box appears.
3. Select the eye dropper tool.
4. Select the color in the bitmap that you want to make transparent. The color is copied to the color square that you selected in the color palette.
5. Select the color square to apply the transparent color to the bitmap. All the pixels in the image that are that color become transparent.

Change the radius of a rounded rectangle

You can increase and/or decrease the corner radius of a Rounded Rectangle.

Increase or decrease a rounded object's radius

1. Select the object.
2. On the **Animation** menu, in the **Object** group, display the hidden commands, and then select **Enlarge Radius** or **Reduce Radius**.

Substitute object text

You can edit the text of objects that have text, such as symbols, cells, or buttons.

When you change a text string, it retains all of its original attributes, font, style, color, and so on. Text formatting also applies to numeric values.

Change the text in an object

1. Select the object or button with the text. Do either of the following:
   - On the **Animation** menu, in the **Substitute** group, select **Strings**.
   - Right-click the text object, point to **Substitute** and then select **Substitute Strings**.
2. In the **New String** box, type the new string and select **OK**.

Change a portion of text in a series of text objects

1. Select all the text objects.
2. On the **Animation** menu, in the **Substitute** group, select **Strings**.
3. Select **Replace**.

   The **Replace Text** box appears.
4. In the **Old Text** box, type the portion of text to replace.
5. In the **New** box, type the new text string.
6. Select **OK**.

   The new text string replaces the old text string in all the selected objects.

Print window information about WindowMaker objects

You can print information about the InTouch graphical objects that you have placed in an application window. InTouch graphical objects include simple objects, such as lines, buttons, and text, as well as complex objects, such as cells, ActiveX controls, SmartSymbols, and Industrial graphics.

You can print details for graphical objects to either a printer or an .html file. An .html file is created for each window. The .html file contains:

- A picture of the window as a referenced .png file.
- A listing of details about each graphical object in the window.

If you print graphical object details for multiple windows, a summary .html file is created that has links to the window-specific .html files.

Print information about WindowMaker objects

1. Open an InTouch application in WindowMaker.
2. On the quick access toolbar, select **Print**.

   The **WindowMaker Printout** dialog box appears.
3. Select the **Windows** checkbox.
4. Specify the windows to print:

- **All** prints the images/graphics for all windows in the application.

  Note: Printing all windows of an application to an XPS file can cause WindowMaker to stop responding if the windows contain many graphics or large graphics that exceed the available memory of the computer hosting WindowMaker. You should print these windows one at a time to avoid problems caused by memory shortages.

- **Selected** prints only the images/graphics for specific windows. The **Windows to Print** dialog box appears. Select the windows in your application you want to print and select **OK**.
- **Batch** prints only the information for windows specified in a .csv file.

  For details on the .csv format, see .CSV format for printing windows.

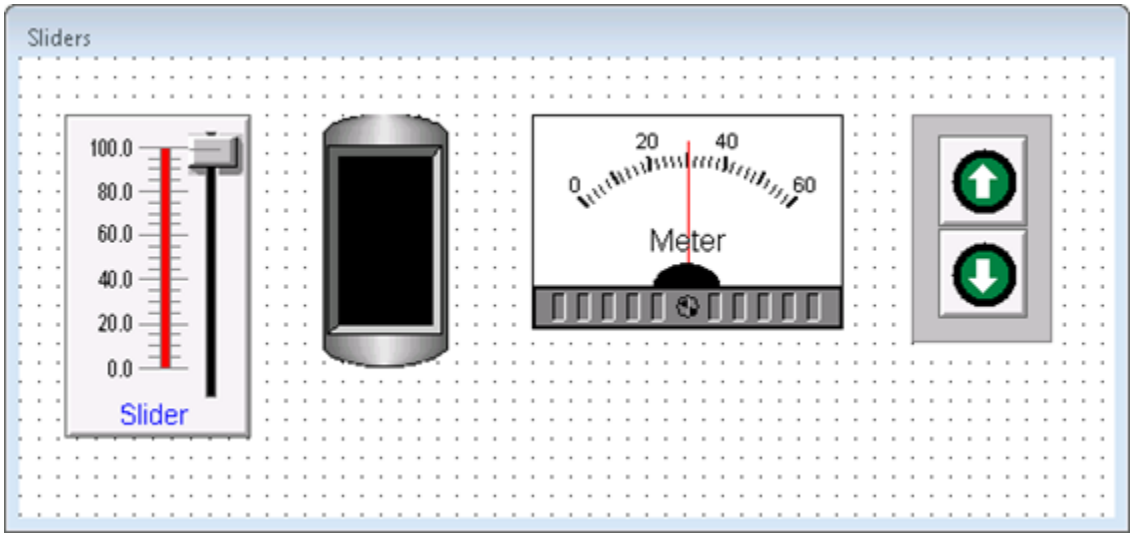
5. Select the **Image/Graphics** checkbox.
6. Select **Next**. The **Select Output Destination** dialog box appears.
7. Do one of the following
   - Select **Send output to Printer** to print the information.
   - Select **Send output to HTML File** to create a summary .html file, a single .html file, and .png file for each window you specified. If .html and .png files exist, they are automatically overwritten.
8. Select **Print**.

Animate objects

You can animate objects or symbols by means of animation links. Animation links connect the values of tags or expressions to your objects or symbols.

For example, you can:

- Create a slider or tank symbol that shows the liquid level in the tank.
- Create a meter that shows a range of values.
- Create touch screen symbols for operator control.



Two types of animation links

There are two basic types of animation links: display links and touch links.

- Display links show information to operators. Examples of display links are changing colors, changing fill levels, horizontal or vertical movements, and blinking objects.
- Touch links enable operator input into the system. Examples of touch links are sliders or push buttons that respond to operator input.

You can define multiple links for objects or symbols. By combining various links, you can create almost any screen animation effect.

Data display animations

Data display animations only show information to operators. These animations do not support operator input.

Create value displays

Use a value display text object to show the value of a tag. This lets you show things like fill levels, on/off status, or alarm messages.

You can use any one of three types of *value display links* to show messages at run time.

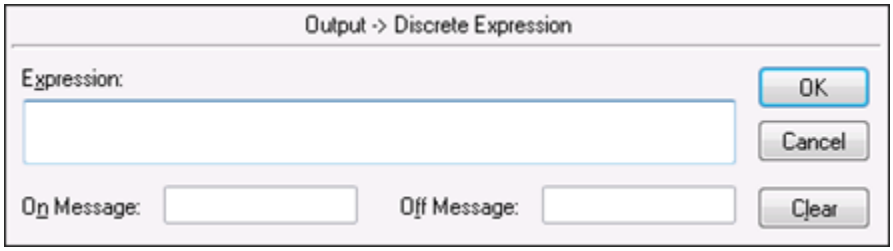
Value Display Type	Shows
Discrete	Discrete values such as on or off
Analog	The value of an analog expression such as fill level or speed.
String	The value of a string expression such as "Fill Level = 100".

You can use up to 1023 characters in an expression. If you need a larger expression, create a QuickFunction and call it in your expression.

Messages appear in the location of the original text object using the font, size, color, alignment, and linked attributes set for that object. The original contents of the field have no effect on the message at run time.

Create a discrete value display link

1. Right-click the text object and select **Animation Links**. The **Animation Links** dialog box appears.
2. In the **Value Display** area, select **Discrete**. The **Output** -> **Discrete Expression** dialog box appears.



3. In the **Expression** box, type the name of a discrete tag or an expression that equates to a discrete value. For example:

```
Cooling_Pump
```

4. In the **On Message** box, type the message to appear when the value of the expression equals 1, true, on, or yes. For example:

```
Pump is ON
```

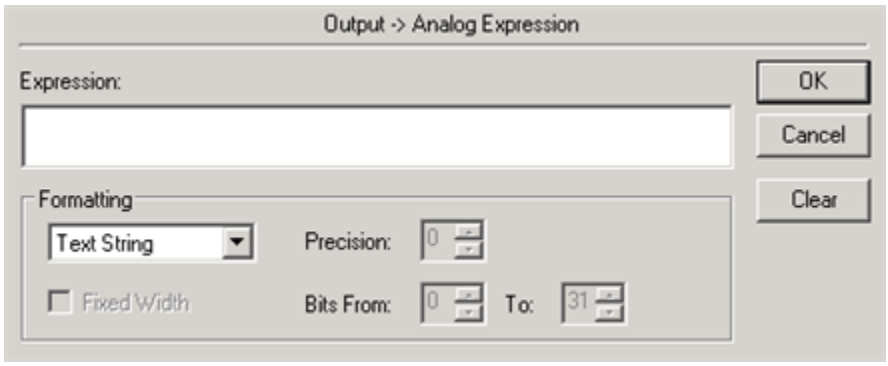
5. In the **Off Message** box, type the message to appear when the value of the expression equals 0, false, off, or no. For example:

```
Pump is OFF
```

6. Select **OK**.

Create an analog value display link

1. Right-click the text object and select **Animation Links**. The **Animation Links** dialog box appears.
2. In the **Value Display** area, select **Analog**. The **Output -> Analog Expression** dialog box appears.



3. In the **Expression** box, type an analog (integer or real) tagname or an expression that equates to an analog value. For example:

```
Tank_CV*0.06
```

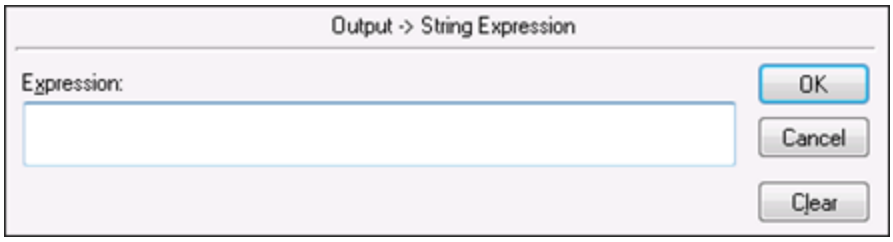
4. In the **Formatting** area, in the list, select each data type for which you want to configure run time advanced formatting. The **Fixed Width** checkbox, **Precision** box, and **Bits From** and **To** boxes become available based on the data type you select. For information on configuring these options, see Advanced formatting for text.

Note: During run time, the analog value input link field can be resized by selecting and dragging with the pointer and mouse.

5. Select **OK**.

Create a string value display link

1. Right-click the text object and select **Animation Links**. The Animation Links dialog box appears.
2. In the **Value Display** area, select **String**. The **Output -> String Expression** dialog box appears.



3. In the **Expression** box, type the name of a message tag or an expression that uses a message tag. For example:

```
"The Tank Level is:" + Text(TankLevel,"#")
```

4. Select **OK**.

Note: During run time, the string value input link field can be resized by selecting and dragging with the pointer and mouse.

Create movement

You make objects move at run time by using location links. You can make an object move horizontally, vertically, or both, as the value of an analog tag or expression changes. For example, as a tank level increases and decreases, an indicator moves up and down.

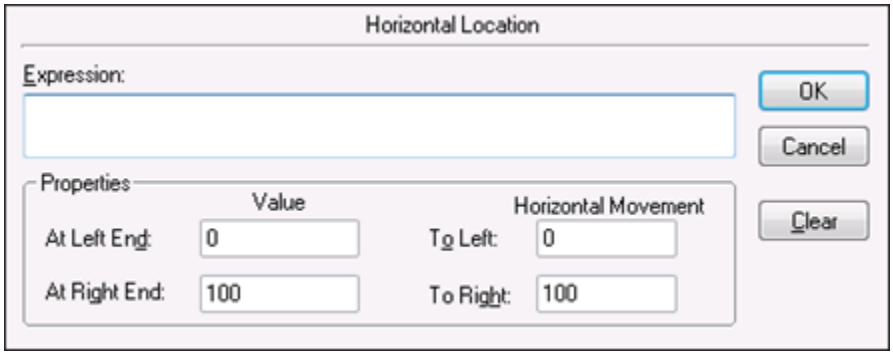
Create horizontal movement

1. Place the object on the screen in the starting location.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.
3. In the **Location** section, select **Horizontal**.

   The **Horizontal Location** dialog box appears.



4. In the **Expression** box, type an analog tag name or an expression that equates to an analog value.
5. In the **Properties** section, configure how far the object moves. Do the following:
   a. In the **At Left End** box, type the value of the analog tag when the object should be at its farthest left position.
   b. In the **At Right End** box, type the value of the analog tag when the object should be at its farthest right position.
   c. In the **To Left** box, type the number of pixels the object should move to the left of its starting position.

d. In the **To Right** box, type the number of pixels the object should move to the right of its starting position.

6. Select **OK**.

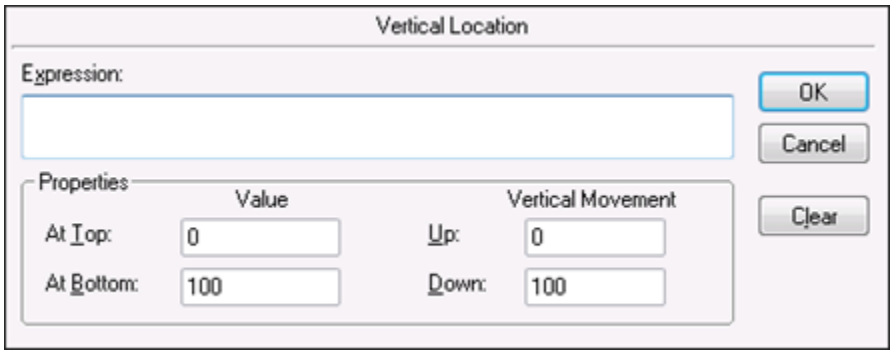
Create vertical movement

1. Place the object on the screen in the starting location.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.
3. In the **Location** section, select **Vertical**.

   The **Vertical Location** dialog box appears.



4. In the **Expression** box, type an analog tag name or an expression that equates to an analog value.
5. In the **Properties** section, do the following:
   a. In the **At Top** box, type the value of the analog tag when the object should be at the top position.
   b. In the **At Bottom** box, type the value of the analog tag when the object should be at the bottom position.
   c. In the **Up** box, type the number of pixels the object should move up from the starting position.
   d. In the **Down** box, type the number of pixels the object should move down from the starting position.
6. Select **OK**.

Create rotation

You can make an object move around a center point as the value of an analog tag changes by using orientation links. For example, as pressure increases or decreases, a pointer can move around a dial.

The orientation link uses the center of the object or symbol as the default center of rotation. You can offset the center of rotation.

Orientation links are not supported for Industrial graphics.

Tip: Draw a temporary rectangle from the center of the object to the rotation center point. Now you can read the X and Y offset dimensions in pixels from the W, H boxes in the status bar.

Create an orientation link

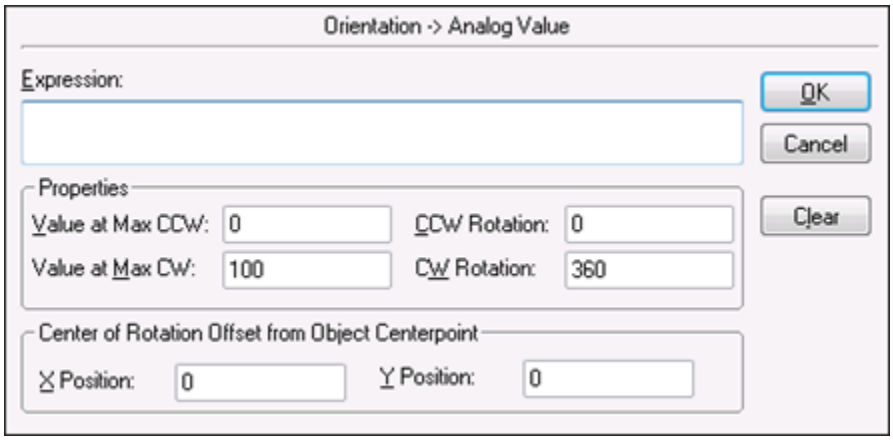
1. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.

2. In the **Miscellaneous** section, select **Orientation**.

   The **Orientation -> Analog Value** dialog box appears.



3. In the **Expression** box, type the name of an analog tag or an expression that equates to an analog value.
4. In the **Properties** section, do the following:
   a. In the **Value at Max CCW** box, type the value the expression must be for the object to rotate to the maximum counter-clockwise position.
   b. In the **Value at Max CW** box, type the value the expression must be for the object to rotate to its maximum clockwise position.
   c. In the **CCW Rotation** box, type the degrees the object rotates counter-clockwise when the **Value at Max CCW** is reached.
   d. In the **CW Rotation** box, type the degrees the object rotates clockwise when the **Value at Max CW** is reached.
5. In the **Center of Rotation Offset from Object Centerpoint** section, do the following:
   a. In the **X Position** box, type the horizontal offset of the rotation centerpoint. Enter the offset in pixels from the centerpoint of the object.
   b. In the **Y Position** box, type the vertical offset of the rotation centerpoint. Enter the offset in pixels from the centerpoint of the object.
6. Select **OK**.

Animate sizes

You can vary the height and/or width of an object according to the value of an analog tag or expression by using object size links.

For example, a pressure indicator can become larger as pressure increases, or an object on a conveyor can appear to move toward the viewer by becoming larger.

Object size links not only control the size of an object, but the direction in which the object changes size through the use of an anchor for the animation.

Create an object size height link

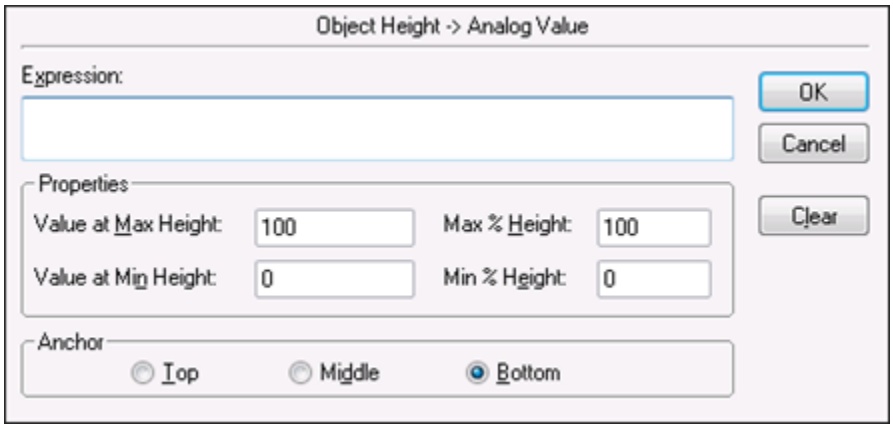
1. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.

2. In the **Object Size** section, select **Height**.

   The **Object Height -> Analog Value** dialog box appears.



3. In the **Expression** box, type the name of an analog tag or an expression that equates to an analog value.
4. In the **Properties** section, do the following:
   a. In the **Value at Max Height** box, type the value of the tag or expression when the object reaches maximum height.
   b. In the **Value at Min Height** box, type the value of the tag or expression when the object reaches minimum height.
   c. In the **Max % Height** box, type the percentage of its original height that the object will be when the tagname or expression reaches the value set in the **Value at Max Height** box. The percent figures are expressed as a percentage of the drawn size of the object. The drawn size is always the 100% size.
   d. In the **Min % Height** box, type the percentage of its original height that the object will be when the tagname or expression reaches the value set in the **Value at Min Height** box. The percent figures are expressed as a percentage of the drawn size of the object. The drawn size is always the 100% size.
5. Select the **Anchor** point from which the object enlarges.
   - Select **Top** for the object to enlarge from its top downward.
   - Select **Middle** for the object to enlarge from its centerpoint outwards in both directions.
   - Select **Bottom** for the object to enlarge from its bottom upwards.
6. Select **OK**.

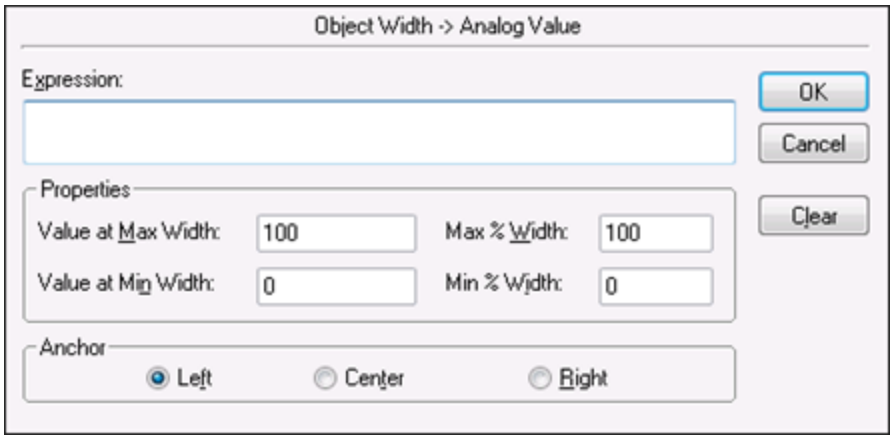
Create a object size width link

1. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.
2. In the **Object Size** section, select **Width**.

   The **Object Width -> Analog Value** dialog box appears.



3. In the **Expression** box, type the name of an analog tag or an expression that equates to an analog value.
4. In the **Properties** section, do the following:
   a. In the **Value at Max Width** box, type the value of the tag or expression when the object reaches maximum width.
   b. In the **Value at Min Width** box, type the value of the tag or expression when the object reaches minimum width.
   c. In the **Max % Width** box, type the percentage of its original width that the object will be when the tagname or expression reaches the value set in the **Value at Max Width** box. The percent figures are expressed as a percentage of the drawn size of the object. The drawn size is always the 100% size.
   d. In the **Min % Width** box, type the percentage of its original width that the object will be when the tagname or expression reaches the value set in the **Value at Min Width** box. The percent figures are expressed as a percentage of the drawn size of the object. The drawn size is always the 100% size.
5. Select the **Anchor** point from which the object enlarges in width.
   - Select **Left** for the object to enlarge from its left side.
   - Select **Center** for the object to enlarge from its centerpoint outwards in both directions.
   - Select **Right** for the object to enlarge from its right side.
6. Select **OK**.

Animate colors

You can animate color changes to any object by using color links. Changes can be based on the value of an analog or discrete tag, the value of an analog or discrete expression, or a discrete or analog alarm status.

You can use three kinds of color links to animate objects:

- Line Color
- Fill Color
- Text Color

For each of these three kinds of color links, four types of expressions can control color changes.

Expression Type	Changes the color based on
Discrete	The value of a discrete tag or expression.
Analog	The value of an analog tag or expression. You can define ten colors to represent differing values.
Discrete Alarm	The alarm state of a tag, Alarm Group, or Group Variable.
Analog Alarm	The alarm state of an analog tag, Alarm Group, or Group Variable. You can define five colors to represent five alarm conditions.

WARNING! Objects do not go into an alarm state when using an analog alarm link if the link is to a remote tag from an unconverted application created before InTouch version 7.11.

All discrete color links are created in the same way. The following procedure describes creating a fill color link.

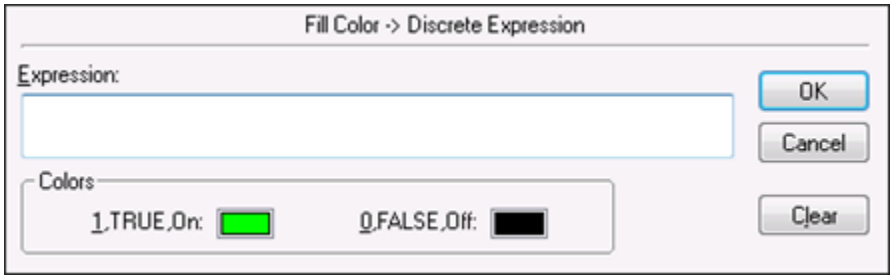
Create a discrete fill color link

1. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.
2. In the **Fill Color** section, select **Discrete**.

   The **Fill Color -> Discrete Expression** dialog box appears.



3. In the **Expression** box, type the name of a discrete tag or a discrete expression that equates to true or false.

   Discrete expressions can contain analog tags. For example, TankLevel >= 75. In this example, when the value of the variable "TankLevel" is greater than or equal to 75, the fill color of the object changes.
4. In the **Colors** section, select each color box to open the color palette. Select the color to use for each state.
5. Select **OK**.

Create an analog expression color link

1. Right-click the object and then select **Animation Links**.

   The **Animation Links** dialog box appears.
2. In the **Fill Color** section, select **Analog**.

   The **Fill Color -> Analog Expression** dialog box appears.

3. In the **Expression** box, type the name of an analog tag or an expression that equates to an analog value.
4. In the **Break Points** section, do the following.
   - Specify the breakpoint values where the object changes color.

   Tip: You do not have to use ten different colors. For example, if you only want the object to change color three times, type three values and use the same color for the remaining values. If you need a more versatile range, review the analog fill capabilities of Industrial Graphics.

   - In the **Colors** section, select a color for each breakpoint.
5. Select **OK**.

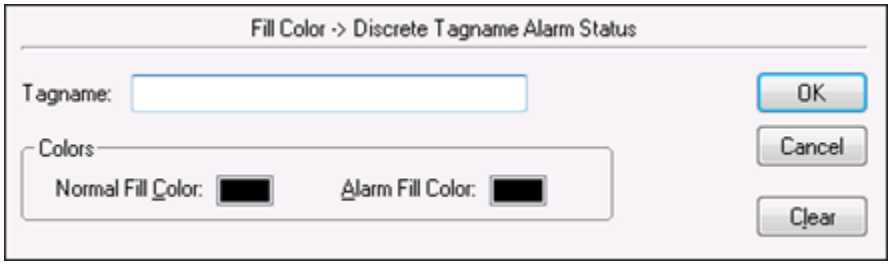
Create a discrete alarm status color link

1. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.
2. In the **Fill Color** section, select **Discrete Alarm**.

   The **Fill Color -> Discrete Tagname Alarm Status** dialog box appears.



3. In the **Tagname** box, type the name of the discrete tag to associate with the object.
4. In the **Colors** section, select a color for each alarm state.
5. Select **OK**.

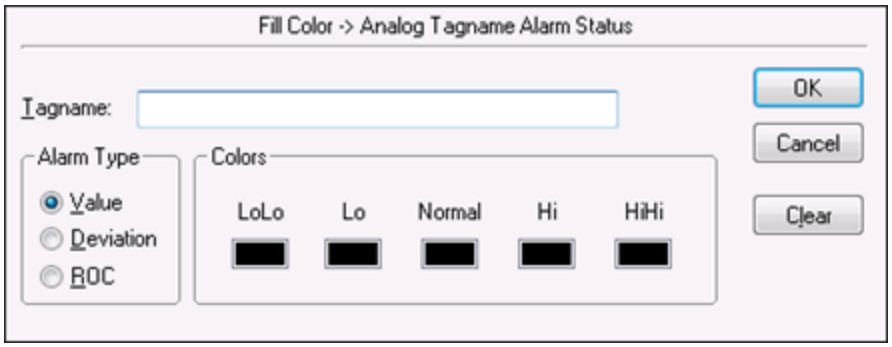
Create an analog alarm status color link

1. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.
2. In the **Fill Color** section, select **Analog Alarm**.

   The **Fill Color -> Analog Tagname Alarm Status** dialog box appears.



3. In the **Tagname** box, type the name of an analog tag to associated with the object.
4. In the **Alarm Type** section, select from one of the three types of alarms to associate with the object.

Alarm Type	Use up to
Value	Five colors to show the status of the value alarms.
Deviation	Three colors to show the status of the deviation alarms.
ROC (Rate of Change)	Two colors to show the status of a rate-of change alarm.

5. In the **Colors** section, select a color for each alarm state.
6. Select **OK**.

Animate fill levels

You can vary the fill level of an object using a percent fill link. The percent fill is based on the value of an analog tag or expression. You can create horizontal fills, vertical fills, or both.

For example, you can use a vertical fill link to show the level of liquid in a tank or a horizontal fill link to show the progress of a process.

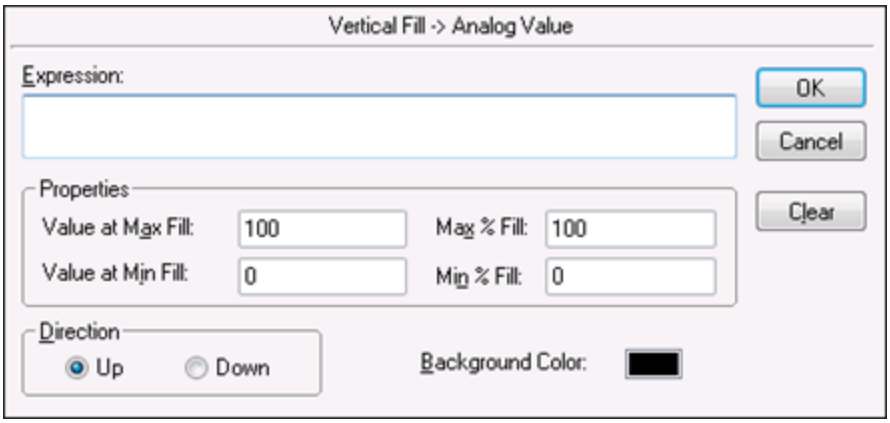
You create the horizontal and vertical percent fill links the same way.

Create a percent fill link

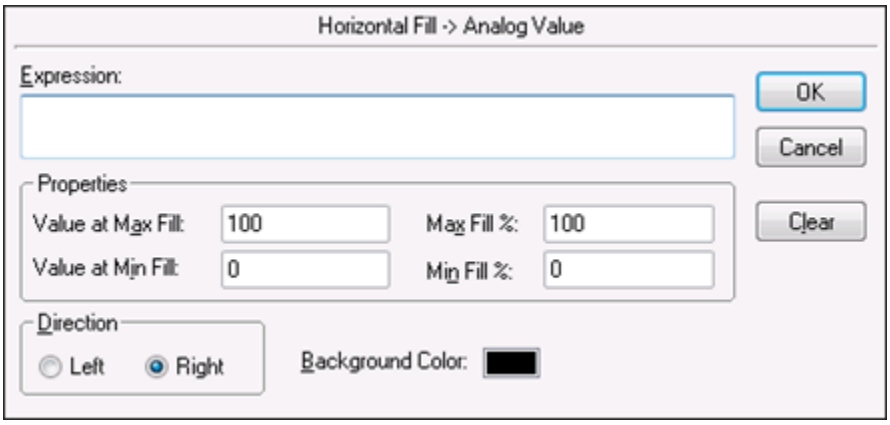
1. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and select **Animation Links**.

   The **Animation Links** dialog box appears.
2. In the **Percent Fill** section, do one of the following:
   - Select **Vertical**, the **Vertical Fill -> Analog Value** dialog box appears.



- Select **Horizontal**, the **Horizontal Fill -> Analog Value** dialog box appears.



3. In the **Expression** box, type the name of an analog tag or an expression that equates to an analog value.
4. In the **Properties** section, do the following:
   a. In the **Value at Max Fill** box, type the value of the expression that will result in the object being filled to its maximum level.
   b. In the **Value at Min Fill** box, type the value of the expression that will result in the object being filled to its minimum level.
   c. In the **Max % Fill** box, type the percentage (0-100) that the object will be filled when the expression reaches the level set in the **Value at Max Fill** box.
   d. In the **Min % Fill** box, type the percentage (0-100) that the object will be filled when the expression reaches the level set in the **Value at Min Fill** box.
5. In the **Direction** section, select the direction to fill from.
6. In the **Background Color** box, select the color of the unfilled portion of the object.
   - The actual fill color is the color that you select for the object when you draw it.
   - If you link both vertical percent fill and horizontal percent fill links to the same object, the last color you select is the background color.
7. Select **OK**.

Make objects blink

You can create animated objects that blink based on the values of tags by using blink links. For example, you can

create an object that blinks red when a certain piece of equipment is on, or when an alarm set point is reached.

The blink animation synchronization enables all active blink animations on all open windows to blink in unison based on their common blink speed (slow, medium, or fast).

Tip: Discrete expressions can contain analog tagnames. For example, TankLevel > 75. In this example, when the value of the TankLevel tag is greater than 75, the object begins blinking.

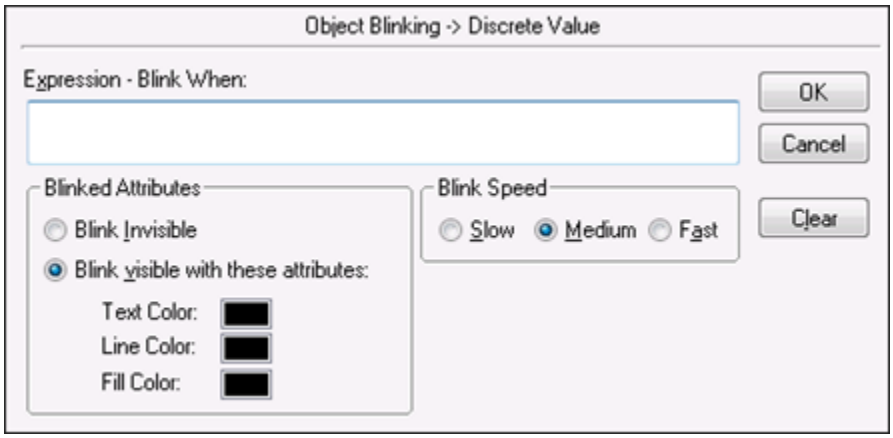
Create a blink link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Miscellaneous** section, select **Blink**.

   The **Object Blinking -> Discrete Value** dialog box appears.



4. In the **Expression - Blink When** box, type the name of a discrete tag or an expression that equates to a discrete value.
5. In the **Blinked Attributes** section, do the following:
   - Select **Blink Invisible** to set the object to blink by disappearing and reappearing in the window.
   - Select **Blink visible with these attributes** to set the object to remain visible, but change color when activated.
   - Select the **TextColor**, **Line Color** or **Fill Color** boxes to select colors for those parts of the object. The color palette appears.

     Note: If you select a fill blink color that is the same as the object's fill color, the object does not appear to blink.
6. In the **Blink Speed** section, set the blinking speed of the object. Select either **Slow**, **Medium**, or **Fast**.
7. Select **OK**.

Set the blink frequency for WindowMaker

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.

3. On the **Preferences** tab, in the **Blink frequency in msec** section, type the number of milliseconds to use for the three speeds.

   Note: Changes you make to these settings are global and affect all blink speeds in your application.

4. Select **OK**.

Enable visibility

You can create links to hide objects based on the values of various tags by using visibility links. Using visibility links, you can:

- Create the impression that moving objects only move in one direction, by hiding them when they move in the wrong direction.
- Create the impression that a moving object has stopped.
- Cause an object such as an alarm or error message to become visible only when it is activated.

Tip: Discrete expressions can contain analog tags, for example,
TankLevel >= 75. In this example, when the value of the tag, TankLevel is greater than or equal to 75, the object becomes visible in the window.

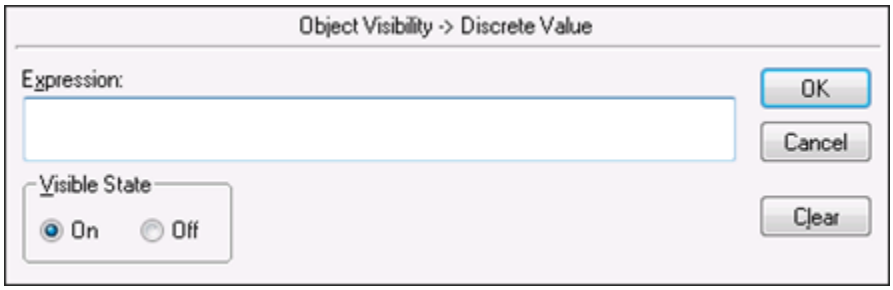
Create a visibility link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Miscellaneous** section, select **Visibility**.

   The **Object Visibility -> Discrete Value** dialog box appears.



4. In the **Expression** box, type the name of a discrete tag or an expression that equates to a discrete value.
5. Select the **Visible State** for the object. If you select **Off**, the object is invisible when the value of the expression is true. If you select **On**, the object is visible when the value of the expression is true.
6. Select **OK**.

Disable objects

You can impose a level of security on your application with disable links. For example, you can disable touch sensitive objects based on operator access level or name. Or you can secure a button from tampering if no one is logged on.

A disabled state of ON means the touch functionality of the object or button is turned off and is not active as long as the expression is true.

Tip: Discrete expressions can contain analog tagnames. For example, TankLevel >= 75. In this example, when the value of the variable "TankLevel" is greater than or equal to 75, the object is disabled.

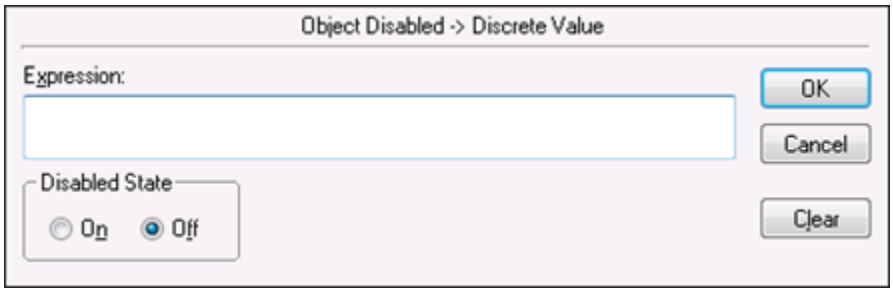
Create a disable link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Miscellaneous** section, select **Disable**.

   The **Object Disabled -> Discrete Value** dialog box appears.



4. In the **Expression** box, type the name of a discrete tag or expression that equates to a discrete value.
5. In the **Disabled State** section, do one of the following:
   - Select **On** to set the disabled state so that the object does not activate while the discrete tag or expression is true.
   - Select **Off** to remove the disabled state and enable the object to function while the discrete tag or expression is true.
6. Select **OK**.

Configure ToolTips

You can create ToolTips to give users information about an object on the screen by using tooltip links. ToolTips appear when the pointer moves over the object, and disappears when the pointer is moved away. The duration of time the tooltip appears and the positioning of the tooltip are determined by the operating system.

You can set either an expression or static text for your tooltips.

- Create a static tooltip to show the same message every time the tooltip appears.
- Create an expression tooltip so that every time the tooltip appears, the expression is evaluated and shown as the tooltip text.

For the following example expression, the text appears as the current value of the msgTooltipTag01 message tag.

```
msgTooltipTag01
```

For this example, the literal string appears followed by the current value of the iTemp tag and the result appears as the tooltip text:

```
"Current temp. is " + StringFromIntg (iTemp,10)
```

The width of the Tooltip window can be changed in Windows 7 and later versions of Windows. You can edit the InTouch.INI file to add an entry for Tooltip width.

Examples:

- Tooltipwidth=200
- A value of -1 (Tooltipwidth=-1) shows tooltip text on a single line without a line break
- If no tooltip width is specified, the default width of the tooltip window is 88.

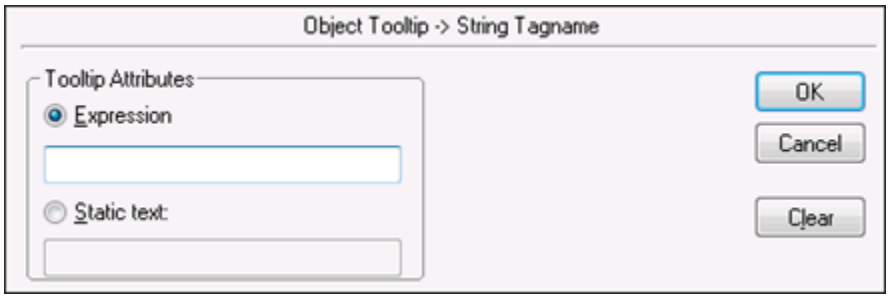
Create a tooltip link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Miscellaneous** section, select **Tooltip**.

   The **Object Tooltip -> String Tagname** dialog box appears.



4. In the **Tooltip Attributes** section, select either **Expression** or **Static** text.
   - If you select **Expression**, enter an expression that evaluates to a message value. This can be a simple message tagname or a more complex expression.
   - If you select **Statictext**, enter a static message, up to 131 characters, as the tooltip text.
5. Select **OK**.

Position a touch-sensitive window

You can make a window appear during run time in a precise location relative to a touch sensitive object. For example, an operator can select an object to view the status, name, or other data linked to that object. When the operator selects the object, either by a click or mouse-over, the window appears in the location you specify.

Use the script functions ShowAt() or ShowTopLeftAt() with the system read only tags $ObjHor and $ObjVer to locate the window relative to the object. You can also use fixed positions in these functions.

If the Windows **Display Properties** is set to the **Windows XP** theme, this functionality is erratic under certain circumstances.

The syntax looks like this:

```
ShowTopLeftAt (windowname, $ObjHor, $ObjVer);
```

where

windowname: the name of the window to be opened.

$ObjHor: the horizontal position of the center of the object selected.

$ObjVer: the vertical position of the center of the object selected.

The new window appears with its top left corner at the center of the selected object.

A similar script function opens the window with its center at the center of the selected object. The syntax looks like this:

```
ShowAt (windowname, $ObjHor, $ObjVer);
```

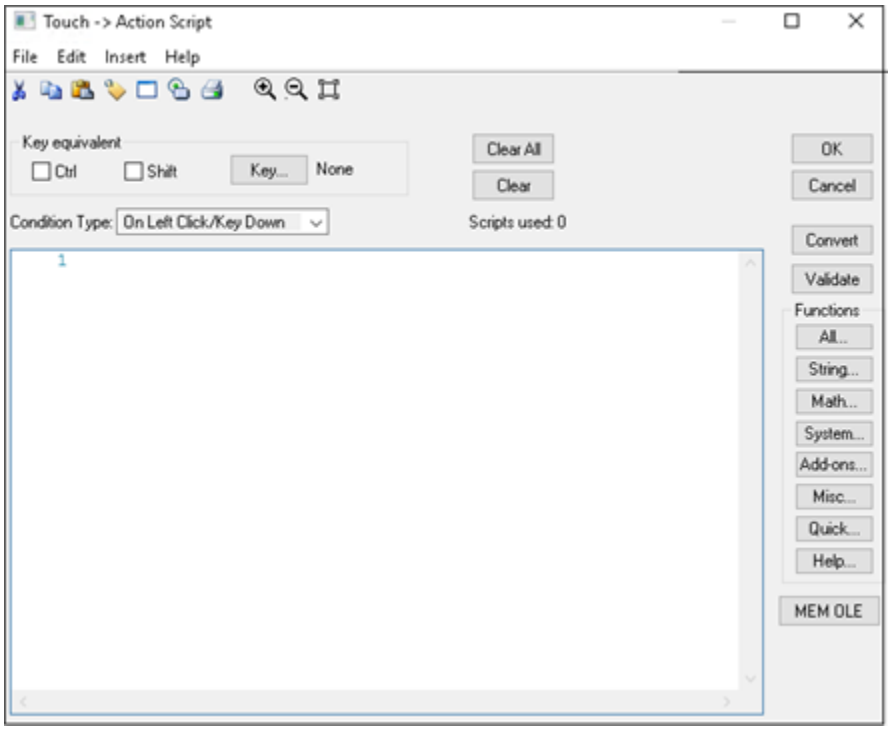
Open a window at the selected object

1. Design, name, and create the window to appear.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Touch Pushbuttons** section, select **Action**.

   The **Touch -> Action Script** dialog box appears.



4. Type one of following scripts according to the syntax defined previously:

   ```
   ShowTopLeftAt (windowname, $ObjHor, $ObjVer);
   ```

   or

   ```
   ShowAt (windowname, $ObjHor, $ObjVer);
   ```
5. In the **Condition Type** box, click the mouse action to open the window.
6. Select **OK**.

$ObjHor system tag

Contains the horizontal pixel location of the center of the object in focus.

Category

system

Usage

```
$ObjHor
```

Data Type

Integer (read only)

See Also

$ObjVer

$ObjVer system tag

Contains the vertical pixel location of the center the object in focus.

Category

system

Usage

```
$ObjVer
```

Data Type

Integer (read only)

See Also

$ObjHor

Data entry animations

To create objects that enable operator interaction, use touch links. Touch links enable operators to input data

into a running application. For example, an operator can log on with a keyboard, turn a valve on or off, enter a new alarm setpoint, or start or stop a process.

A frame appears around a touch-sensitive object when it has the focus. An object gets the focus when the user moves the cursor over it or presses the TAB or arrow keys to move the focus to the object.

If you expect your users to use the TAB key to select touch sensitive objects, try to arrange them in horizontal patterns. Pressing the Tab key moves the focus from one object to another from left to right beginning at the top of the window then moving down.

If a touch link object contains text objects that are placed on top of each other, the top text object is the only one that appears.

An operator can activate a touch-sensitive object by selecting it, pressing an assigned key equivalent, pressing **Enter** when the object frame appears, or by actually touching it if using a touch screen display device.

You can create nine types of user input touch links:

Touch Link	Action
User Inputs	• Discrete • Analog • String
Sliders	• Vertical • Horizontal
Pushbuttons	• Discrete Value • Action • Show Window • Hide Window

When a text field is used for input, text appears on the screen as the keys are pressed.

If you don't want text to appear as you type it, select the **Input Only** option in the configuration panel for the link.

Enable discrete input

You can design operator input touch links to change the value of discrete tags from one state to another. For example to turn a pump on or off, use discrete links.

Create a discrete input link

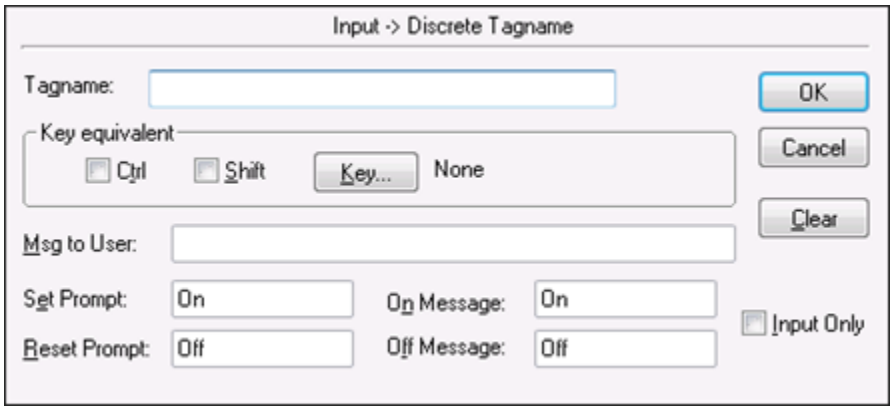
1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.

3. In the **Touch Links** area, under **User Inputs**, select **Discrete**.

   The **Input -> Discrete Tagname** dialog box appears.



4. In the **Tagname** box, type the name of a discrete tag.
5. Optionally, in the **Key Equivalent** area, assign a key equivalent to the link. For more information, see Create keyboard shortcuts.
6. Configure the discrete input details. Do the following:
   - In the **Msg to User** box, type the message to appear in the **Input** dialog box.
   - In the **Set Prompt** and **Reset Prompt** boxes, type the messages to appear on the buttons the operator will select to turn the discrete value on and off.
   - In the **On Message** and **Off Message** boxes, type the messages to appear in the text field associated with the object when the value is on or off.
7. Select the **Input Only** checkbox to prevent the input from appearing in a text field associated with the object.
8. Select **OK**.

Enable analog input

You can create an object for operators to input real values such as alarm set point or conveyor speed by using an analog input link. If you select the **Allow decimal notation input only** option under WindowViewer properties then you cannot input non-numeric values like 1E2 in the input field. For more information on runtime settings, see Configure general WindowViewer properties in the *AVEVA™ InTouch HMI Creating Standards for InTouch HMI Components* documentation.

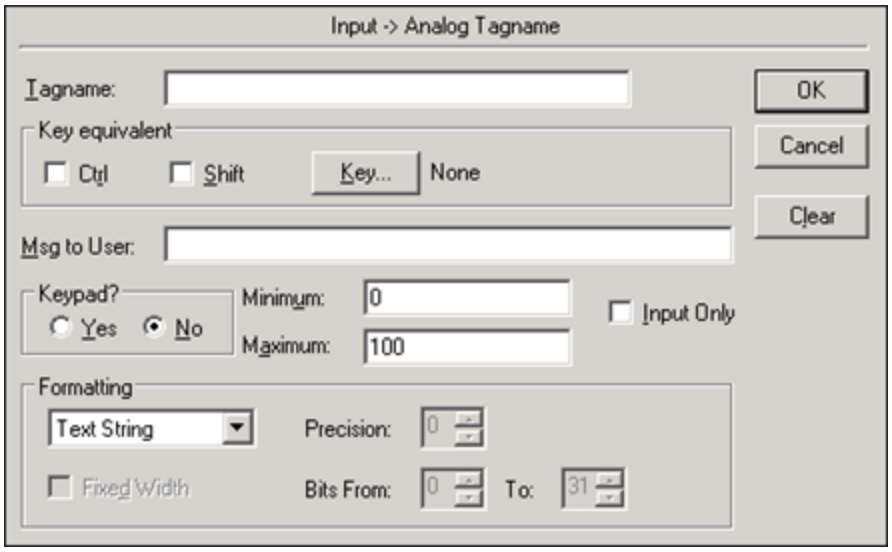
Create an analog input link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Touch Links** area, under **User Inputs**, select **Analog**.

   The **Input -> Analog Tagname** dialog box appears.



4. In the **Tagname** box, type the name of an analog tag.
5. In the **Key Equivalent** area, assign a key equivalent to the link. For more information, see Create keyboard shortcuts.
6. Configure the analog input details. Do the following:
   - In the **Keypad?** area, select **Yes** if you want to show an on-screen numeric keypad to enter a value. In the **Msg to User** box, type the prompt message that you want to appear in keypad.
   - In the **Minimum** and **Maximum** boxes, type the minimum and maximum input values for the tag. These settings limit the user input for the animation at run time. Constant analog values (integer or real) are supported, as well as references. The default values are 1 and 100, respectively. The maximum value must be greater than the minimum value.

     If you are using a reference, see the Using References for the Minimum and Maximum Limits of an Analog Input Animation section for more information.
7. Select the **InputOnly** checkbox to prevent the input from appearing in a text field associated with the object.
8. In the **Formatting** area, in the list, select each data type for which you want to configure run time advanced formatting. The **Fixed Width** checkbox, **Precision** box, and **Bits From** and **To** boxes become available based on the data type you select. For information on configuring these options, see the Advanced Formatting for Text section.
9. Select **OK**.

Use references for the minimum and maximum limits of an analog input animation

Constant analog values and references are supported for the minimum and maximum limits of an analog user input animation. You configure these limits using the **Input -> Analog Tagname** dialog box.

A reference can either be a remote analog tag or a local analog tag like a memory tag or an I/O tag.

- When InTouch I/O tags are used for the minimum and maximum values, only the InTouch I/O tags are evaluated. If the I/O tag refers to a remote tag reference that is assigned a string value, then the I/O tag

value is evaluated as 0.

- When remote tag references are used for the minimum and maximum values, the references are internally evaluated as strings. If the remote references are assigned with string values that fail to get converted to analog values, a warning message appears in the Logger. The **Minimum** and **Maximum** boxes show values based on the data type of the reference to which user input is configured.
- When the user input in the **Minimum** or **Maximum** field contains the letter 'e', the input is either a string or an exponential value. When the user input contains numbers followed by the letter 'e', followed by numbers (±N e ±N format, where N is a number), the user input is considered as an exponential value. When the user input containing 'e' contains any other alphabet (even another 'e'), the user input is considered as a reference name.

If a reference goes into bad quality at run time, the following occurs.

For the reference in the **Minimum** box:

- If the primary reference of the animation is an InTouch tag, the tag's configured minimum value is retrieved from the tag database and used for the minimum value.
- If the primary reference of the animation is an external reference, the minimum value of the data type is used. If the configured data point is a string, the last known value is used to determine the data type.
  - If the last known value contains a decimal point, the data type is real.
  - If the last known value does not contain a decimal point, the data type is an integer.
  - If there is no last known value, then integer is used.

For the reference in the **Maximum** box:

- If the primary reference of the animation is an InTouch tag, the tag's configured maximum value is retrieved from the tag database and used for the maximum value.
- If the primary reference of the animation is an external reference, the maximum value of the data type is used. If the configured data point is a string, the last known value is used to determine the data type.
  - If the last known value contains a decimal point, the data type is real.
  - If the last known value does not contain a decimal point, the data type is an integer.
  - If there is no last known value, then integer is used.

Enable string input

You can create an object to enable users to input text strings such as batch names, operator ID, or passwords by using string input links.

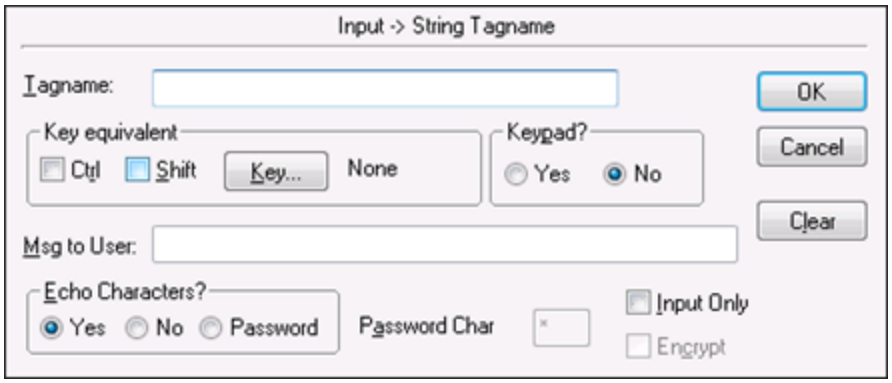
Create a string input link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Touch Links** area, under **User Inputs**, select **String**.

   The **Input -> String Tagname** dialog box appears.



4. In the **Tagname** box, type the name of a message tag.
5. Optionally configure a key equivalent and/or keyboard.
   - In the **Key Equivalent** area, assign a key equivalent to the link. For more information, see Create keyboard shortcuts.
   - In the **Keypad?** area, select **Yes** if you want to show a keyboard for inputting the new value. In the **Msg to User** box, type the message to appear in keyboard.
6. In the **Echo Characters?** area, select if you want characters to appear in the input box as the user types the string.
   - Select **Yes** to show the typed text in the input box.
   - Select **No** to not show the typed text.
   - Select **Password** to use a "masking" character instead of the typed text. In the **Password Char** box, type the masking character. Select the **Encrypt** checkbox to encrypt the password.

     Important: Password encryption only works within the context of the InTouch HMI. Do not encrypt the string if you want to pass it to an external security system, such as the operating system or a SQL Server database. The external security system cannot read the encrypted password string and access will fail.
7. Select the **InputOnly** checkbox to prevent the input from appearing in a text field associated with the object.
8. Select **OK**.

Enable sliders

You can create objects that users can drag back and forth through the use of slider touch links. As the user moves the object, it alters the value of the tag linked to it. You can link an object to a horizontal or a vertical slider.

When you make an object into a slider, you set the reference location, which is the point on the object the cursor uses to lock onto it.

You can use both horizontal and vertical links on a single object, so that the value of two analog tags are altered simultaneously.

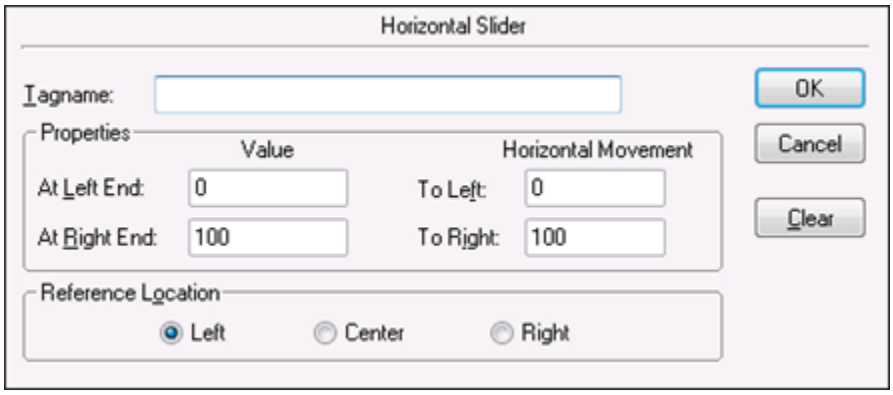
Create a horizontal slider link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Slider** area, select **Horizontal**.

The **HorizontalSlider** dialog box appears.



4. In the **Tagname** box, type the name of an analog tag.
5. In the **Properties** area, do the following:
   a. In the **At Left End** box, type the value for the tag when the slider is in its farthest left position.
   b. In the **At Right End** box, type the value for the tag when the slider is in its farthest right position.
   c. In the **To Left** box, type the number of pixels the slider can move to the left.
   d. In the **To Right** box, type the number of pixels the slider can move to the right.
6. In the **Reference Location** area, select the location on the object that the cursor will use to lock onto the object.
7. Select **OK**.

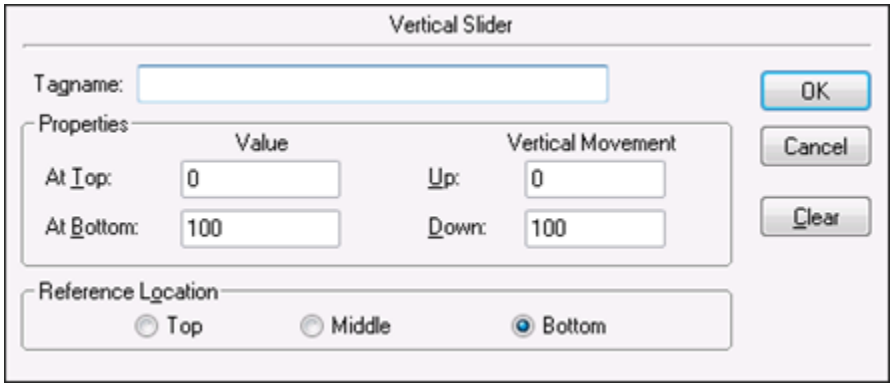
Create a vertical slider link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Slider** area, select **Vertical**.

   The **VerticalSlider** dialog box appears.



4. In the **Tagname** box, type the name of an analog tag.
5. In the **Properties** area, do the following:
   a. In the **At Top** box, type the value for the tag when the slider is in its farthest up position.

b. In the **At Bottom** box, type the value for the tag when the slider is in its farthest down position.
c. In the **Up** box, type the number of pixels the slider can move up.
d. In the **Down** box, type the number of pixels the slider can move down.

6. In the **Reference Location** area, select the location on the object that the cursor will use to lock onto the object.
7. Select **OK**.

Enable push buttons

You can create touch sensitive objects that start action scripts by using touch push button links.

Action scripts can set tags to specific values, start and control other applications, execute functions and so on.

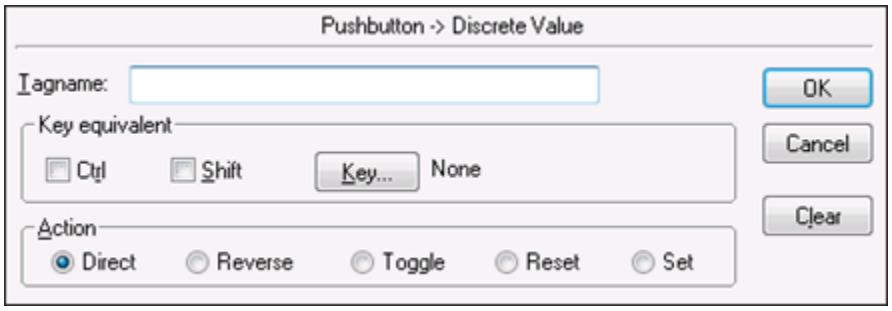
Create a discrete value touch link

1. Select the object you want to apply the animation to.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Touch Pushbutton** area, select **Discrete Value**.

   The **Pushbutton -> Discrete Value** dialog box appears.



4. In the **Tagname** box, type a discrete type tagname.
5. Select **Key** to assign a key equivalent to the link.
6. In the **Action** area, select one of the following types:
   - Select **Direct** to set the value equal to 1 as long as the push button is pressed and held down. The value automatically resets to 0 when the button is released.
   - Select **Reverse** to set the value equal to 0 when the push button is pressed and held down. The value automatically resets to 1 when the button is released.
   - Select **Toggle** to reverse the state of the discrete tag.
   - Select **Reset** to set the value equal to 0.
   - Select **Set** to set the value equal to 1.
7. Select **OK**.

Create an action script link

1. Select the object you want to apply the animation to.

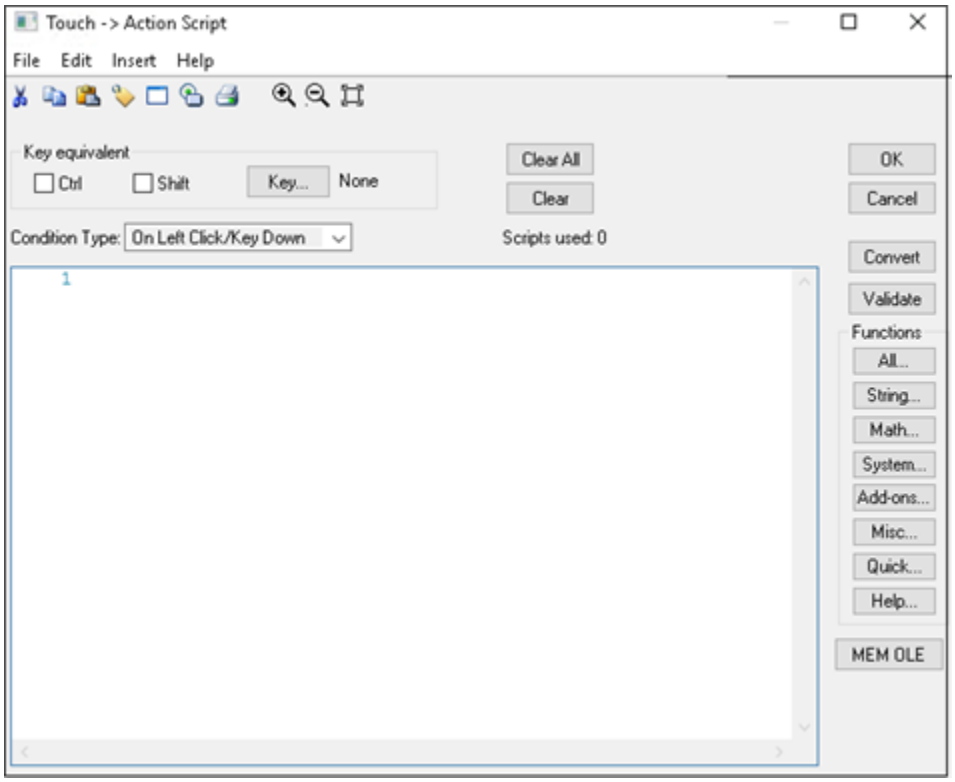
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.

3. In the **Touch Pushbutton** area, select **Action**.

   The **Touch-> Action Script** dialog appears.



4. In the **Condition Type** list, select a script type. For more information on the types of action scripts, see Script triggers.

   Note: If you assign a key equivalent link to an action push button and to the same key is used for a key script, the key equivalent link you assign takes precedence over the key script.

5. In the **Script Editor** window, type the script to execute when the object is activated.
6. Select **OK**.

Open and close windows

You can create touch links to open and close other InTouch windows by using show window and hide window links.

You can program objects to open more than one window at a time. However, if you program your link to open more than one window, be aware of any intersections and window types.

If one of the opening windows is a replace type window and intersects another opening window, the other window closes before opening.

Create a show (or hide) window link

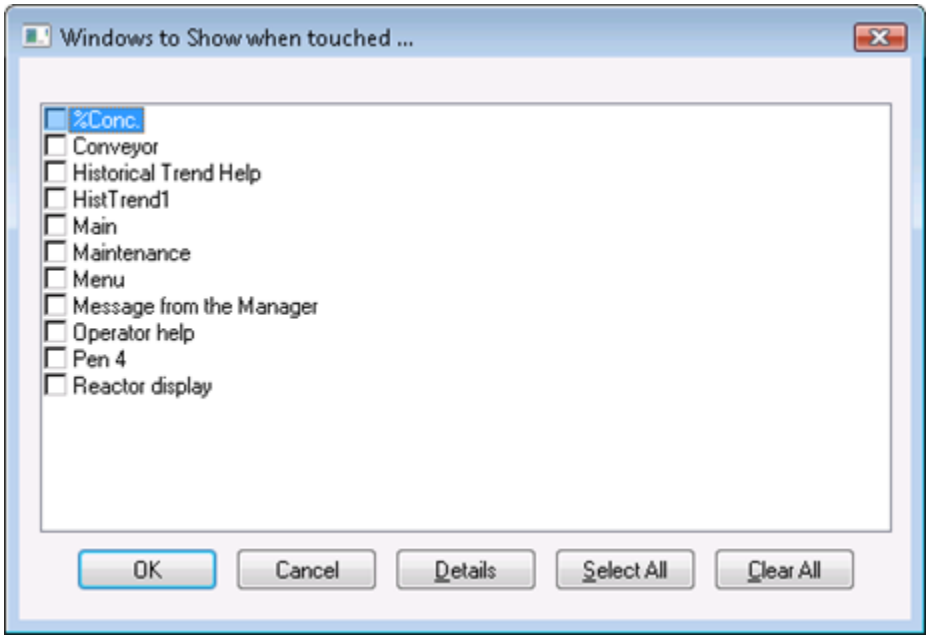
1. Select the object you want to apply the animation to.

2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
3. In the **Touch Pushbutton** area, select **Show Window** or **Hide Window**.

   The **Windows to Show when touched** or **Windows to Hide when touched** dialog box appears.



4. Select the window(s) to open or hide.
5. Select **OK**.

Configure on-screen keyboards

On-screen keyboards enable operator input in situations where a keyboard is not connected to the computer. You can designate one of three on-screen keyboard types.

- The standard InTouch keyboard or keypad. This is the default keyboard.
- The Windows system keyboard. The Windows keyboard is a fully functional QWERTY-type keyboard with function keys, print screen key, number lock key, directional arrows, and so on.
- The re-sizable keyboard or keypad. This keyboard can be resized during run time.

You can also open the keyboard by using the DialogStringEntry() and DialogValueEntry() function in a script. For more information, see DialogStringEntry() function and DialogValueEntry() function.

Configure the on screen keyboard type

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.
3. On the **Preferences** tab, in the **Keyboard** area, select type of keyboard you want.
4. If you select the re-sizeable keyboard, choose **Options** to select font, location, and dimension properties of

the keyboard.

5. Select **OK**.

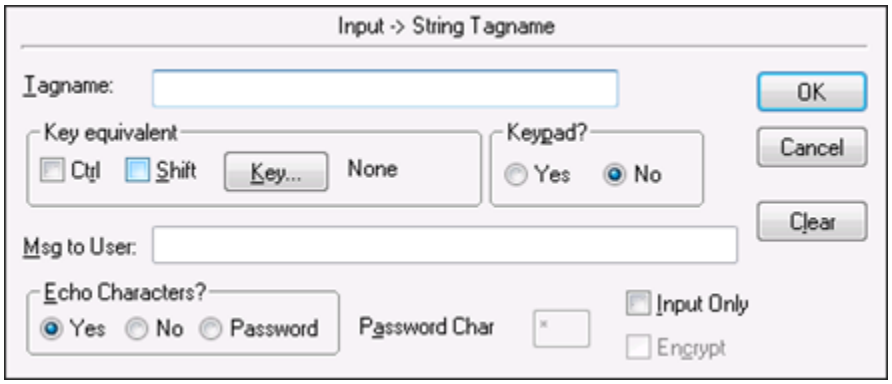
Make an on screen keyboard appear

1. Configure the on-screen keyboard type.
2. Select the object you want to apply the animation to.
3. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

   The **Animation Links** window appears.
4. In the **Touch Links** area, under **User Inputs**, select **String**.

   The **Input->String Tagname** dialog box appears.



5. In the **Keypad?** area, select **Yes**.
6. Select **OK**.

DialogStringEntry() function

Shows an alphanumeric keyboard on the screen, enabling the operator to change the current string value of a message tag in the Tagname Dictionary.

Category

misc

Syntax

```
[Result=]DialogStringEntry(MessageTag_Text, UserPrompt_Text);
```

Parameters

MessageTag_Text

The name of the message tag to be modified. This value is a string value. Specify the tagname within quotes or use the .Name dotfield without quotes. You can also use a message tag as a pointer.

UserPrompt_Text

The user message to show at the top of the keyboard.

Return Value

Returns one of the following integer values:

0 = Cancel was pressed.

1 = OK was pressed.

-1 = Internal error.

-2 = Could not initiate.

-3 = Tagname not defined.

-4 = Tagname is not a Message type.

-5 = Unable to write.

Remarks

This function is used primarily in applications containing touch screens.

Example

```
Errmsg=DialogStringEntry(MyMessageTag.Name, "Enter a new string...");
Errmsg=DialogStringEntry("MyMessageTag","Enter a new string...");
```

For example, the following script opens an alphanumeric keyboard, allowing modification of MyMessageTag while showing the message "Enter a new string..." at the top of the keyboard:

```
MessageTagX="MyMessageTag"; {assign the string MyMessageTag (which is actually the tagname
to be modified) to the Memory Message tagname MessageTagX}
MessageDisplay="Enter a new string..."; {assign the new message string to the Memory
Message tagname MessageDisplay}
Errmsg=DialogStringEntry(MessageTagX, MessageDisplay); {quotes are not required because
MessageTagX was defined as a Message tagname}
```

See Also

DialogValueEntry()

DialogValueEntry() function

Shows the numeric keypad on the screen, enabling the user to change the current value of a discrete, integer or real tag.

Category

misc

Syntax

```
[Result=] DialogValueEntry(ValueTag_Text, LowLimit, HighLimit, UserPrompt_Text);
```

Parameters

ValueTag_Text

The name of the discrete, integer, or real tag to modify. This value is a string value. Specify the tagname within quotes or use the .Name field without quotes. You can also use a message tag as a pointer.

LowLimit

The minimum value of the tag. (This should be >= the tagname's definition for minimum value, minimum raw or minimum engineering unit, as applicable).

HighLimit

The maximum value of the tag. (This should be <= the tagname's definition for maximum value, maximum raw or maximum engineering unit, as applicable).

UserPrompt_Text

The user message to show at the top of the keypad.

Return Value

Returns one of the following integer values:

0 = Cancel was pressed.

1 = OK was pressed.

-1 = Highlimit<=Lowlimit.

-2 = Could not initiate.

-3 = Tagname not defined.

-4 = Tagname is not a Discrete, Integer or Real type.

-5 = Write failed.

Remarks

This function is used primarily in applications containing touch screens.

Example(s)

```
Errmsg=DialogValueEntry(MyIntegerTag.Name, MyIntegerTag.MinEU, MyIntegerTag.MaxEU, "Enter a
new value...");
Errmsg=DialogValueEntry("MyIntegerTag", -100, 100, "Enter a new value...");
```

For example, the following script brings up the numeric keypad, allowing modification of MyIntegerTag using a minimum and maximum limit of -100 and 100 (respectively), while showing the message "Enter a new value..." at the top of the keypad:

```
TagnameX="MyIntegerTag"; {assign the string MyIntegerTag (which is actually the tagname to
be modified) to the Memory Message tagname TagnameX}
Min=-100; {assign the minimum value allowed for the tagname to the Memory Real/Integer
```

```
tagname Min}
Max=100; {assign the minimum value allowed for the tagname to the Memory Real/Integer
tagname Max}
MessageDisplay="Enter a new value..."; {assign the new message string to the Memory Message
tagname MessageDisplay}
Errmsg=DialogValueEntry(TagnameX, Min, Max, MessageDisplay); {quotes are not required
because TagnameX was defined as a Message tagname. By assigning a Discrete, Integer or Real
tagname to TagnameX, the function will modify that assigned tagname}
```

See Also

DialogStringEntry()

Common animation tasks

Common animation tasks include selecting tags, creating keyboard shortcuts, changing tagname references and replacing placeholder tags.

Select tags or attributes

Use the **Select Tag** dialog box, also called the tag browser, to select

- Tags defined in a local or remote InTouch application.
- Application Server object attributes with the Galaxy Browser.

You open the **Select Tag** dialog box by double-clicking in any text box that requires a tag name for input.

Select an InTouch tag

You can select tags defined in a local or remote InTouch application. You can create references to tags in any tag source that supports the Tagname Dictionary interface.

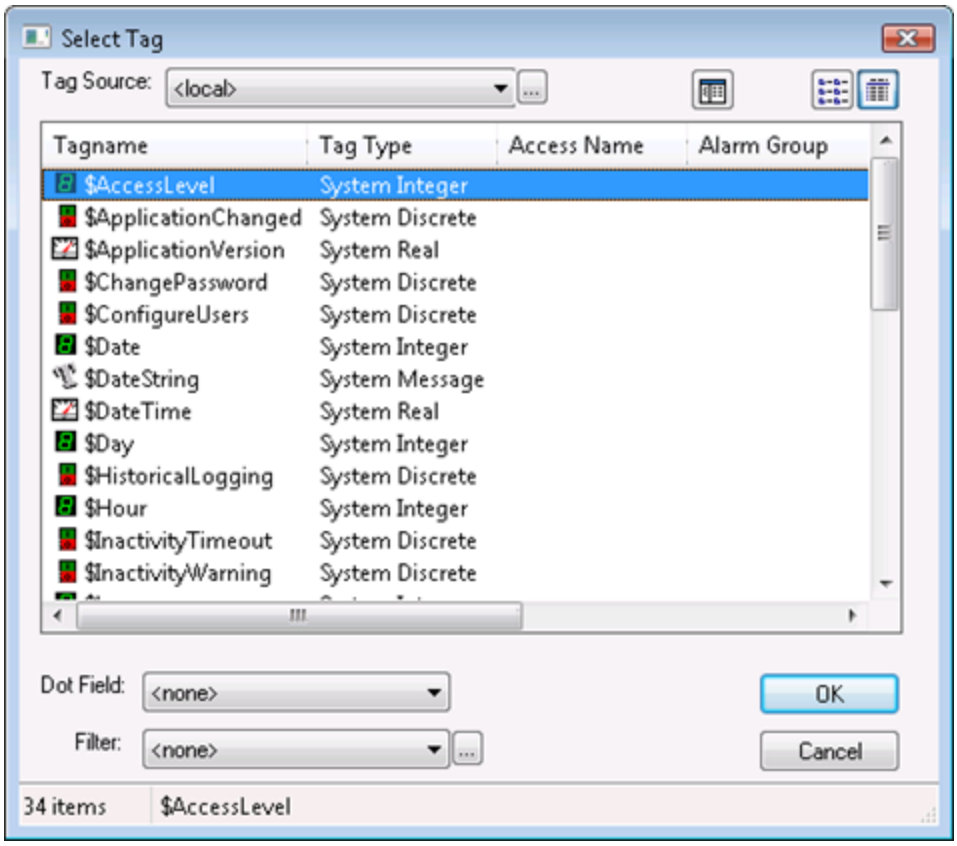
For example, remote tag references enable your application to access data from I/O Servers without creating tags in the local Tagname Dictionary.

You can set a dotfield for each InTouch tag you select. Dotfields can access, monitor, and modify tag properties. If you do not select a dotfield, the .Value dotfield is used. For more information about dotfields, see Use tag dotfields to view or change tag properties.

Select an InTouch tag

1. Double-click any text box that requires a tag name for input.

   The **Select Tag** dialog box appears.



2. In the **Tag Source** list, select the name for the tag source or select the browse button to define a new tag source to use.

   For more information on defining tag sources, see Data access with I/O.

3. In the **Filter** list, select a filter to reduce the number of tagnames shown in the window. To define a filter, select the ellipsis button. For more information, see Create a tag filter.

4. Select a tagname in the window.

   You can change the way the tagnames are shown in **Select Tag** dialog box. For more information, see Change the view in the Select Tag dialog box.

5. In the **Dot Field** list, select a dotfield to append to the selected tagname.

   Dotfields can access, monitor and modify tag properties. If you do not select a dotfield, the .Value dotfield is used.

6. Select **OK**.

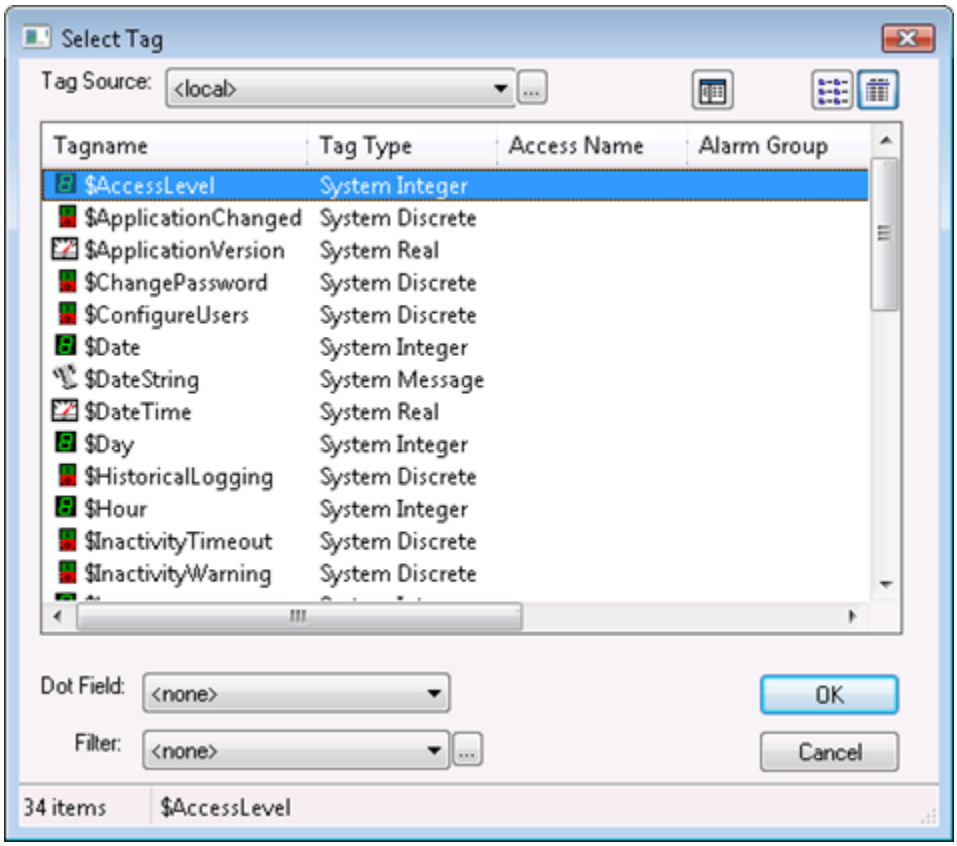
Select an application server object attribute

You can select attributes that are associated with an Application Server object. To do this, you must first add the Galaxy that contains the object as a data source for your InTouch application. For more information about data sources, see Data access with I/O.

Select an object attribute

1. Double-click any text box that requires a tag name for input.

The **Select Tag** dialog box appears.



2. In the **Tag Source** list, select the name of the tag source that uses the Galaxy or select the browse button to define a new tag source to use.

   The **Galaxy Browser** dialog box appears.

3. Use the **Galaxy Browser** to find and select an object attribute. For more information, see the Application Server documentation.
4. Select **OK** to close the **Galaxy Browser**.

   The attribute reference appears in the text box that requires a tag name.

   Note: To return to the **Select Tag** dialog box from the **Galaxy Browser**, select the **Back** button in the bottom right corner of the **Galaxy Browser**.

Create a tag filter

If you have a large number of tags in your Tagname Dictionary, finding the tag for an animation can be cumbersome. For example, you may only want to see the tags assigned to a particular Access Name or Alarm Group. You can configure a filter so that only a subset of tags are shown in the **Select Tag** dialog box.

You can use the following wildcard expressions in your filter.

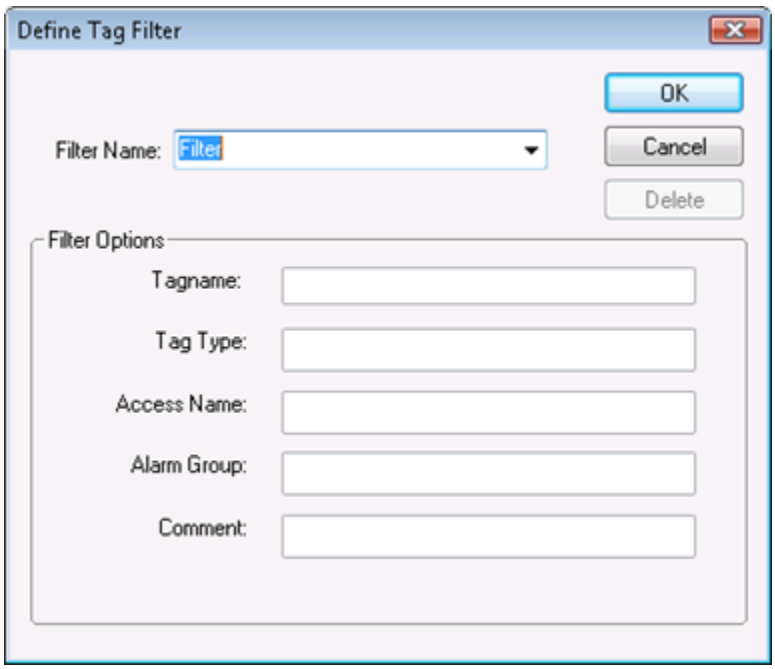
- The multiple character wildcard is the asterisk (*). For example, "Asyn*" searches for all tagnames beginning with the characters "Asyn".
- The single character wildcard is the question mark (?). For example, the "Tag?" filter searches for all four-

character tagnames that begin with "Tag". The "Tag*" filter searches for all tagnames that begin with "Tag".

- Any sequence of valid tagname characters, together with the two wildcard characters, is acceptable in a filter. Valid tagname characters are: A-Z, a-z, 0-9, !, @, -, #, $, %, _ and &.

Define a search filter

1. In the **Select Tag** dialog box, select the ellipsis button next to the **Filter** list. The **Define Tag Filter** dialog box appears.



2. In the **Filter Name** box, type a filter name.
3. In the **Filter Options** area, configure the filter criteria. Do any of the following:
   - In the **Tagname** box, type the tag name.
   - In the **Tag Type** box, type the tag type.
   - In the **Access Name** box, type the local Access Name.
   - In the **Alarm Group** box, type the Alarm Group name.
   - In the **Comment** box, type the comment expression.
4. Select **OK**. The **Filter Name** appears in the **Filter** list in the **Select Tag** dialog box. You can select the filter to show the tags meeting the filter criteria.

Delete a search filter

1. In the **FilterName** box, select the filter.
2. Select **Delete**.

Change the view in the Select Tag dialog box

The **Select Tag** dialog box has three different views: List, Details, and Tree View.

To see the	Select the	Description
List view	List view button 	Small icons appear next to the tagnames according to the type of each tagname.
Details view	Details view button 	You see the same small icons and tagnames, plus Tag Types, Access Name, Alarm Group, and Comments. Sort the list by selecting the column headers.
Tree View	Tree view icon 	The Tree View displays the tagnames in two views. You can access the member tagnames in any SuperTag template.

Create keyboard shortcuts

You can assign a specific key on the keyboard to activate certain animation links by using key equivalent links. The key equivalent is only operational when the object with the link is visible or selected. If the object has a visibility or disable link, the key equivalent is not active when the object is disabled.

You can define the same key in multiple windows. However, the definition in the most recently opened window is the active one. In the case of overlay windows, the key is active in the window on top.

Note: If any object or action push button in the active window is assigned to the same key used for a key action script, the key equivalent link on the key in the active window takes precedence over the execution of the key action script.

The animation links that support key equivalents appears the **Key Equivalent** area in their link dialog boxes.

Key links are only available for function keys 1-16. If you are using a custom keyboard that has more than 16 function keys, get a device driver from your manufacturer to access the extended function keys on your system.

Assign a key to a link

1. Open the **Animation Links** dialog box for the type of link you are configuring.
2. Select **Ctrl** and/or **Shift** if you want the operator to hold down either or both of these keys when pressing the key equivalent.
3. Select **Key**. The **Choose key** dialog box appears.
4. Select the key to assign to the link.

   The **Animation Link** dialog box reappears with the name of the selected key next to the **Key** button.
5. Select **OK**.

Change tagname references

When you duplicate an object, you get an exact copy of the original including links, animation, scripts and so on. However, if you need to use a different tag on a duplicated object, you must substitute a tag.

You can select and substitute a tag for any object and you can select multiple objects and substitute all their tags at the same time.

If your system license supports a limited number of tagnames, convert your local tags to remote tag references to reduce the number of tags defined in your local Tagname Dictionary.

Substituting a tag works for all tags and references.

Substitute a tag with another tag

1. Select the object(s) associated with the tag to change to another tag.
2. On the **Animation** menu, in the **Substitute** group, select **Tags**.

   The **Substitute Tagnames** dialog box appears.
3. In the **New Name** box, type the new tagname.
   - If you double-click a tag in the **New Name** box, its definition in the Tagname Dictionary appears.
   - If you erase the tagname then double-click the blank box, the **Select Tag** dialog box appears.
4. Select **OK**.

   The tag associated with the object is automatically changed.

Convert placeholder tags

If you import or export a window or QuickScript to or from your current application, all the tags associated with that window or QuickScript are transferred with the window.

The local tags are not automatically added to your application database. Instead, they are automatically marked as placeholder tags.

Remote tag references are not affected, and are not marked as placeholder tags.

You must convert the placeholder tags to existing local tags, define them in the new application's Tagname Dictionary, or make them remote tag references.

Notice the placeholders ?d:, ?i:, ?m: and ?r: preceding the tags. They indicate the type that the tag was originally defined as:

Placeholder Symbol	Tagname Type
d	Discrete
i	Integer
m	Message
r	Real

Note: Remote tag references are not shown as placeholders but as remote tag references such as: **PLC2:Temperature**.

You can use several methods in the **Substitute Tagnames** dialog box to convert placeholder tags to local tags. For more information, see Export and import InTouch components in AVEVA™ InTouch HMI Application Run Time documentation.

Tip: If you manually convert tagnames and you no longer need the original tag defined in the original Tagname Dictionary, you can update the tagname use counts and then delete the unused tag.

By importing a window or QuickScript from another application, and converting all of the tagnames associated with the animation links or QuickScript(s) to remote tagname references, you can create an application that instantly receives data from hundreds of remote tagnames without defining a single tag in your local Tagname Dictionary.

Advanced formatting for text

Using InTouch WindowMaker, you can configure the intelligent or advanced formatting options, which will be used at run time by InTouch WindowViewer. These options only apply if you are using the advanced formatting features on an analog value display or analog user input animation. The present value of InTouch WindowViewer will be retained if you do not configure advanced formatting options.

The advanced formatting options appear in the **Input -> Analog Tagname** dialog box and the **Output -> Analog Expression** dialog box.

Advanced formatting of numeric value display provides you with a native way of formatting numerical data based on data type and data quality. Advanced formatting enables you to format analog values at run time in InTouch WindowViewer.

You can either use text string or enable the advanced formatting options on an individual animation basis. When you migrate an animation from an earlier InTouch version, or create an animation for the first time, text string is set as the default setting.

The advanced formatting configuration options are:

- Text string, which uses the existing InTouch formatting.
- Real, which is formatted as the real value based on the size of the value at run time. The global decimal precision configuration is used to determine the number of decimal places.
- Fixed decimal point, of which the formatting depends on the type of values:
  - Real values: The value is formatted with the user-specified number of decimal places configured using the **Precision** control.
  - Integer values: The value is formatted as an integer without a decimal point and is indented on the right to accommodate for where the decimal places would have been.
  - Discrete values: The value is formatted as a 0 or 1 without a decimal point and is indented on the right to accommodate for where the decimal places would have been.
- Integer, which is always formatted as an integer without a decimal point.
- Exponential, which is always formatted as an exponent taking into consideration the fixed decimal point.
- Hex, which is displayed in hexadecimal format based on the configured Boolean range.
- Binary, which is always formatted as a binary string taking into consideration the configured Boolean range.

In the **Fixed Width** box, you can increase the text size specified in design time. This option is not available if **Text String** formatting is selected for the formatting style.

At design time, if you have selected the **Fixed Width** checkbox in the animation configuration, then the value rendered at run time does not exceed the length of the text field. If the value exceeds the length of the text field, the globally configured "Fixed Field Width Too Large Character" takes the length of the text field. Formatting with fixed width stops the text element from expanding horizontally at run time to accommodate a value size.

If you do not specify any text for the text field at design time, the fixed width setting is ignored and the full value is shown. For instance, if you enter "" (no text) at design time, it is equivalent to 0 for the text field width. This does not allow any text to be rendered if the **Fixed Width** option is enabled. In this situation, the **Fixed Field**

Width setting is set to "None" as default at run time.

If you select **Clear**, the advanced formatting options are reset to the following:

- **Text String** is selected for the **Formatting** option.
- The **Fixed Field Width** option is cleared and subsequently disabled since this option is not available when **Text String** formatting is selected.

The following values are set by default:

- 0 is set for the Fixed Decimal precision control.
- 0 is set for the Exponential precision control.
- 0 and 31 are set for the Hex bit range controls.
- 0 and 31 are set for the Binary bit range controls.

Format Controls:

- The Precision control accepts an integer value and is only enabled if you have selected the **Fixed Dec. Point** or **Exponential** options. The valid range of integer values for this control is from 0 through 8, with 0 being the default value. The bit range consists of two spin controls, each of which takes integers as inputs. These controls are only available if you have selected either the **Hex** or the **Binary** option. For these two controls, you need to specify the bit value, ranging from 0 through 31. You can specify the bit range in the forward or reverse order. You can also specify a single bit, using the same bit specification for the range. The default numeric range for the bit specification is 0 through 31. The range is zero based to stay consistent with other places in InTouch where bits are specified.

For versions of InTouch earlier than 3.5 that contain configured analog user input animations or analog value display animations, the migrated animation is set to **Text String** formatting and advanced formatting is off.

For later versions, the advanced formatting options are migrated forward.

Format text objects as numbers

Text objects can be used to display static or dynamic numeric values. You can show the integer or real value of an analog tag by associating a Value Display - Analog animation link to a text object.

To specify the format of an analog tag value shown by animating a text object, the following characters are used:

Character	Description
#	Forces the display of an analog value to an integer.
0	Represents a digit at each specified position of a real number. Forces leading zeros to the integer part of a number.
,	Inserts a comma at the specified position of a real number.
.	Places a decimal point at the specified position of a

Character	Description
	real number.

Other text format properties like font, size, and color apply to numeric values shown in the string.

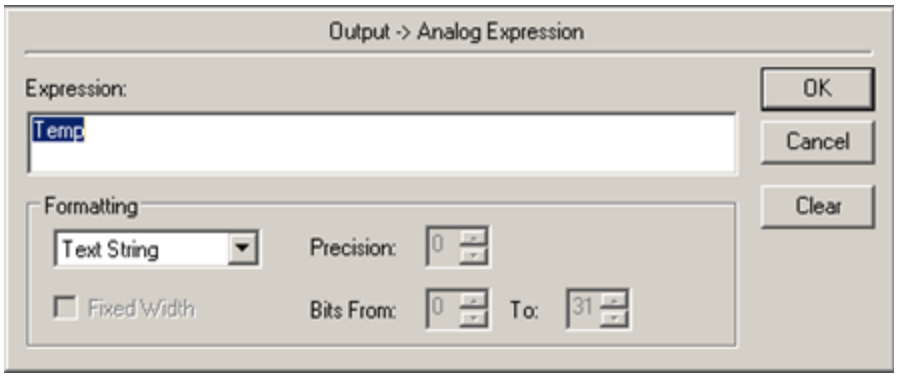
Examples of formatting text objects as numbers

The following table shows examples of displayed analog values when animated format characters are used in a text object. In these examples, the current value of a temperature I/O tag is 123.45 Cº. This tag is assigned as the reference value of a text object using an Analog animation link.

Formatted Text Object	Displayed Value of the Text Object
Temp = # C	124 C Displays the temperature as an integer. Only one # needs to be entered in the text object.
Temp = #.# C	123.5 C The fractional part of the temperature is rounded to a single number using the # character. The integer part of the temperature is unchanged.
Temp = 00000 C	00124 C Adds leading zeros to the integer part of the temperature.
Temp = 000.0 C	123.5 C The fractional part of the temperature is rounded to a single digit because of the single zero in the format.
Temp = 00.00 C	123.45 C The original temperature value remains unchanged.
Temp = 0,000.#	0,123.5 C A leading zero and comma are forced at the integer part of the temperature reference value. The fractional part is rounded to a single number using the # character.

Specify the format of text objects as numbers

1. Create a text object in a window and insert one or more characters to format a reference value associated with the text object.
2. Double-click the text object to show the **Animation Links** selection dialog box.
3. In the **Value Display** section, select **Analog**. The **Output -> Analog Expression** dialog box appears.



4. In the **Expression** field, type an analog tagname or expression.
5. Select **OK**.
6. Save your changes to the window.
7. Run the application to verify the number appears correctly in the text object.

Work with the Industrial Graphic Editor (IGE)

The Industrial Graphic Editor is the tool you use to create an industrial graphic. Refer to Industrial Graphic Editor User documentation for instructions on using the editor. The following sections will list any InTouch HMI specific configurations or settings to be used with the Industrial Graphic editor.

Manage Industrial Graphics

Industrial Graphics are graphics you can create to visualize data in an HMI/SCADA system.

You use the Industrial Graphic Editor to create Industrial Graphics from basic elements, such as rectangles, lines, and text elements. You can also use the Industrial Graphic Editor to embed and configure an Industrial Graphic from the Graphic Toolbox library of graphics.

The Industrial Graphic Editor supports import of Scalable Vector Graphics (SVG) as Industrial Graphics. You can perform the following:

- Import an SVG into the Industrial Graphic Editor. The graphic elements automatically get converted to an Industrial Graphic, which can include many primitives.
- Insert the SVG while using the **Image** icon from the **Tools** panel of the Industrial Graphic Editor.
- Use SVG for the **UpImage** and **DownImage** on a button.
- Use SVG for the **Quality & Status** display icons under **Styles** configuration.

For more information on importing SVG and SVG limitations, refer to the "Importing of SVG as an Industrial Graphic" section of the Industrial Graphic Editor help.

After you create an Industrial Graphic, you can embed it into another graphic or an HMI system window and use it at run time.

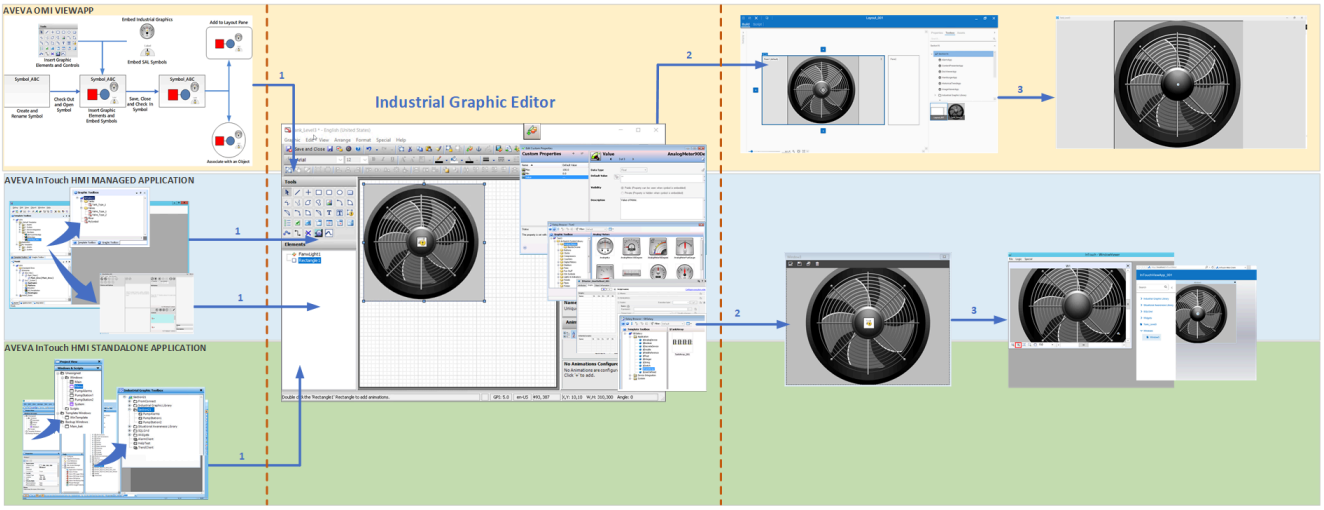
You can embed an industrial graphic in a template or instance of an object providing several ways to visualize object-specific information quickly and easily. Embedding a graphic in a template means that you can update one graphic and cascade the changes throughout your application.

Depending on your development requirements, you can select where and how to store industrial graphics.

- Create and store graphics as a standard set that you can reuse, such as a generic valve graphic.
- Store graphics as templates if you want to use the graphics in multiple instances at run time.
- Store graphics for use in a specific application.

Industrial Graphic Editor workflows for use with HMI/SCADA applications

The Industrial Graphic Editor is the standard graphic editor used across various HMI/SCADA applications. It can be used to create graphics that can be embedded into HMI/SCADA applications to represent assets on the plant floor. The overall workflow is described in the following sections.



InTouch managed applications

The workflow for adding a graphic to an InTouch HMI managed application starts with the IDE.

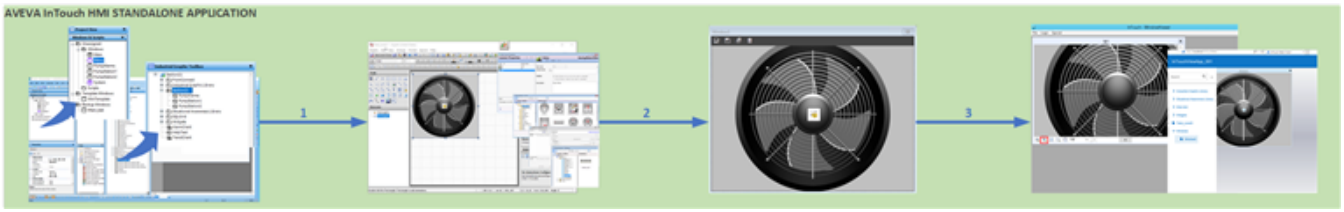
1. Create or edit a graphic, either
   a. From the Visualization folder:
      a. Create a new graphic or select an existing graphic.
      b. Open the graphic to launch the Industrial Graphic Editor.

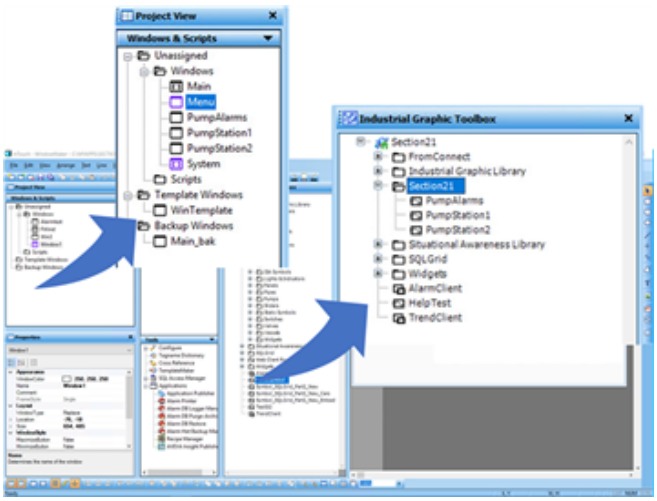
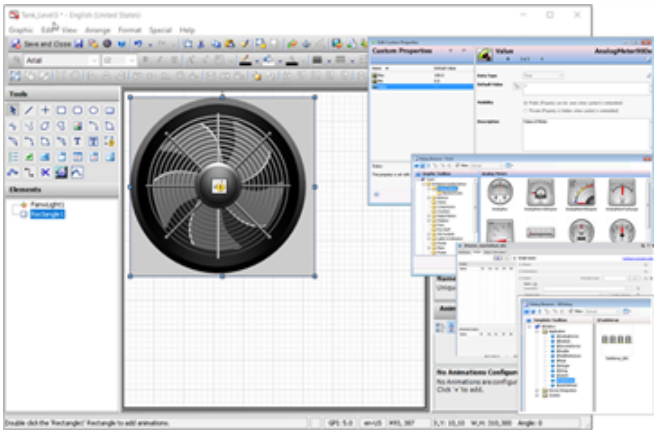
   - or -

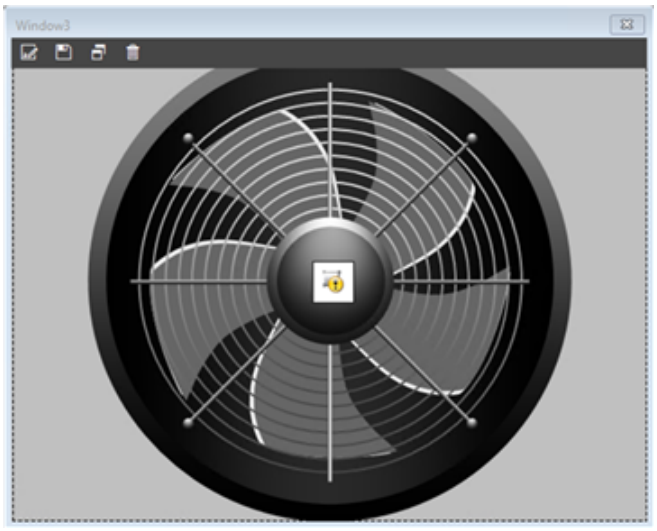
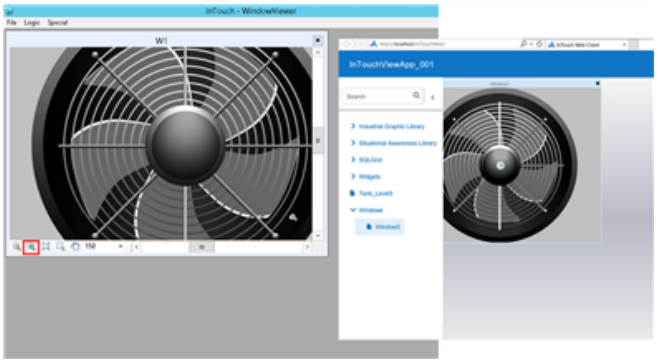
   a. From the Object Editor for the object you want to associate the graphic with:
      a. In the Content pane, select the **+** button to add a graphic to the object.
      b. Select the **Edit Content** button to launch the Industrial Graphic Editor.
2. Edit the graphic in the Industrial Graphic Editor. Modify graphical elements as needed:
   - Insert graphic elements and embed graphics.
   - Add/modify scripts, references, custom properties, and animations to enable the graphic to represent the asset, and to respond to commands and data changes at run time.
3. Embed the graphic into a window.
4. To view the graphic in runtime, launch Window Viewer or the InTouch web client.

InTouch standalone applications

The workflow for InTouch HMI standalone applications starts with WindowMaker.



Step 1 	In InTouch HMI WindowMaker create symbols using the Industrial Graphic Editor. Edit the graphic to launch the Industrial Graphic Editor.
Step 2 	In the Industrial Graphic Editor use scripting, custom properties, tag and object references, styles, tools, animations and formatting to build the graphics.
Step 3	Embed the graphic in a Window.

	
Step 4 	To view the graphic in runtime, launch WindowViewer or the Web Client.

Symbol change propagation

Any changes to a source Industrial graphic are propagated to all embedded Industrial graphics. This affects the Industrial graphics in WindowMaker and the Industrial graphics inherited by AutomationObjects.

If a change is made to an Industrial graphic and it is used in an open InTouch window, the Symbol Changed icon appears in the bottom right corner of the status bar.

When you double-click this icon, the embedded Industrial graphic is updated with the changes.

The following example shows you how symbol change propagation works.

Propagate a symbol change

1. Follow the example from Create new Application Server object instances automatically.
2. In WindowMaker, open the window that shows the Valve symbol.
3. Open the Industrial graphic ValveSymbol that is hosted by the Application Server object template $Valve1.
4. Make some changes and select **Close and Save**. The changes are propagated to Application Server object instance Valve1_E22. In WindowMaker, the Symbol Changed icon appears.
5. Double-click the icon to accept the changes. The embedded Industrial graphic is updated.

Symbol dynamic size propagation

You can control the way that size changes of the source symbol are propagated to its embedded symbols. For example, a size change is:

- Resizing one of the elements in the source symbol so that the symbol boundary changes.
- Adding elements to or removing elements from the source symbol so that the symbol's boundary changes.

This feature is called dynamic size change propagation and can be enabled or disabled.

For more information about dynamic size propagation, see Industrial Graphic Editor User documentation.

Compare WindowMaker and Industrial Graphic Editor

You can use the Industrial Graphic Editor to do most of the tasks you do in InTouch WindowMaker. You can also use many of the same shortcut keys.

The Industrial Graphic Editor features that are not available in InTouch WindowMaker:

- Additional elements.
- Additional and enhanced appearance of the elements.
- Additional and enhanced design-time functionality.

Appearance

The Industrial Graphic Editor extends the InTouch graphic configuration. For example, you can use:

- Gradients for line, fill, and text color.
- Patterns for line, fill, and text color.
- Textures for line, fill, and text color.
- Partial transparency.
- Fill behavior in relation to a symbol or screen.

Elements

Elements are the graphical objects you use to create an Industrial Graphic. The Industrial Graphic Editor provides elements that are not available in InTouch WindowMaker, such as:

- Curves and closed curves.
- Arcs, pies, and chords defined by two or three points.
- Status elements to conditionally show an icon depending on quality and status of attribute data .
- Path graphics that you create by joining line-based elements together to form a new closed element.
- Windows common controls, such as the Calendar control and Date Time Picker control.

Enhanced functionality

The Industrial Graphic Editor provides a entire array of enhancements to make your life easier when creating visualization for your manufacturing environment.

Usability enhancements

The Industrial Graphic Editor makes it easy to select and configure elements. You can:

- Select elements from a list as well as from the canvas. This lets you select elements beneath others without having to move them.
- View and change properties and animation (links) of an element by simply selecting it on the canvas.
- Edit elements contained in groups and path graphics without having to break the group or path graphic. This is called inline editing.

Style replication

Using the Format Painter, you can simply apply the style of one element by selecting another element, even an element of a different type.

Animation replication

Using the Industrial Graphic Editor you can copy, cut, and paste animations from one element to another element, even to an element of a different type.

Element positioning

The Industrial Graphic Editor extends the positioning feature of InTouch WindowMaker and lets you:

- Distribute elements equally in horizontal or vertical direction.
- Make elements same horizontal and/or vertical size.
- Increase or decrease horizontal or vertical space.
- Remove horizontal or vertical space between elements.
- Lock an element so that you do not accidentally move or edit it.
- Rotate any element at design time by any angle around a center of rotation.
- Apply resizing and rotating to multiple elements at the same time.
- Move the z-order of an element one level backward or forward.
- Align text within text boxes and buttons.

Group functionality

The Industrial Graphic Editor uses the concept of groups instead of the cell and symbol concepts of InTouch WindowMaker. You can:

- Embed groups within groups.
- Edit individual elements within a group (or an embedded group) without breaking up the group.
- Easily remove elements from or add elements to existing groups.

Extensibility with custom properties

You can add custom properties to a symbol or embedded symbol. You can connect custom properties to AutomationObject attributes, element properties, and even InTouch tags. You can use the custom properties as you would with any pre-defined property at design time and run time.

Note: Application Server custom properties referencing InTouch tags that have hyphens in their names will not work in run time. For example, "InTouch:TAG-1" will not work in run time.

Element styles

Element Styles define one or more of the fill, line, text, blink, outline, and status properties of graphic elements. Apply an Element Style to a graphic element to set the element to the preconfigured properties defined in that Element Style. The element's local properties that are defined in the Element Style are disabled.

Element Styles help drive standards for screen builders and others who are creating symbols.

Miscellaneous enhancements

Using the Industrial Graphic Editor, you can:

- Access the properties of the elements and custom properties of the symbol through scripting.
- Set the tab order of the elements.
- Use line end styles, such as arrows.
- Dynamically disable specific animations from elements without losing the configuration information.
- Use image meta files and other image formats.
- Use anti-aliasing to improve how the symbol is shown.

Procedures for common WindowMaker tasks and techniques

Most of the configuration that you do in InTouch WindowMaker can be easily done in the Industrial Graphic Editor. There are some differences between and similarities of graphics, animations, and scripts.

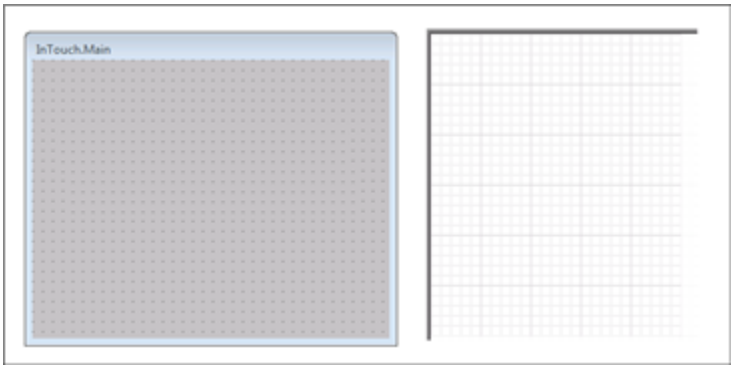
Use graphics

You can use the Industrial Graphic Editor in basically the same way as you use InTouch WindowMaker. The Industrial Graphic Editor includes a drawing area on which you can place graphical objects to construct a visual representation of production processes and to provide an interface between a human and a machine.

Some objects you use in InTouch do not exist in the Industrial Graphic Editor, such as ActiveX controls and some Wizards. Their functionality is replaced by other controls that are more powerful and can integrate better.

Use the drawing area

The drawing area of the Industrial Graphic Editor is called the canvas. You use it like an InTouch window. Its maximum size is 2,000 by 2,000 pixels.



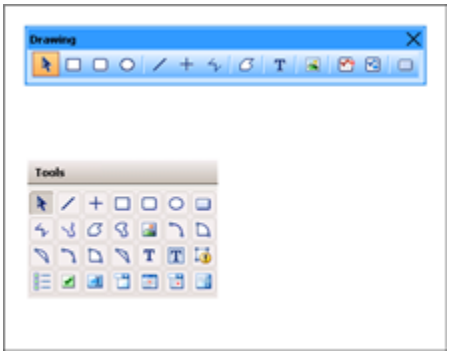
Set the drawing area color

You cannot set the drawing area color in the Industrial Graphic Editor. The drawing area color is transparent and inherits the color of the environment that the symbol is embedded into.

If you embed an Industrial graphic into an InTouch window, the area between the elements adopts the color of the InTouch window.

Use basic objects

InTouch basic objects such as rectangles, ellipses, and polylines can be drawn in very similar way in the Industrial Graphic Editor. The basic objects are called elements in the Industrial Graphic Editor.



Use complex objects

InTouch objects such as ActiveX controls, Wizards, cells, and symbols do not exist in the Industrial Graphic Editor.

You can, however, import SmartSymbols into an Industrial Graphic. When you import a SmartSymbol, the elements and animations of a SmartSymbol are converted.

In the Industrial Graphic Editor, you can create groups of elements. Groups maintain the properties of the contained individual elements. You can set the TreatAsIcon property of a group to change the behavior of the group.

Use wizards

You cannot import InTouch Wizards to an Industrial Graphic or into the Graphic Toolbox. Instead, use:

- The Industrial Graphic Library, which you can import into the Graphic Toolbox.
- Windows controls that are part of the Toolbox. You can use:
  - Radio button groups
  - Checkboxes
  - Edit boxes
  - Combo boxes
  - Calendar control
  - DateTime picker
  - List boxes

Use animations

You can use animations in the Industrial Graphic Editor to set run-time behavior of the symbols as you would in InTouch WindowMaker. You can configure one or more animations for an element or symbol. The data can come from various sources.

Configure data sources

In InTouch WindowMaker, you use the Tagname Dictionary to define variables that hold values. In the Industrial Graphic Editor, data sources can be:

- AutomationObject attributes.
- Custom properties and inherited properties of the symbol.
- InTouch tags themselves. The Industrial Graphic Editor uses a special InTouch reference you can use to directly connect to InTouch tags.

Use data types

Industrial graphics use the data types, which are different than InTouch data types.

The following table shows you the data types of both and how they correspond to each other:

InTouch	Application Server	Description
Discrete	Boolean	Boolean value. For example: 0 or 1
Integer	Integer	Integer value. For example: -4, 7, or 22
Real	Float or Double	Float or double value with different precision. For example: 3.141, -5.332, or 1.343e+17 **Float**: 32 bit. IEEE single precision floating point standard, used when 6-7 significant digits are needed. Default is NAN. **Double**: 64 bit. IEEE double, used when 15-16 significant digits are needed. Default is NAN.
Message	String	String value. For example: "Hello World"
n/a	DateTime	Datetime value. For example: "04/13/2006 04:03:22.222 AM"
n/a	ElapsedTime	Float value that represents a time that has elapsed in seconds. It is shown often in the following format, but is stored as a float value. [–][DDDDDD] [HH:MM:]SS[.fffffff] Values are as follows: • DDDDDD is from 0 to 999999 • HH is from 0 to 23 • MM is from 0 to 59 • SS is from 0 to 59 • fffffff is fractional seconds to right of the decimal Elapsed time can be positive or negative.
n/a	InternationalizedString	A special string data type that can store special characters.

You can configure Industrial Graphics to retrieve data from the Galaxy.

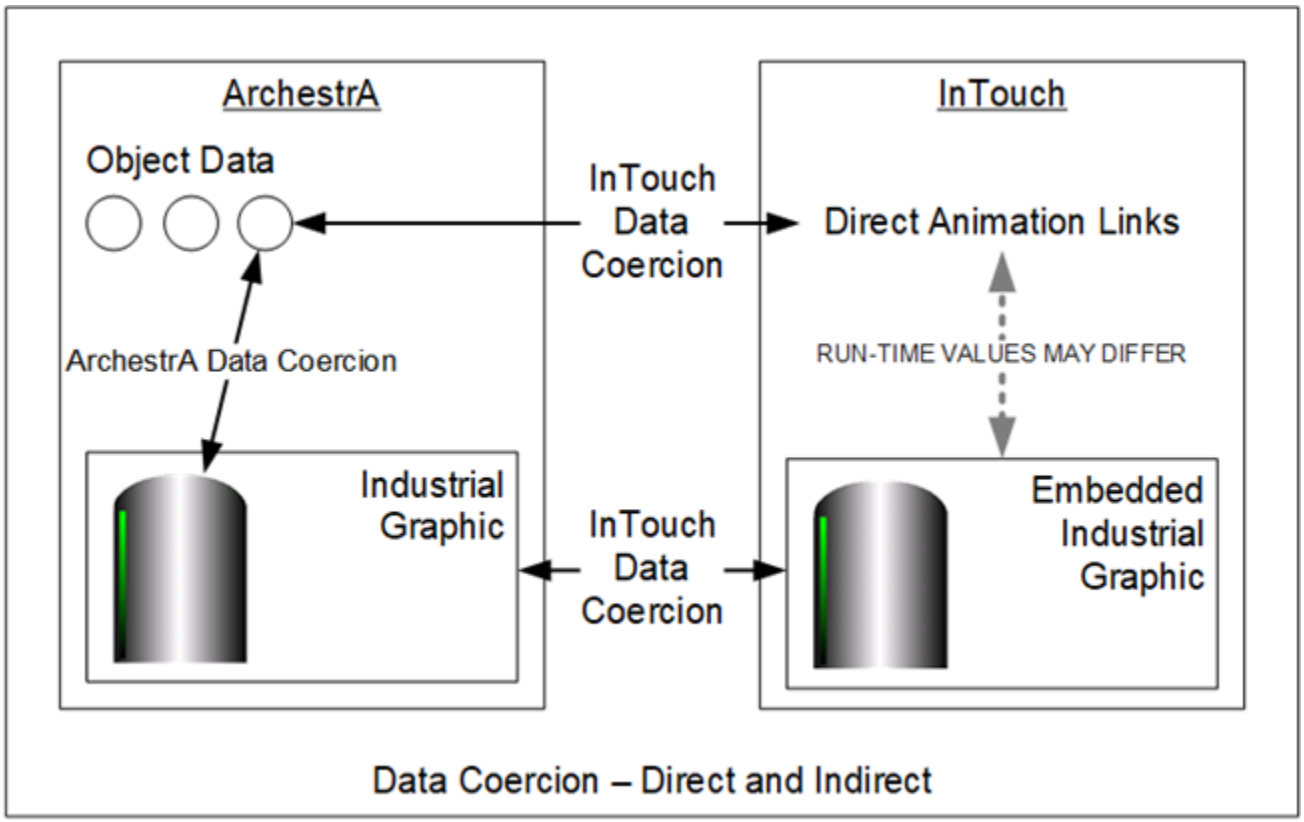
When the source data type is different than the data type it is used for, the data is coerced according to the rules

of IDE data coercion and a string value of "-10" is coerced to "True" in the animation.

If you embed this Industrial graphic into an InTouch window, the data type of the animation link is coerced according to the InTouch data coercion. The embedded Industrial graphic shows "True" in the InTouch HMI.

However, if you directly create an discrete animation display link in the InTouch HMI that points at the original data source, the resulting value can be different.

In this example the string value "-10" is shown as "False" in the InTouch HMI.



Using Animations

You configure InTouch animations using the **Animation Links** dialog box. You can open this dialog box by double-clicking an InTouch object.

You configure animations in the Industrial Graphic Editor using the **Edit Animations** dialog box, which is normally opened by double-clicking an element.

Some of the animation types are different and others have been grouped to simplify configuration. Use the following table to find the equivalent animation type in the Industrial Graphic Editor:

InTouch Animation	Industrial Graphic Editor Animation
User Inputs - Discrete	User Input - Boolean
User Inputs - Analog	User Input - Analog

InTouch Animation	Industrial Graphic Editor Animation
User Inputs - String	User Input - String
Sliders - Vertical	Slider Vertical
Sliders - Horizontal	Slider Horizontal
Touch Pushbuttons - Discrete Value	Pushbutton - Boolean
Action	Action Scripts
Show Window	(not supported)
Hide Window	(not supported)
Line Color - Discrete	Line Style - Boolean
Line Color - Analog	Line Style - Truth Table
Line Color - Discrete Alarm	converted to Line Style
Line Color - Analog Alarm	converted to Line Style
Fill Color - Discrete	Fill Style - Boolean
Fill Color - Analog	Fill Style - Truth Table
Fill Color - Discrete Alarm	converted to Fill Style
Fill Color - Analog Alarm	converted to Fill Style
Text Color - Discrete	Text Style - Boolean
Text Color - Analog	Text Style - Truth Table
Text Color - Discrete Alarm	converted to Text Style
Text Color - Analog Alarm	converted to Text Style
Object Size - Height	Height
Object Size - Width	Width
Location - Vertical	Location Vertical
Location - Horizontal	Location Horizontal
Percent Fill - Vertical	% Fill Vertical
Percent Fill - Horizontal	% Fill Horizontal

InTouch Animation	Industrial Graphic Editor Animation
Miscellaneous - Visibility	Visibility
Miscellaneous - Blink	Blink
Miscellaneous - Orientation	Orientation
Miscellaneous - Disable	Disable
Miscellaneous - Tooltip	Tooltip
Value Display - Discrete	Value Display - Boolean
Value Display - Analog	Value Display - Analog
Value Display - String	Value Display - String

Use scripts

You can configure scripts in Industrial Graphic Editor the same way as you do in InTouch WindowMaker. There are, however, some small differences:

InTouch Script	Industrial Graphic Editor Script
Application Script	(not available)
Window Script	Symbol Predefined Script
Key Script	Action Script animation with a key trigger
Condition Script	Symbol Named Script with an OnTrue, OnFalse, WhileTrue or WhileFalse trigger
Data Change Script	Symbol Named Script with a DataChange trigger
QuickFunction	(not available)
ActiveX Event Script	(not available)
Action Script	Action Script animation

Use application scripts

In the InTouch HMI, application scripts can be triggered:

- One time when the application starts in WindowViewer.

- Periodically when the application runs in WindowViewer.
- One time when the application shuts down in WindowViewer.

Industrial graphics correspond to InTouch applications and enable you to configure predefined scripts directly associated with symbol. Such are:

- On Show
- While Showing
- On Hide

Use key scripts

You cannot use key scripts in the Industrial Graphic Editor, but you can associate an element with an action script that is activated by a key combination.

Use condition scripts

You can configure a script that runs when a condition is fulfilled by using the Symbol Scripts feature. It lets you define triggers that run a script when a value or expression:

- Is fulfilled. (WhileTrue)
- Becomes fulfilled. (OnTrue)
- Is not fulfilled. (WhileFalse)
- Becomes no longer fulfilled. (OnFalse)

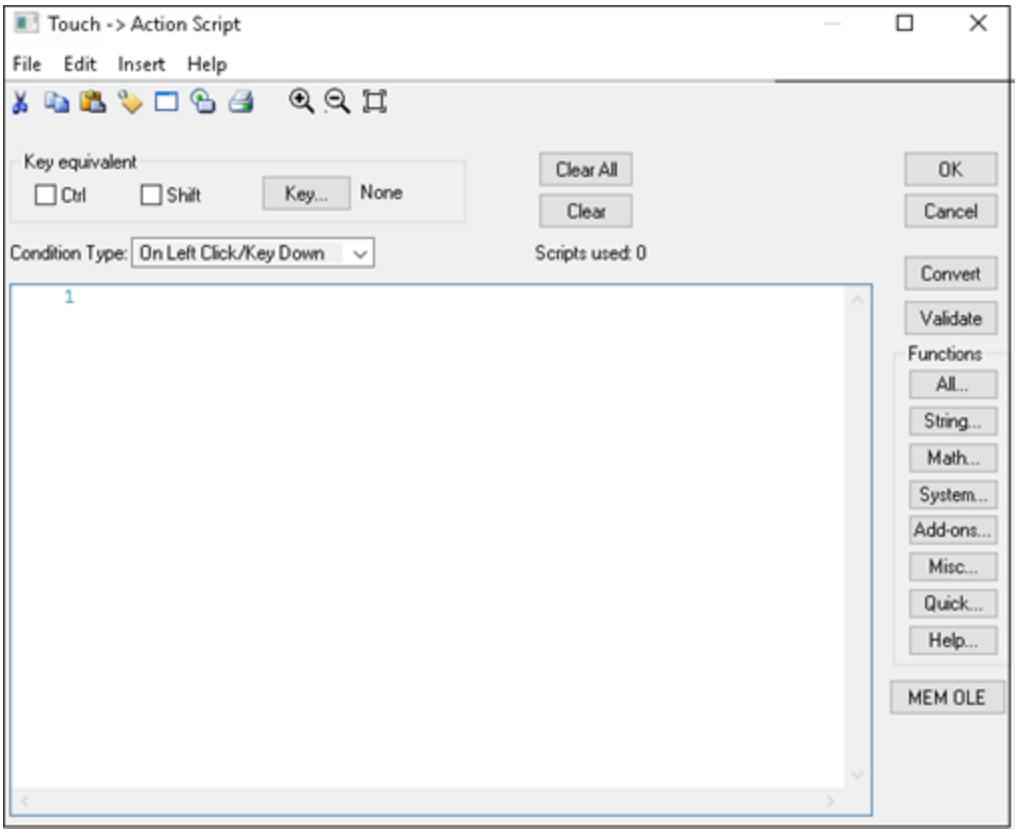
Use data change scripts

You can configure a script that is run when a value or expression changes by using the Symbol Scripts feature. It lets you define a trigger that runs a script when a value or expression changes.

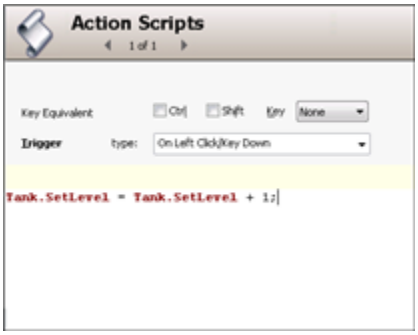
Use action scripts

You can configure Action Scripts in the Industrial Graphic Editor the same way as you would in InTouch WindowMaker. When the run-time user interacts with an element, such as with the mouse or by pressing a key, an action script can run.

You use the InTouch action script window to create action scripts.



You use the Industrial Graphic Editor action script window to create action scripts.



You can configure action scripts for individual elements or for the entire symbol.

You can use many of the predefined functions of InTouch WindowMaker in the Industrial Graphic Editor. For a complete list of InTouch predefined functions that can be used with Industrial Graphics, see Import action scripts.

Other InTouch script types, such as application scripts and key scripts, can be configured with Application Server AutomationObjects.

Connect animations with InTouch tags

You can connect an element animation and appearance to an InTouch tag. The InTouch tag provides values at run time that control the animation and appearance of the element.

You can connect an element animation to an InTouch tag by:

- Configuring a reference with the **intouch:tag** syntax. This syntax is not required for referencing tags in an InTouch HMI stand-alone application using Industrial Graphics.
- Using a custom property and configuring the custom property in the embedded Industrial graphic in InTouch to reference an InTouch tag. Configuring an Application Server attribute reference to the managed InTouchViewApp object that contains the InTouch tags as attributes. The InTouchViewApp object uses the functionality of an InTouchProxy object.
- Configuring an Application Server attribute reference to an InTouchProxy object that contains the InTouch tags as items. This is a special case of configuring an Application Server attribute reference.

  Important: The History Summary data type only works with Application Server object attributes intended for AVEVA OMI ViewApps or InTouch HMI managed applications. Do not attempt to use custom properties assigned the History Summary data type in InTouch HMI Modern applications.

Use the InTouch:Tagname syntax

When you use the **intouch:tagname** syntax, the animation connects to the InTouch tag of the node the graphic is used. There are some restrictions on how you can use this syntax:

- Unlike in Application Server, you cannot use true and false as Boolean values. Use 1 and 0 instead.
- If you want to make a reference to an InTouch SuperTag, use the following syntax instead:

  `attribute("InTouch:SuperTag\Member")`
- The “InTouch: ” prefix is not needed for an InTouch Standalone application with Industrial Graphics. Tag browsing and runtime tag binding will work without the “InTouch:” prefix.
- Application Server custom properties referencing InTouch tags which have hyphens in their names will not work in run time. For example, "InTouch:TAG-1" will not work in run time.

Set the input mode

In some boxes you can enter a value or expression that uses static and/or references to attributes and element properties. Boxes that support both input methods have an **Input Mode** selection icon.

Select:

- **Static Mode input** icon to specify literal static value or expression such as 3.141 or "Test".
- **Reference Mode input** icon to specify a reference to an attribute or element property such as: Tank_001.PV.

  Note: To use static string values with or without references in Reference mode, you can enclose them with double-quotes such as: "Description: "+Tank_001.Desc

When you configure an element to reference one of its own properties in a configuration field or a script, you can just use its property name. For Industrial graphics, there are no self -referencing keywords such as "me." as used for AutomationObjects.

You can, however, use the "me." keyword to reference attributes of the AutomationObject that is hosting the Industrial graphic you are currently configuring.

Connect animations with InTouchViewApp attributes

To be able to browse for InTouch tags, you must first:

- Create a managed InTouch application by deriving an InTouchViewApp template and configuring it in WindowMaker.
- Derive an instance of the InTouchViewApp derived template.

The InTouch tags are represented by attributes of the InTouchViewApp object instance.

Use the galaxy browser InTouch tag browser tab

You can select InTouch tags directly from the Galaxy Browser when configuring a reference requiring an InTouch tag for an Industrial graphic animation or client script.

When invoked from either the animation editor or the script editor, the Galaxy Browser displays an **InTouch Tag Browser** tab in line with the **Attribute Browser** and **Element Browser** tabs.

The **InTouch Tag Browser** tab lists all InTouchViewApp instances and templates for the current Galaxy in the left pane. The right pane displays the InTouch tags for the selected InTouchViewApp. The **DotFields:** list box will display the dotfields associated with the selected tag.

The **Dotfields** list box below the right pane enables you to specify dotfields for the selected tag.

The **InTouch Tag Browser** tab behaves as follows:

- The InTouch Tag Browser functionality is available only from the animation editor or the script editor.
- The Galaxy Browser reads InTouch tags from the Tagname Dictionary.
- The Tagname Dictionary component is installed and available to the Galaxy Browser whether or not InTouch HMI is installed.
- Tags are refreshed only when the Galaxy Browser is closed and reopened.
- All tags remain in memory until you close and reopen the Galaxy Browser.
- If the InTouchViewApp is checked out by the current user, then the Galaxy Browser reads the latest Tagname Dictionary content of that InTouchViewApp.
- If the InTouchViewApp is checked in and is accessed by any user, then the Galaxy Browser always reads the checked-in version of the Tagname Dictionary.
- If the InTouchViewApp is checked out by a user other than the current user, the Galaxy Browser reads the most recently checked in Tagname Dictionary of that InTouchViewApp.
- If you select an InTouchViewApp template, the output reference string syntax is <InTouch:selectedTag>. If you select an InTouchViewApp instance, the output reference string syntax is <SelectedInTouchViewAppInstance.selectedTag>.

Connect animations to InTouch tags

1. Double-click the element.

   The **Edit Animations** dialog box appears.
2. Select the animation from the Animation list.
3. Select the parameter.
4. Select the browse button.

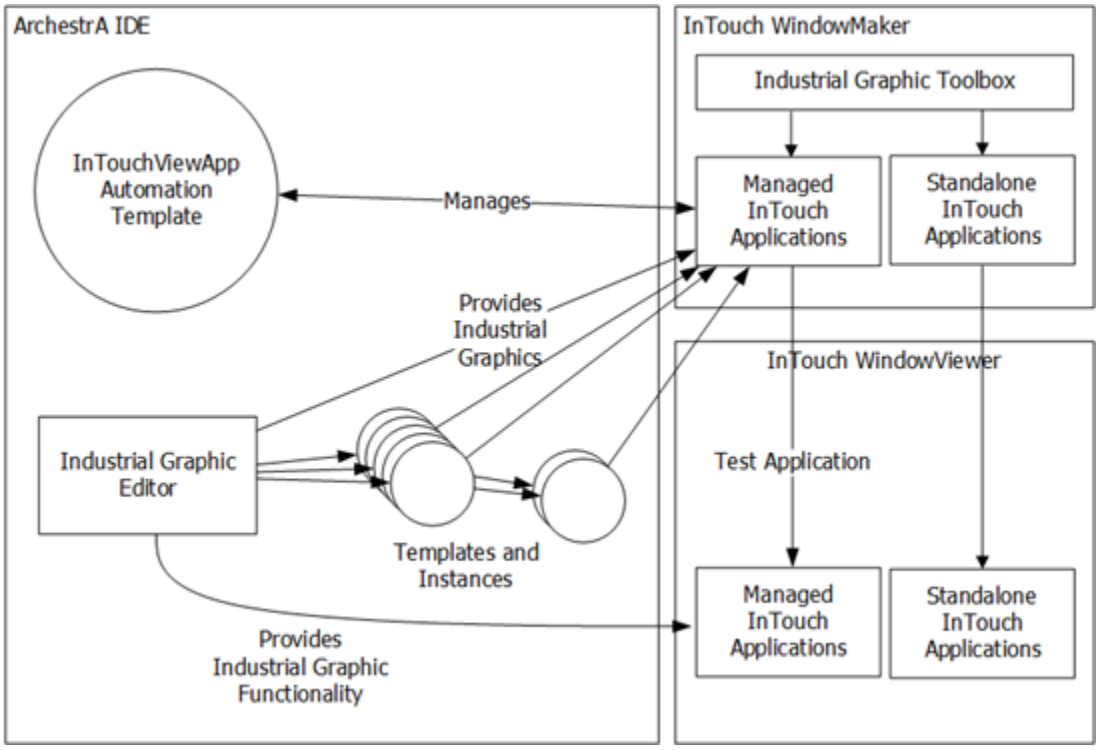
   The **Galaxy Browser** appears.
5. Select the **InTouch Tag Browser** tab to show the **InTouch Tag Browser** page.

6. Select the InTouchViewApp object that corresponds to the managed InTouch application. The right panel shows the InTouch tags.
7. Select a tag and select **OK**.

   The selected reference to an InTouch tag appears in the configuration box.

Use Industrial Graphics in WindowMaker

You can use Industrial Graphics created with the Industrial Graphic Editor in your managed or standalone InTouch applications. You can also add Industrial Graphics directly from WindowMaker's Industrial Graphic Toolbox. For more information about working with Industrial Graphics and Situational Awareness Library symbols, see *Industrial Graphic Editor User* documentation or WindowMaker help.



You can:

- Embed Industrial Graphics into an InTouch window.
- Drag and drop Industrial Graphics directly from the Industrial Graphics Toolbox into an InTouch window.
- Resize embedded Industrial Graphics.
- Add a limited number of InTouch animations to Industrial Graphics.
- Configure the custom properties of embedded Industrial Graphics.
- Start the Industrial Graphic Editor.
- Test the Industrial Graphics in WindowViewer.
- Create a new Instance of the AutomationObject that hosts the embedded Industrial Graphic.

Embed Industrial Graphics into an InTouch window

You can embed an Industrial Graphic into InTouch windows of your managed and standalone InTouch application.

The Industrial Graphic can be part of:

- The Graphic Toolbox.
- An object template.
- An object instance.

The Industrial Graphics that can be embedded include symbols whose elements have Element Styles applied to them and Symbol Wizards created using the Symbol Wizard Editor.

You can publish a managed InTouch application containing Industrial Graphics in the published application. However, you cannot add new Industrial Graphics or edit existing symbols in an application that has been published.

Embed Industrial Graphics from automation templates

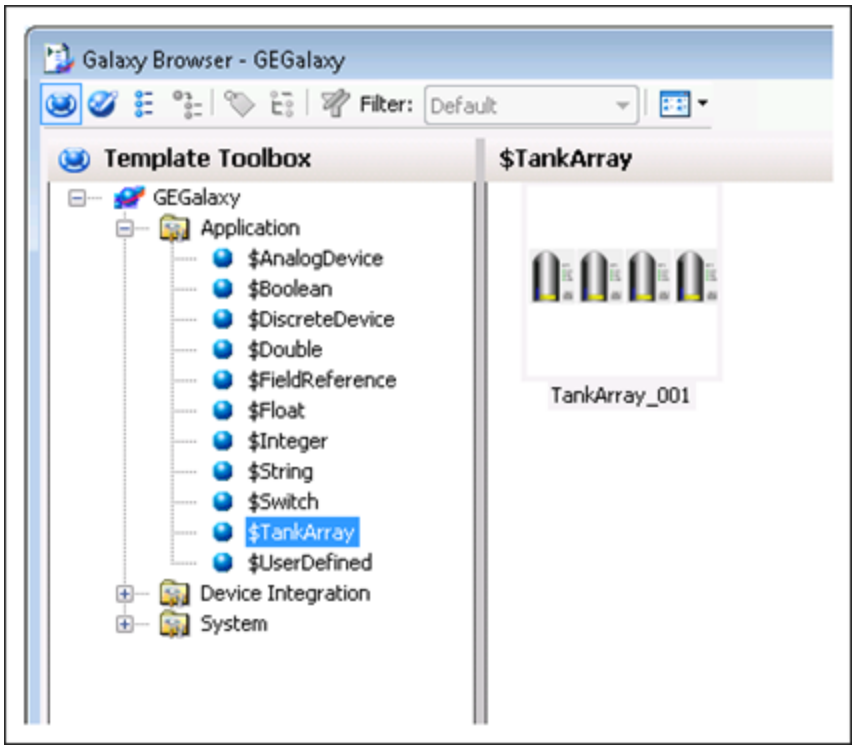
You can embed an Industrial Graphic from an Automation template that hosts Industrial Graphics. At the same time, a new derived instance of the selected template is created.

For information about creating a new instance based on an Industrial Graphic already on the InTouch window, see Test Industrial Graphics in WindowViewer.

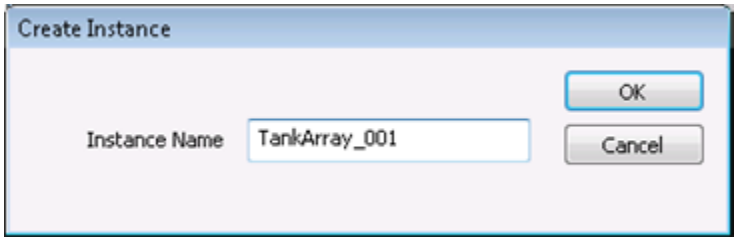
Embed an Industrial Graphic from an Automation template

1. Open WindowMaker.
2. Right-click an object, and select **Embed Industrial Graphic**.

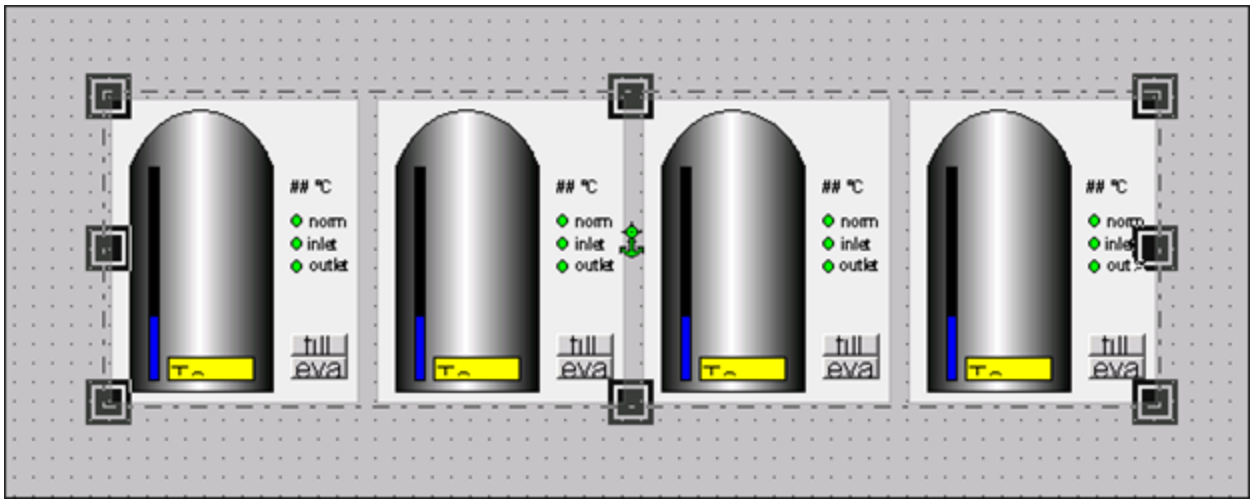
   The **Galaxy Browser** dialog box appears.
3. Select the Template Toolbox icon. The **Template Toolbox** list appears on the left.



4. Select the template that contains the Industrial Graphic you want to embed. The Industrial Graphics contained in the selected template appear on the right.
5. Select the Industrial Graphic you want to embed and select **OK**. The Galaxy Browser closes and the insertion icon appears if the pointer is over the InTouch window.
6. Select in the InTouch window where you want to embed the Industrial Graphic. The **Create Instance** dialog box appears.



7. In the **Instance Name** box, type a name for the instance.
8. Select **OK**. An instance is automatically derived from the template with the name you specify. The symbol is embedded into the InTouch window.



Embed Industrial Graphics from instances

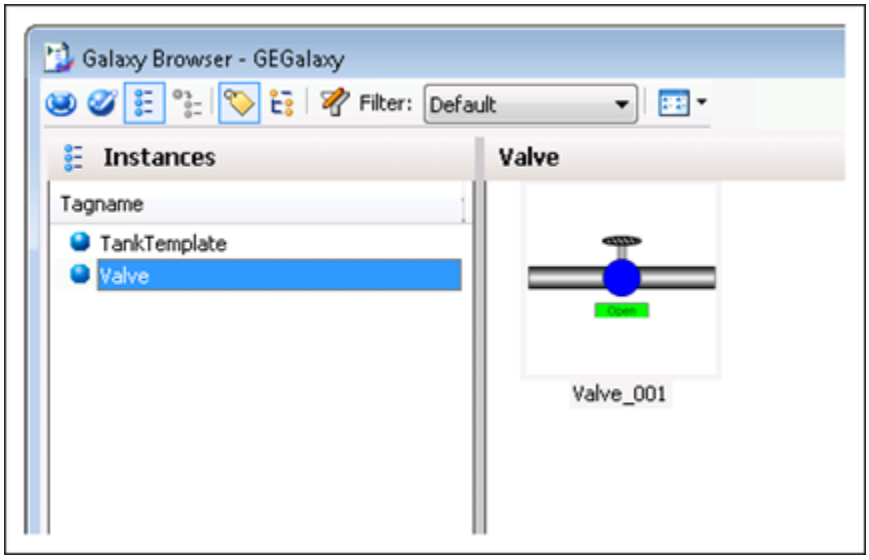
You can embed Industrial Graphics from instances that have Industrial Graphics associated with them.

When you embed an Industrial Graphic from an instance, the symbol is associated with that instance.

Embed an Industrial Graphic from an instance

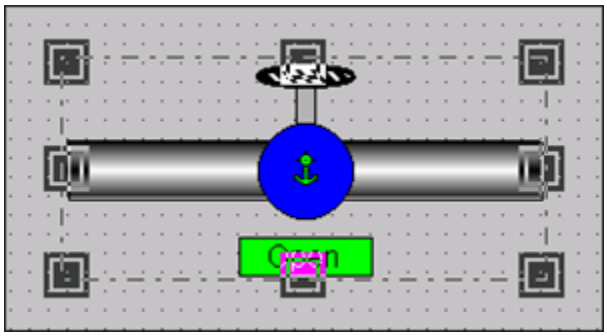
1. Open WindowMaker.
2. Right-click an object, and select **Embed Industrial Graphic**.

   The **Galaxy Browser** dialog box appears.
3. Select the Instances icon. The **Instances** list appears on the left.



4. Select the instance that contains the Industrial Graphic you want to embed. The Industrial Graphics associated with the selected instance appear on the right.
5. Choose the Industrial Graphic you want to embed and select **OK**. The Galaxy Browser closes and the insertion icon appears if the pointer is over the InTouch window.
6. Select the location within the InTouch window to embed the Industrial Graphic. The symbol is embedded

into the InTouch window.



Embed Industrial Graphics from the Graphic Toolbox

You can embed Industrial Graphics from the Graphic Toolbox.

Embed an Industrial Graphic from the Graphic Toolbox

Option 1

1. Open WindowMaker.

   Note: Do not perform any operations related to Industrial Graphics, while Industrial Graphics are loading or if you see the progress bar in the Graphic Toolbox.

2. From the Industrial Graphics pane, select the required Industrial Graphic folder.

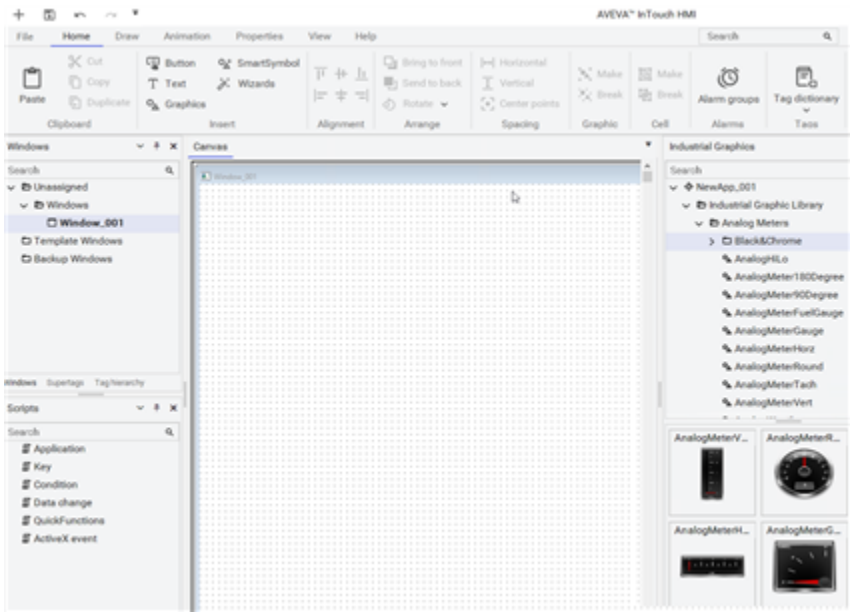
   The Industrial Graphics present under that folder are displayed as thumbnails.

   OR

   Search for the required Industrial Graphic in the Search box.

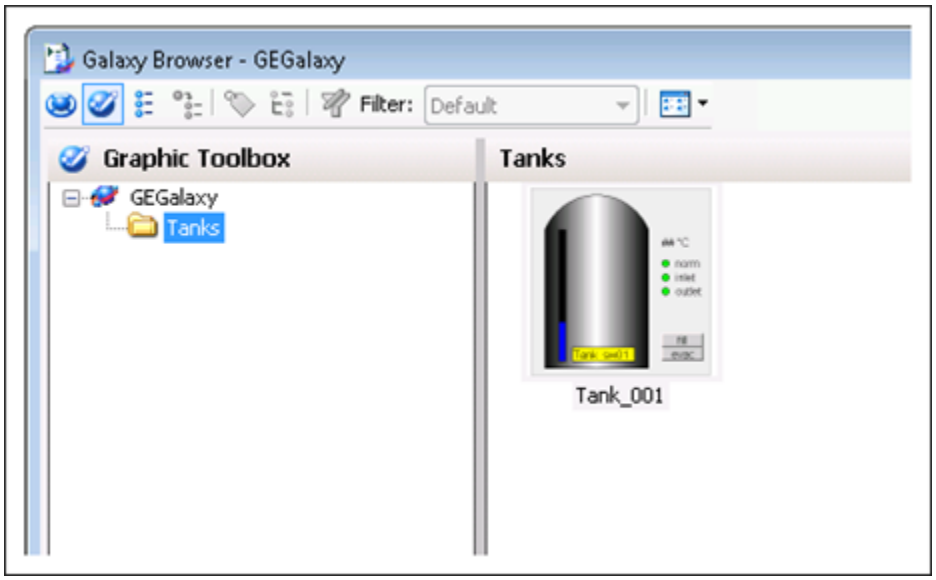
3. Select the required graphic and drag and drop to the canvas.

   A Frame Window will be created.



Option 2

1. Open WindowMaker.
2. Right-click an object and select **Embed Industrial Graphic**.
3. The **Galaxy Browser** dialog box appears.
4. Select the Graphic Toolbox icon  . The **Graphic Toolbox** list appears on the left.



5. Select the Industrial Graphic you want to embed and select **OK**. The insertion icon appears if the pointer is over the InTouch window.
6. Select the location within in the InTouch window to embed the Industrial Graphic. The symbol is embedded into the InTouch window.

Embed Industrial Graphics with element styles applied

You can embed Industrial Graphics that have Element Styles applied to them. An Element Style defines one or more of the fill, line, text, blink, and outline properties of a graphic. The visual properties defined in an Element Style are applied to the graphic. Element Styles make it easy to apply consistent styles to elements. Element Styles also establish visual standards for screen builders and others who create symbols.

An Industrial graphic that has an Element Style applied to it is embedded into an InTouch window just like any other Industrial Graphic.

For more information about using Element Styles with Industrial Graphics, see "Work with element styles" in *Industrial Graphic Editor User* documentation.

Element Styles in Managed InTouch Applications

WindowViewer can run only one application at a time. If a platform is deployed on a local node, the configured styles of the Galaxy take precedence over any configured styles in any other standalone applications.

Embed symbol wizards

You can embed Symbol Wizards into a window of a managed InTouch application from the System Platform IDE.

Embed a Symbol Wizard

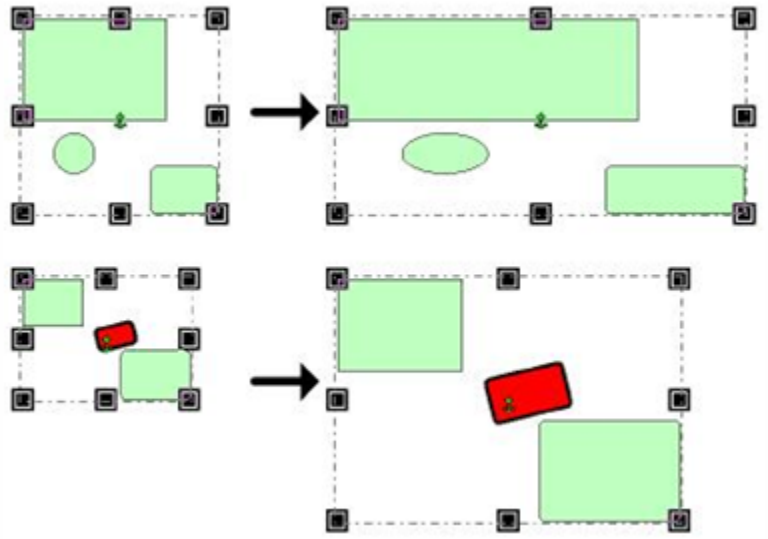
1. Open WindowMaker and show the window that will contain the embedded symbol.
2. Right-click an object, and select **Embed Industrial Graphic**.

   The **Galaxy Browser** dialog box appears.
3. Select the Graphic Toolbox icon  . The **Graphic Toolbox** list appears on the left.
4. Select the Symbol Wizard you want to embed into an application and select **OK**.
5. Select the location within the InTouch window to embed the Symbol Wizard. The symbol is embedded into the InTouch window using its default Symbol Wizard configuration.

Resize embedded Industrial Graphics

After embedding an Industrial Graphic into an InTouch window, you can resize it with its handles or with the width and height value entry as you would with any other InTouch object.

However, if the Industrial Graphic contains at least one rotated element, you can only resize the Industrial Graphic proportionally.



You cannot resize the embedded Industrial Graphic smaller than its minimum size. The minimum size may be determined by the pen width of a contained element.

You can reset the embedded Industrial Graphic to its original size when it was created in the Industrial Graphic Editor.

Resize an embedded Industrial Graphic

1. Select the Industrial Graphic so that the handles appear.
2. Do one of the following:
   - Drag the one of the handles to resize the Industrial Graphic to the new size.
   - Enter width and height in the **W** and **H** boxes on the status bar.

Resize an embedded Industrial Graphic to its original size

- Right-click the embedded Industrial Graphic you want to change to its original size, point to **Industrial Graphic**, and then select **Symbol - Original Size**. The embedded Industrial Graphic is changed to its original size.

Configure Industrial Graphics in WindowMaker

You can configure embedded Industrial Graphics in WindowMaker by:

- Standard editing, such as copying, cutting, pasting, duplicating, resizing, moving, and deleting.
- Setting up WindowMaker animation links.
- Connecting an Industrial Graphic with InTouch tags.
- Selecting an alternate instance of the same parent.
- Selecting an alternate symbol of the same instance.
- Enabling or disabling dynamic size propagation.

Configure WindowMaker animation links of an Industrial Graphic

You configure WindowMaker animation links of an embedded Industrial Graphic in the same way as any other InTouch object. You can only configure animation links that are external to the embedded Industrial Graphic, such as:

- Object size
- Object location
- Visibility
- Enablement

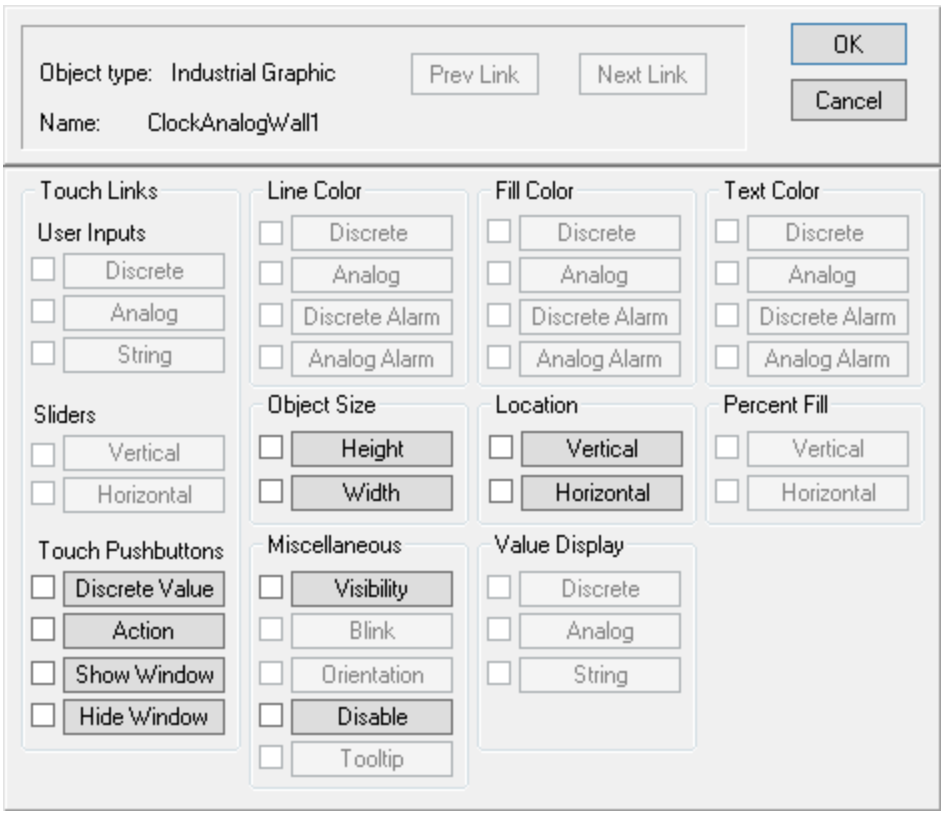
The animation links configured in WindowMaker are independent from those configured in the Industrial Graphic Editor. They do not inherit the settings of the Industrial Graphic and take precedence when run in WindowViewer.

Configure WindowMaker animation links of an embedded Industrial Graphic

1. Select the embedded Industrial Graphic.
2. On the **Draw** menu, in the **Mode** group, select **Animate**.

   Alternatively, right-click the object and then select **Animation Links**.

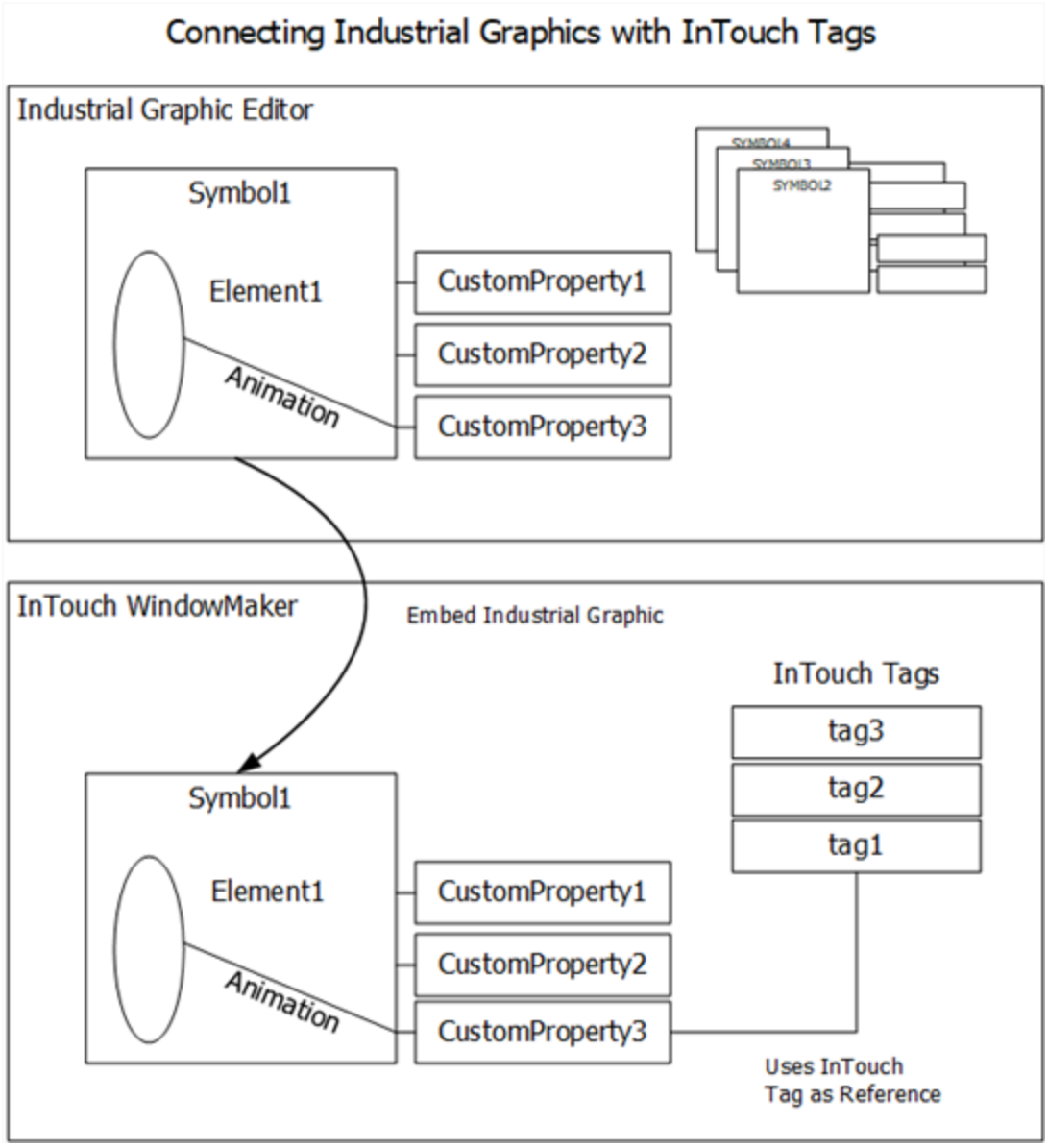
   The **Animation Links** window appears.



3. Make any changes as you would to any other InTouch object.
4. Select **OK**.

Connect Industrial Graphics to InTouch tags

You can connect Industrial Graphics to InTouch tags by overriding the custom properties of an embedded Industrial Graphic. Custom properties expose the properties of an Industrial Graphic to InTouch. Custom properties may or may not be used internally by the animations of the Industrial Graphic.



When you embed an Industrial graphic into an InTouch window, the references in the animation links are converted, as shown in the following table:

Industrial Graphic	Embedded Industrial Graphic
Object.Extension	galaxy:Object.Extension
intouch:Tagname	Tagname

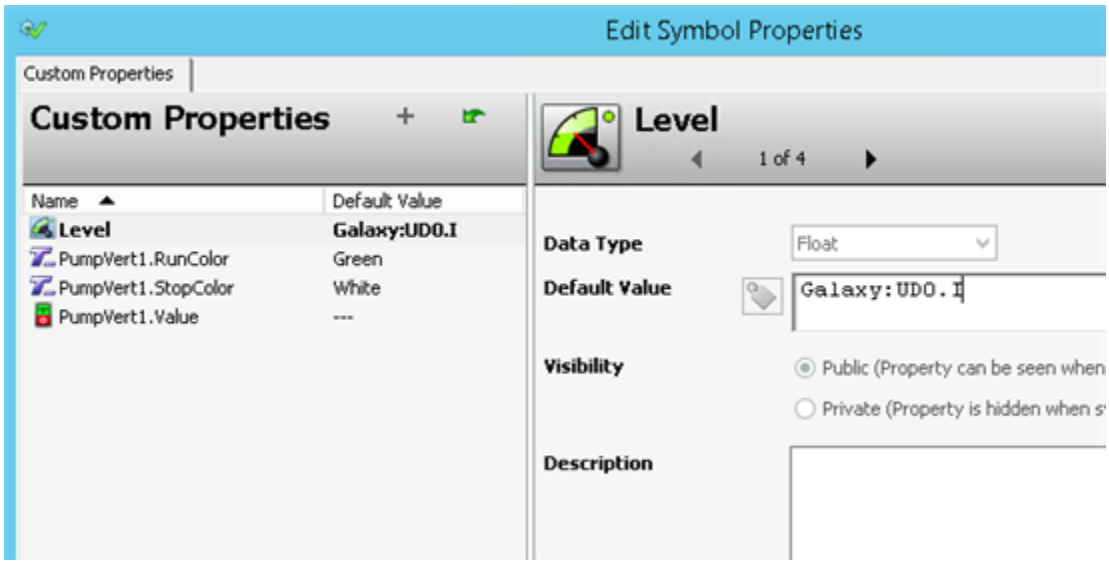
For more information about custom properties, see *Industrial Graphic Editor User* documentation.

You can select InTouch tags directly from the Galaxy Browser when working within the animation editor or the script editor. When invoked from either of those editors, the Galaxy Browser lists all InTouchViewApp instances for the current Galaxy, and lists all InTouch tags and dot fields for the selected InTouchViewApp instance.

For more information about using the Galaxy Browser to select InTouch tags, see Browse Application Server object attributes from InTouch.

Connect an Industrial Graphic to an InTouch tag

1. Right-click the embedded Industrial graphic in the InTouch window, point to **Industrial Graphic**, and then select **Edit Symbol Properties**. The **Edit Symbol Properties** dialog box appears.



2. Select the custom property you want to connect to an InTouch tag. The configuration for the selected custom property appears in the right pane.
3. In the **Default Value** box, do one of the following:
   - Type the name of the InTouch tag.
   - Select the browse button and select a tag from the **Select Tag** dialog box.

4. To restore the original value of the custom property, select **Restore**.
5. Select **OK**. Any animation in the Industrial Graphic that is configured with the selected custom property now uses the InTouch tag value during processing.

Example of connecting Industrial Graphics to InTouch tags

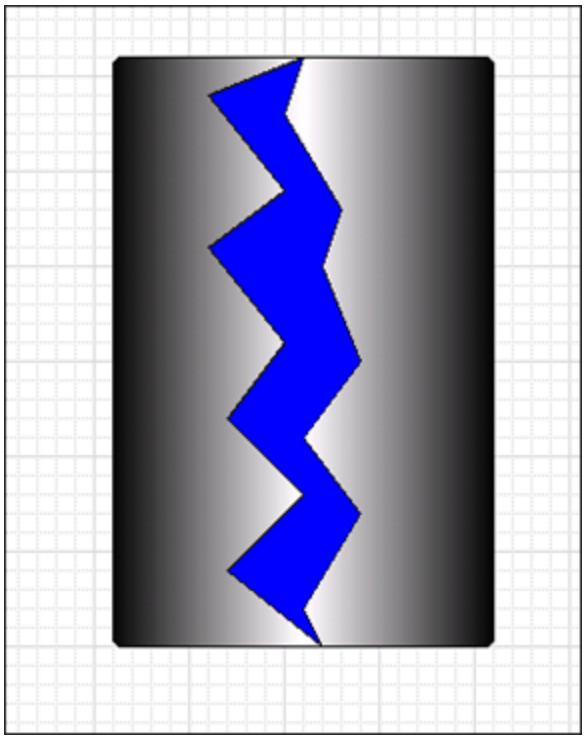
This example shows you how to connect a tank symbol with a percent vertical fill animation created by the Industrial Graphic Editor to an InTouch tag.

You do this in three main steps:

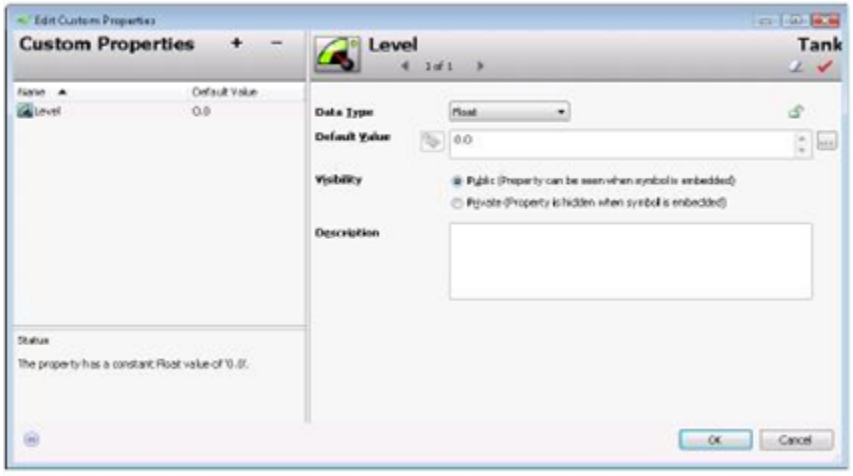
- Create a sample Tank as an Industrial Graphic.
- Create the InTouch application.
- Derive and view the sample tank in WindowViewer.

Create a sample tank as Industrial graphic

1. In the IDE, create a new symbol called "Tank" and open it in the Industrial Graphic Editor.
2. Paste a rectangle on the canvas. Change its appearance as needed.
3. Create a colored polygon element that represents a cutout of the tank to show the tank level.

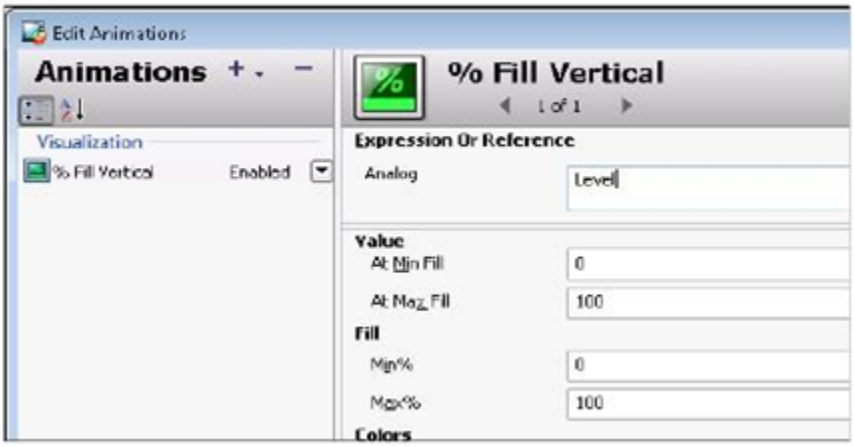


4. Select the canvas.
5. On the **Properties** menu, in the **Graphic** group, select **Edit**.
   Alternatively, double-click the graphic object.
   The **Edit Custom Properties** dialog box appears.
6. Add a custom property called **Level**.
7. Configure the property details. Do the following:
   - In the **Data Type** list, select **Float**.
   - In the **Default Value** box, type 0.



8. Select **OK**.
9. Double-click on the polygon element that represents the tank level.
   The **Edit Animations** dialog box appears.

10. Add a **% Vertical Fill** animation.
11. In the **Analog** box in the right pane, type the name of the custom property. In this example, it is Level.



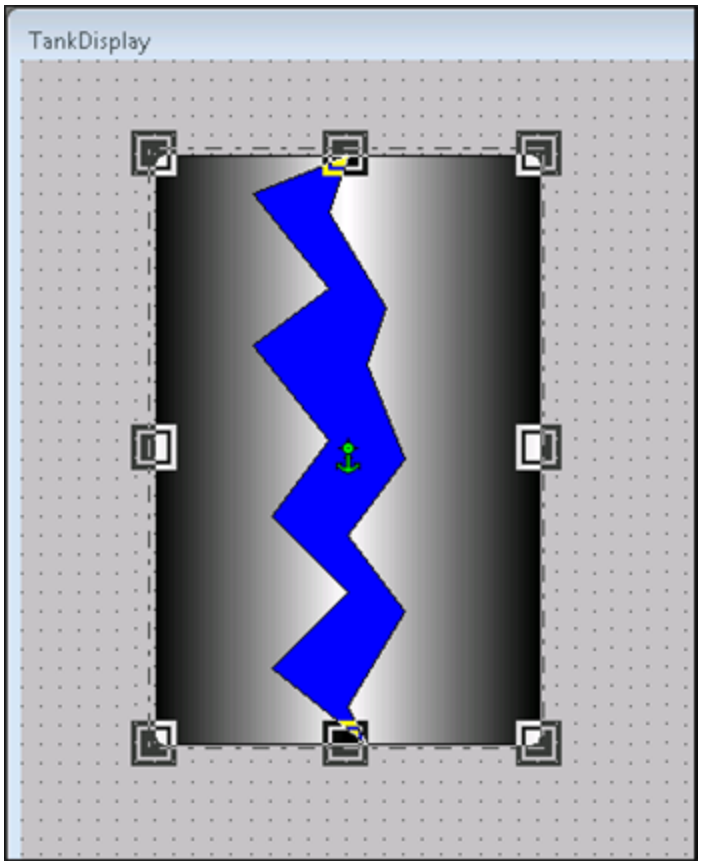
12. Select **OK** to close the **Edit Animations** dialog box.
13. Select **Close and Save** to close the Industrial Graphic Editor.

Create the InTouch application

1. In the System Platform IDE, create a new managed InTouch application. For more information, see Create a managed InTouch application.
2. Open the managed InTouch application in WindowMaker.
3. Create a new window called TankDisplay.
4. Open the Tagname Dictionary and create a new real InTouch tag named TankLevel.
5. Select the Embed Industrial Graphic icon.

   The **Galaxy Browser** dialog box appears.
6. Select the Tank symbol and select **OK**.
7. Select the new location within the window to embed the symbol.

   The Tank symbol is embedded into the window.



8. Right-click the embedded Industrial Graphic, point to **Industrial Graphic "Tank"**, and then select **Edit Symbol Properties**.

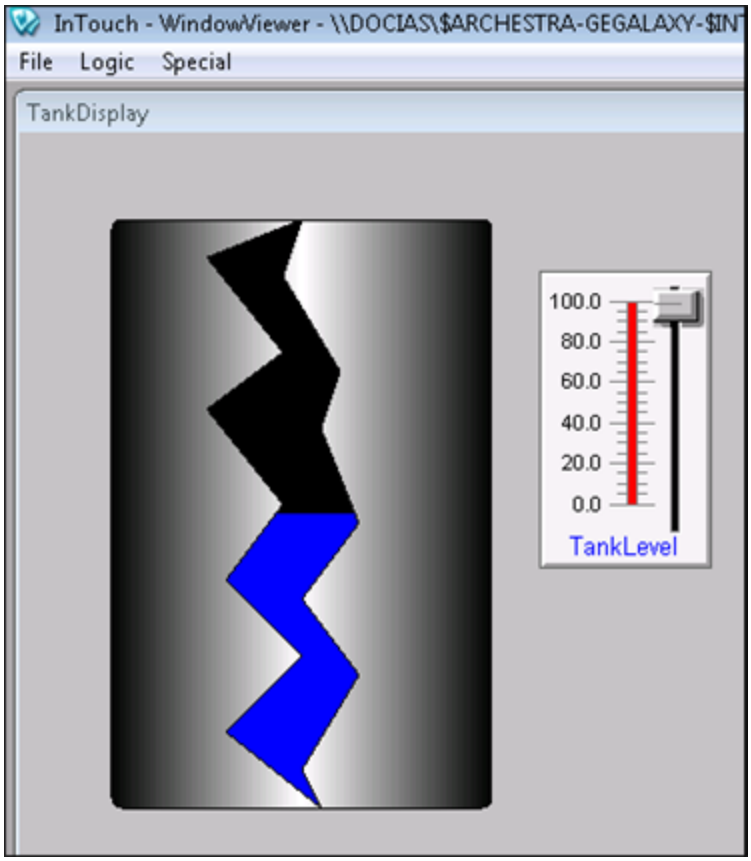
   The **Edit Symbol Properties** dialog box appears.
9. Select the custom property Level.
10. In the **Default Value** box, type `TankLevel`. You can also select the ellipsis button to browse for TankLevel using the **Select Tag** dialog box.
11. Select **OK**.
12. Paste a slider on the window and configure it with the local InTouch tag TankLevel.
13. Save the changes and close WindowMaker.

    The managed InTouch application is automatically checked in.

Derive and test the sample tank

1. In the System Platform IDE, derive an instance of the managed InTouch application and deploy it together with a WinPlatform and a ViewEngine instance.
2. Open the InTouch Application Manager and start the listed application in WindowViewer.

   The tank and the slider appear on the window in WindowViewer.
3. You can move the slider to change the tank level.



Select alternate instances from the same parent

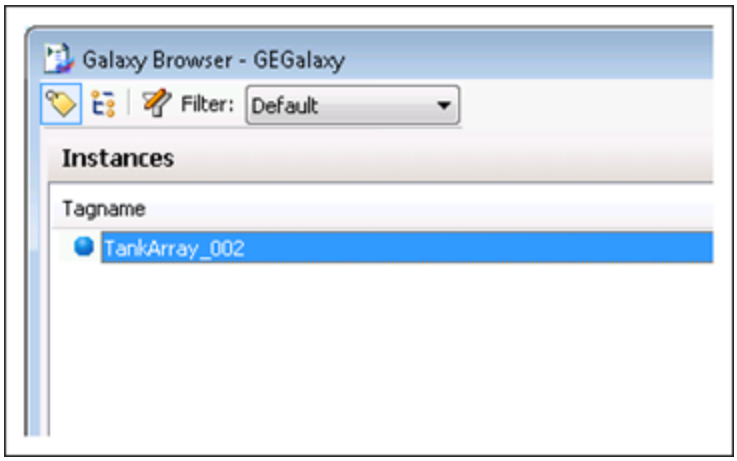
You can redirect all references in the Industrial Graphic to an alternate instance. The appearance of the Industrial Graphic does not change, except possibly for its size, as it is not possible to edit inherited Industrial Graphics.

You cannot use this function with Industrial Graphics that originate from the Graphic Toolbox, as they are not associated with any object.

Select an alternate instance from the same parent

1. Right-click the embedded Industrial Graphic, point to **Industrial Graphic**, and then choose **Select Alternate Instance**.

   The **Galaxy Browser** dialog box appears. It shows all other instances that have the same parent.



2. Select an alternate instance from the list and select **OK**.

   The references of the Industrial Graphic are updated to point at the new alternate instance.

Select alternate symbols of the same instance

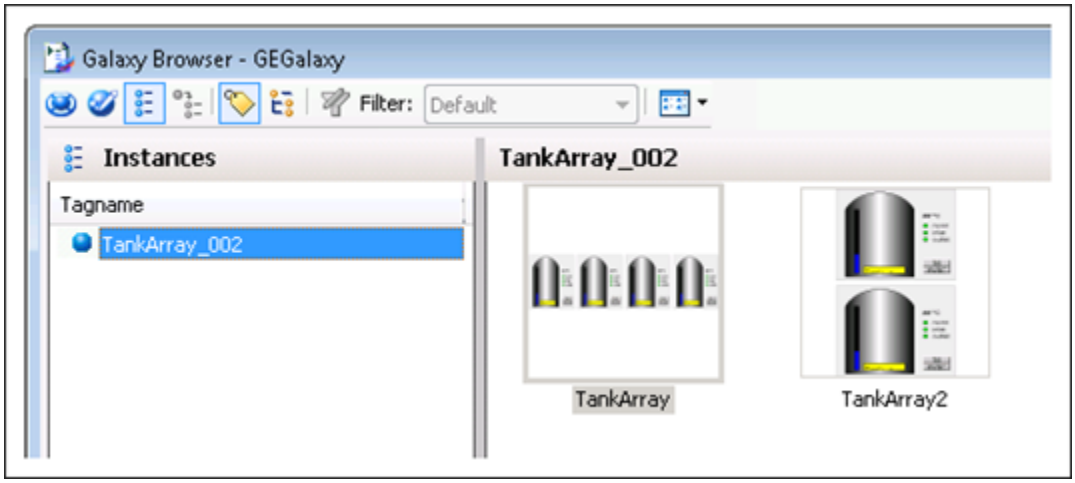
You can replace an embedded Industrial Graphic with another Industrial Graphic belonging to the same instance.

You cannot use this function with Industrial Graphics that originate from the Graphic Toolbox, as they are not associated with any object.

Select an alternate Industrial Graphic from the same instance

1. Right-click the embedded Industrial Graphic, point to **Industrial Graphic**, and then choose **Select Alternate Symbol**.

   The **Galaxy Browser** dialog box appears.



2. Highlight an alternate symbol in the right pane and select **OK**.
3. If the alternate symbol is a different size than the original symbol, a message appears prompting if you want to keep the size of the currently embedded Industrial Graphic. Do any of the following:
   - Select **Yes** to keep the current size of the selected Industrial Graphic.
   - Select **No** to update the size of the selected Industrial Graphic to the size of the new Industrial Graphic.

   In both cases, the embedded Industrial Graphic is updated with the new alternate Industrial Graphic.

Substitute strings in Industrial Graphics

You can substitute all strings in an embedded Industrial Graphic with alternate strings.

Substitute all strings in an embedded Industrial Graphic

1. Select the embedded Industrial Graphic.
2. On the **Animation** menu, in the **Substitute** group, select **Strings**.
   The **Substitute Strings** dialog box appears.
3. Type new strings in the corresponding boxes and select **OK**.
   The strings in the embedded Industrial Graphic are substituted by the new alternate strings.

Substitute references in Industrial Graphics

You can substitute all references in an embedded Industrial Graphic with alternate references.

Substitute all references in an embedded Industrial Graphic

1. Select the embedded Industrial Graphic.
2. On the **Animation** menu, in the **Substitute** group, select **Tags**.
   The **Substitute Tags** dialog box appears.
3. Type new references in the corresponding boxes and select **OK**.
   The references in the embedded Industrial Graphic are substituted by the new alternate references.

Enable or disable dynamic size change propagation of embedded Industrial Graphics

You can enable or disable the dynamic size change propagation of embedded Industrial Graphics.

If the dynamic size change propagation is enabled, any change to the absolute anchor point position of the source symbol:

- Leaves the anchor points of its embedded symbols unchanged.
- Moves the embedded symbol position accordingly.

If the dynamic size change propagation is disabled, any change to the absolute anchor point position of the source symbol:

- Moves the anchor points of its embedded symbols accordingly.
- Leaves the embedded symbol position unchanged.

For more information about dynamic size propagation, see *Industrial Graphic Editor User* documentation.

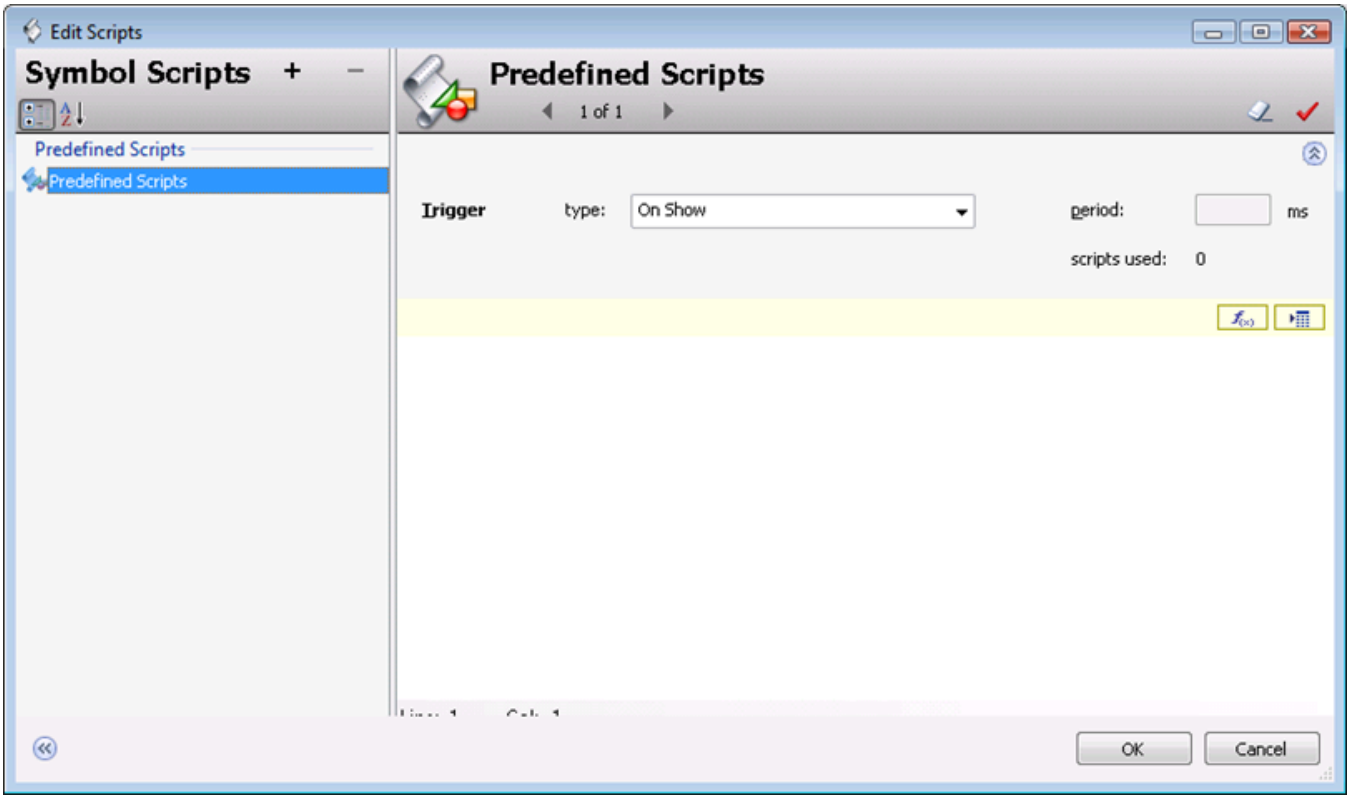
Enable or disable dynamic size change propagation of an embedded symbol

- Right-click the embedded Industrial Graphic, point to **Industrial Graphic**, and then check or uncheck **Dynamic Size Change**.

Associate scripts with Industrial Graphics

You can associate scripts with the Industrial Graphics placed in your InTouch applications. Scripts will animate your graphics or modify their elements while an InTouch application is running.

After selecting an industrial graphic embedded in an InTouch window, you select the expression or reference whose value triggers the script to run. You use the Industrial Graphic Editor to select the trigger that runs the script.

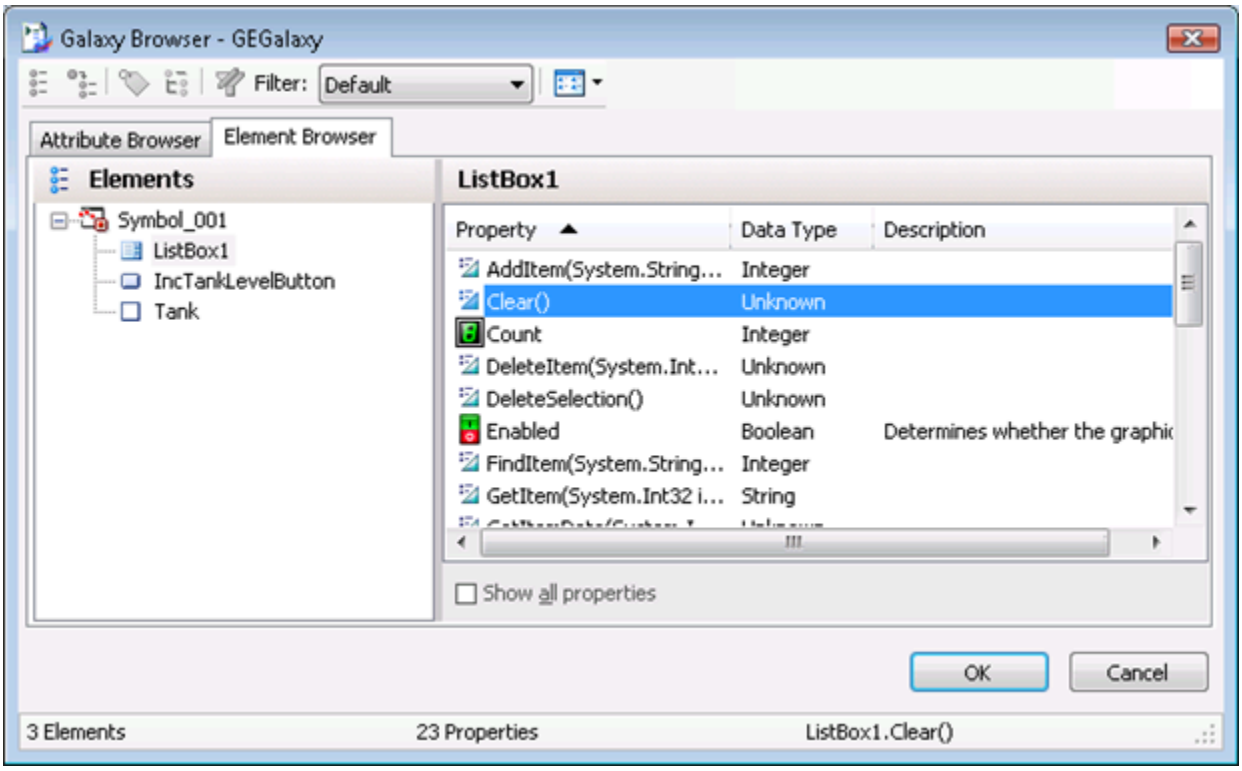


- A graphic script can be either predefined or named. A predefined script runs based on the status of the graphic in the running application. A named script runs when an expression or references associated with the script changes state.
- Predefined scripts are similar to InTouch HMI window scripts. Based upon how you configure the script trigger, a predefined symbol script can run:
  - Once after the graphic opens or is shown.
  - Periodically while the graphic appears in the running application.
  - Once after the graphic closes or is hidden.
- Based upon how you configured the script, a named graphic script can run when the trigger values or expressions are true, false, or transitioning between true and false states. Also, a named graphic script can run when data associated with the trigger expression changes value or its quality state changes value.

Use methods in Industrial Graphic scripts

Some elements support script methods. These methods can perform various functions on the elements themselves at run time. Typically, you configure an action script to access these methods.

You can see the supported properties and methods of any element by opening the Galaxy Browser and selecting the element.



- You can run a script containing an Edit Box control method to load text from a file to the control during run time. You can also run a script to save the current contents of the Edit Box control to a file during run time.
- Edit Box control methods are declared in scripts in the following form:

  ```
  ControlName.SaveText(FileName);
  ```

  where *ControlName* is the name of the Edit box control and *FileName* is the name of the file containing the contents of the control to be loaded or saved. In the example above, SaveText is the name of the method to save the contents of the Edit Box control to a file.
- You can use a script containing the methods of the Combo Box and List Box controls to change the contents of their lists during run time. List items can be added, deleted, or modified.
- Combo Box and List Box control methods are declared in scripts similar to the Edit Box control.

Edit Industrial Graphics in the Industrial Graphic Editor

You can edit embedded Industrial Graphics using the Industrial Graphic Editor that is integrated in the System Platform IDE. You do this by:

1. Opening the embedded Industrial Graphic in Industrial Graphic Editor, modifying the symbol, and then saving it. The Industrial Graphic is updated in the template, instance or in the Graphic Toolbox.

   For more information, see Edit an embedded Industrial Graphic.
2. Accepting the changes in WindowMaker by selecting the Symbol Changed icon in the right bottom corner of the status bar. The changes are then propagated to WindowMaker.

   For more information, see Accept symbol changes in WindowMaker.

Edit an embedded Industrial Graphic

You can easily edit an embedded Industrial Graphic from within InTouch WindowMaker.

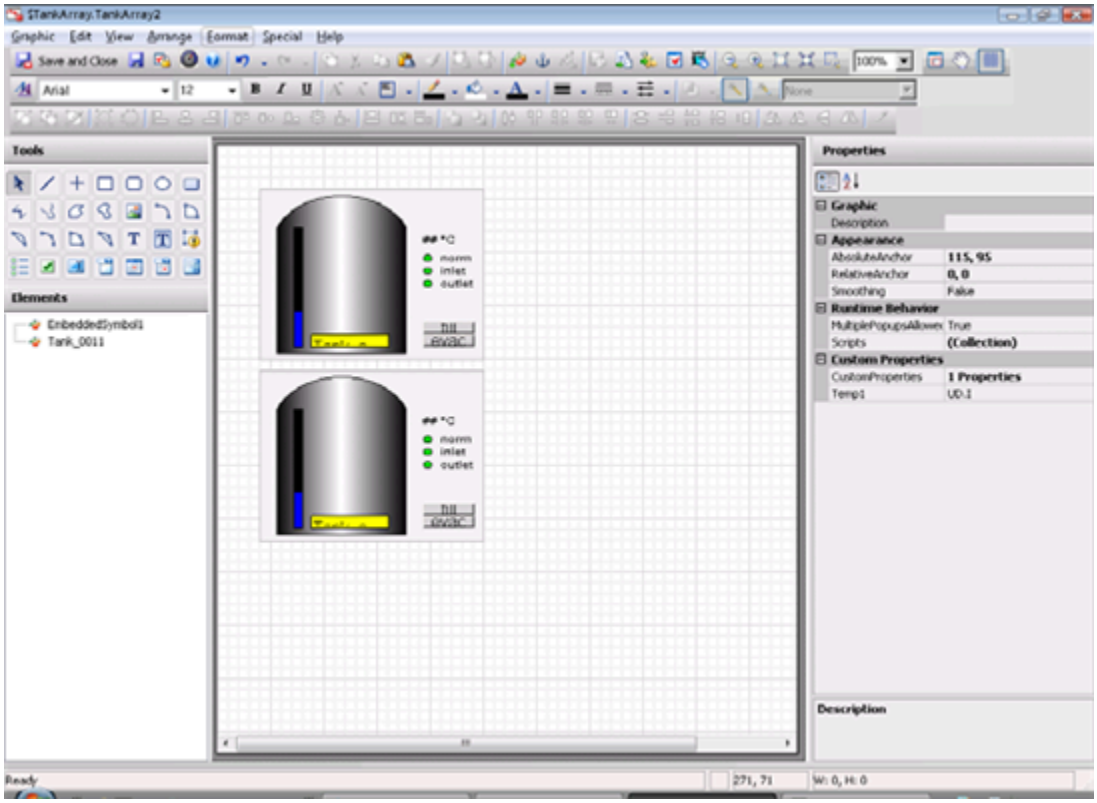
If the source symbol or its embedded symbols are used by other managed InTouch applications, the changes are propagated to the embedded symbols and the InTouch applications.

Any changes you make to the Industrial Graphic are not propagated to the embedded Industrial Graphics automatically. For more information, see Accept symbol changes in WindowMaker.

Edit embedded Industrial Graphics using the Industrial Graphic Editor

1. Right-click the embedded Industrial Graphic, point to **Industrial Graphic**, and then select **Edit Symbol**.

   The **Industrial Graphic Editor** with the Industrial Graphic appears.



2. Edit the Industrial Graphic. For more information, see Create and manage Industrial Graphics User documentation.

   Editing an Industrial Graphic includes applying an Element Style to the symbol. For more information about using Element Styles, see *Application Server User* documentation.
3. Select **Close and Save**. The changes are saved and the Industrial Graphic Editor closes.
4. If the Application Server object is hosted by an instance or template, save and close the object editor in the IDE.

Accept symbol changes in WindowMaker

If an Industrial Graphic changes and you are currently using it in an InTouch window in WindowMaker, you can immediately accept the change in WindowMaker.

If you do not accept the change immediately, the symbol is updated in WindowMaker when the window is closed and opened again.

The symbol is also updated if you switch to WindowViewer to test the application or if you open the application in WindowViewer on a target node.

Immediately accept symbol changes in WindowMaker

- For opened InTouch windows that contain embedded Industrial Graphics, do one of the following:
  - Double-click the Symbol Changed icon in the bottom right corner of the status bar.
  - Close the InTouch window containing the embedded Industrial Graphic and open it again.

  In both cases, the changes made to the Industrial Graphic are reflected in the embedded symbol in the InTouch window.

Accept symbol changes in WindowViewer

If an Industrial Graphic changes and you are currently testing it in WindowViewer, you can accept the change in WindowViewer.

For more information about testing embedded Industrial Graphics, see Test Industrial Graphics in WindowViewer.

Accept symbol changes in WindowViewer when testing

Do one of the following:

- Fast-switch to WindowMaker and then back to WindowViewer.
- Close the InTouch window and open it again. This only works if the **Always load windows from disk** option is checked in the **WindowViewer Properties** dialog box.

In both cases, the changes made to the Industrial Graphic are reflected in the embedded Industrial Graphic in the InTouch window.

Create Graphic Elements and Industrial Graphics using InTouch tags

The Tags tab present in the Properties configuration pane of the Industrial Graphic Editor displays all the tags available in the InTouch application. You can simply drag and drop the tags to the canvas to create graphic elements or Industrial Graphics. When creating multiple symbols at a time, the dot field property will be bounded automatically. This method simplifies the graphic development workflow and significantly reduces the application development time.

Note: This **Tags** tab not available in Industrial Graphic Editor in CONNECT and Managed InTouch.

Create a Graphic Element using InTouch tag

1. In the Industrial Graphic Editor, from the **Tags** pane, drag and drop the tag to the canvas.

   Graphic Elements and **Industrial Graphics** options are displayed.

   Note: You can drag and drop multiple tags together to create multiple graphic elements of same element type and animation.
2. To set the default values
   - Select anywhere on the screen.

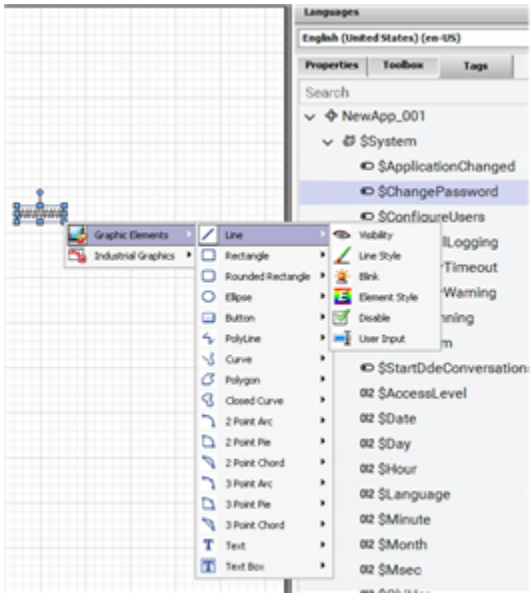
OR

- Hover over the **Graphic Elements** option to select the required graphic element and animation.

  A list of graphic elements and applicable graphic animation by data type are displayed.

3. Hover over the required graphic element.

   Available animations are displayed.



4. Select the required animation.

   A new graphic element associated with the tag created.

Create an Industrial Graphic using InTouch tag

1. In the Industrial Graphic Editor, from the **Tags** pane, drag and drop the tag to the canvas.

   Graphic Elements and **Industrial Graphics** options are displayed.

   Note: You can drag and drop multiple tags together to create multiple Industrial Graphics of same type.

2. To set the default values

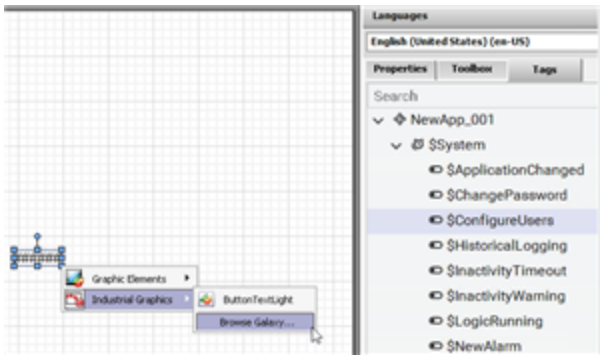
   - Select anywhere on the screen.

     OR

   - Hover over the **Industrial Graphics** option to select the required Industrial Graphic.

     Browse Galaxy option is displayed.

     Note: If you drag same type of tags multiple times and hover over **Industrial Graphics** option, five previously selected graphic names are displayed along with the **Browse Galaxy** option.



3. Select anyone of the previously selected graphic or select **Browse Galaxy** to select a new graphic.
4. If you select the **Browse Galaxy** option, browse and select the required Industrial Graphic and select **OK**.

   A new Industrial Graphic associated with the tag is created.

Note: In the **Custom Properties** of the Industrial Graphic that you have selected if the **Visibility** is **Public** and:

- if the **Default Value** of a **Value** property is empty or ---, then it will be replaced with the name of the tag that you have dragged and dropped.

- if the **Default Value** of the dot properties are empty or ---, then it will be replaced by **tag name.<respective dot property name>**.

In both the above cases, in the Custom Properties of the Industrial Graphic that you have selected, if the **Visibility** of a property is **Private**, then that property will not be visible in the Industrial Graphic that you have created. As the visibility is Private and it should not be visible to other graphic, the custom properties that has **Visibility** as **Private** will not be visible in the other Industrial Graphic.

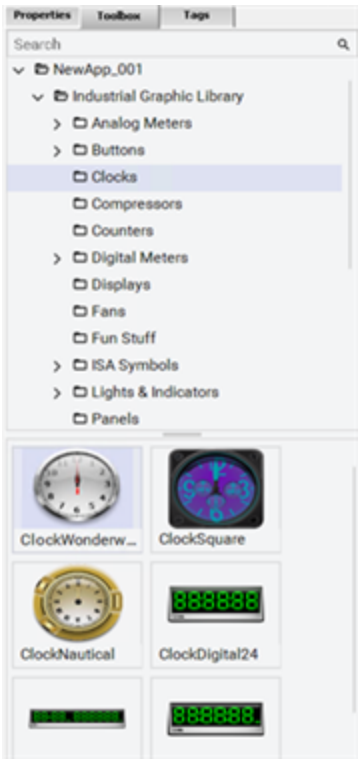
Embed Graphic from the Toolbox tab of Industrial Graphic Editor

You can embed an existing Industrial Graphic into another graphic using the **Toolbox** tab located in the **Properties** configuration pane of the Industrial Graphic Editor. The **Toolbox** tab displays all the Industrial Graphics available in the InTouch library. Once you have embedded the graphic in the other graphic, you can then edit it like any other component of the graphic. You can also embed graphics using Embed Industrial Graphic icon in Industrial Graphic Editor. For more information refer to the Embedding Graphics topic of the Industrial Graphic Editor help.

Embed an existing graphic to another graphic from the Toolbox tab of Industrial Graphic Editor

1. In the **Properties** configuration pane, select the **Toolbox** tab.
2. Navigate through the folders and select the required Industrial Graphic or search for the graphic in the **Search** box.

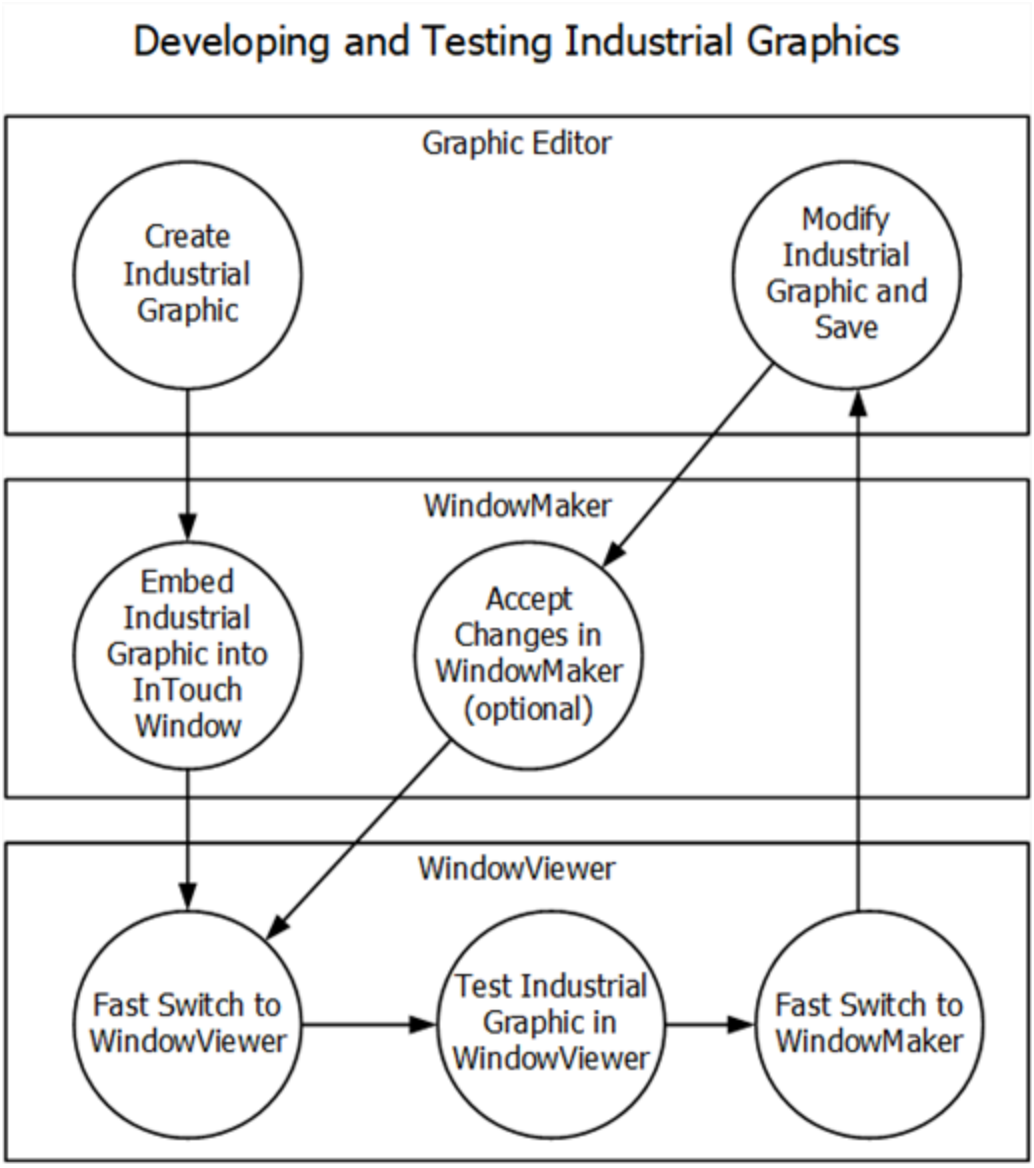
   The graphics in the folder or object you select appear below the navigation area.
3. Select the graphic you want to add and drag it to the desired location on the canvas.



Test Industrial Graphics in WindowViewer

You can test the embedded Industrial Graphic in an InTouch window without having to derive an InTouchViewApp instance. You can test an embedded Industrial Graphic if you previously:

- Created an Industrial Graphic in the Graphic Toolbox, in an automation template or automation instance.
- Created a managed InTouch application.
- Embedded the Industrial Graphic in the managed InTouch application.



Test embedded Industrial Graphics in WindowViewer

1. In WindowMaker, select **Runtime** to switch to WindowViewer.
2. Test the animations, behavior, and appearance of the embedded symbol as you would in a normal run-time environment.
3. You can fast-switch back to WindowMaker to make changes to the way the Industrial Graphic is embedded.

Change and test embedded Industrial Graphics in WindowViewer

1. Make changes to the Industrial Graphic in the Industrial Graphic Editor.
2. Save the changes.

   If WindowViewer is open, then after a short while, a message appears in WindowViewer prompting you to accept the changes. Select **Yes**.

   If WindowViewer is closed, then you can fast-switch from WindowMaker to WindowViewer to see your

changes. It is faster to close WindowViewer and to re-open WindowViewer than to wait for the changes to propagate to the open WindowViewer session.

Estimate graphic performance

You can gauge the performance of an Industrial Graphic at run time using the Graphics Performance Index (GPI).

This tool calculates the estimated call up time when the symbol you are building in the Industrial Graphic Editor is launched at run time. Call up time pertains to the interval between the user or system requesting the pertinent graphic to appear and the graphic appearing on the screen with live data. The calculation is based on contents of the symbol launched in the InTouch WindowViewer at run time, and on a best case time estimate with completed external reference subscriptions.

For more information about using GPI, see "Estimate graphic performance" in *Industrial Graphic Editor User* documentation.

Create new automation instances

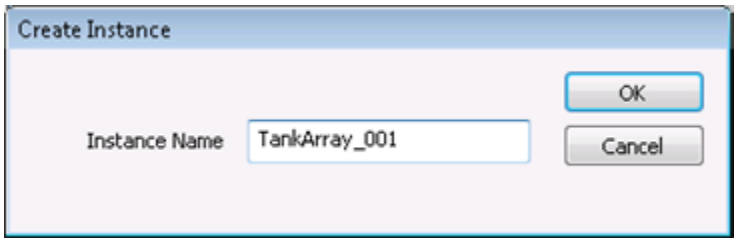
You can quickly create a new instance of the AutomationObject that hosts an embedded Industrial Graphic. This way, you do not need to switch to the System Platform IDE and derive an instance.

The new instance is unassigned, so you need to assign and deploy it in the System Platform IDE before you can use it.

You can only create new Automation instances for Industrial Graphics hosted by templates or instances. Industrial Graphics in the Graphic Toolbox do not have this functionality.

Create a new Automation instance

1. Right-click the embedded Industrial Graphic, point to **Industrial Graphic**, and then select **New Instance**. The **Create Instance** box dialog box appears.



2. In the **Instance Name** box, type a name for the instance.
3. Select **OK**. An instance is automatically derived from the template with the name you specify.

Create new Application Server object instances automatically

If you embed an Industrial Graphic from a template, the InTouch HMI can create an instance of that object and the symbol instance references the new instance.

The following example shows you how to create a new Application Server object instance automatically.

Create a new Application Server object instance automatically

1. Create an object template **$Valve1** and open it in the IDE object editor.

2. On the **Graphics** tab, add an Industrial Graphic called **ValveSymbol**.
3. Create a derived template of the InTouchViewApp object and open it in WindowMaker.
4. Create a new InTouch window and embed the Industrial Graphic **ValveSymbol** from the object template $Valve1. WindowMaker prompts you for an instance name.
5. Type a name, for example `Valve1_E122` and select **OK**. The Industrial graphic is pasted on the InTouch window and the AutomationObject instance Valve1_E122 is created in the Galaxy.

Convert InTouch windows to Industrial Graphics

You can convert the windows of a managed InTouch application to Industrial Graphics. The converted Industrial Graphics appear in the WindowMaker Graphic Toolbox and the IDE Graphic Toolbox. In addition to graphics that appear in windows, InTouch scripts are converted to Application Server scripts.

Prepare to convert windows

Before you convert InTouch windows:

- Only windows from InTouch standalone and managed applications can be converted to Industrial Graphics.
- Windows must be closed in WindowMaker to be converted.

Convert windows

Only the symbols and scripts of a window are converted. The window's color, type, frame, title bar, size control, and **Close** button are excluded from the converted symbol.

Based on the InTouch graphic type, window graphics are converted as follows:

- All InTouch graphic primitives are converted to corresponding Industrial Graphic primitives.
- An InTouch Smart Symbol is converted to an Industrial embedded graphic.
- An Industrial Graphic within a window is converted to an embedded symbol. No new symbol is created for an embedded symbol.
- An InTouch symbol is converted to a group with the property TreatAsIcon=True.
- An InTouch cell is converted to a group with the property TreatAsIcon=False.
- InTouch Windows controls are converted to ArchestrA Windows controls.
- Some InTouch graphic components cannot be converted to Industrial Graphics:
  - InTouch Real-time and Historical trends cannot be converted.
  - ActiveX controls and the Distributed Alarm Display cannot be converted.

Convert animation scripts

All InTouch animation links embedded in windows are converted to the corresponding animations in the Application Server.

No validation warning or error messages are logged during the conversion. You should validate converted scripts

with Industrial Graphic script validation to find any unsupported script syntax.

The following exceptions occur when converting InTouch animation scripts:

- Discrete Alarm and Analog Alarm of Line Color and Fill Color animation links are converted to Boolean and Truth Table of Line Style and Fill Style animations.
- The ShowWindow animation link is converted to an action script with the ShowGraphic script function. The HideWindow animation link is converted to an action script with the HideGraphic script function.
- All InTouch tags configured in animation link expressions have the prefix "InTouch" added to the tag name. For example, Tag1 is converted to Intouch:Tag1.
- The prefix "galaxy:" of Application Server attribute references configured in animation links are removed. For example, galaxy:UD001.Value is converted to UD001.Value.
- All InTouch script functions configured in action scripts are not converted. All scripts are copied over except the InTouch tag and Application Server attribute reference handling.

Known limitations of windows conversions

Converting an InTouch window to an Industrial Graphic does not always achieve complete fidelity. In some cases, window components cannot be converted to a symbol. This section describes known limitations when converting an InTouch window to an Industrial Graphic and any alternative solutions.

- Converting a Window Containing a Vertical Mouse Release Slider SmartSymbol

  A Vertical Mouse Release Slider SmartSymbol contains two types of animation. The symbol uses fill animation to show the current measured value against a scale and a movable slider knob to set a value. A window containing an embedded Vertical Mouse Release Slider SmartSymbol does not retain the movable slider animation when converted to an Industrial Graphic.
- Converting a Window Containing a Symbol Factory Symbol

  Not all types of animations incorporated in Symbol Factory symbols can be converted to an Industrial Graphic. The following types of Symbol Factory symbol animations cannot be converted:

  - Percent Fill
  - Line Color
  - Horizontal Movement
  - Vertical Movement
- Converting a Window containing InTouch ActiveX controls, InTouch OCX DLL, or InTouch DLL objects

  The Windows Conversion does not support InTouch ActiveX controls, InTouch OCX DLLs, and InTouch DLL objects. These components will not be included in the new symbol.
- Migrating InTouch Translation Strings to Industrial graphics

  InTouch translation strings are stored in an XML file in the application's folder. The translated strings of each language are placed in separate XML files. Translations of all windows and SmartSymbols strings are available from the XML file in the application directory after importing localized strings.

  The following procedure explains how to use the Language Assistant tool to migrate InTouch window localization strings to an Industrial graphic.

  1. Convert an InTouch window to an Industrial Graphic.
  2. Import the contents of an InTouch language XML file to Language Assistant to create a global dictionary

of phrases.

3. Export an Industrial Graphic to an XML file.
4. Import the symbol XML file to Language Assistant, which will automatically apply a global dictionary translation to the Industrial Graphic phrases.
5. Publish the Industrial Graphic XML file from Language Assistant.
6. Import the XML file to the Industrial Graphic Toolbox containing the translated text strings.

- Migrating InTouch Scripts to Industrial Graphics

  Scripts containing InTouch or QuickScript functions cannot be converted to an Industrial Graphic.

- Migrating InTouch History Objects to Industrial Graphics

  InTouch windows containing an InTouch Historical Trend are not completely converted to Industrial Graphics. Only some parts of the Historical Trend may appear in the converted symbol. Trend components like scooter bars may not appear in the converted Industrial Graphic.

After converting windows

After conversion, the symbol is added to the System Platform IDE **Toolbox** and the WindowMaker **Industrial Graphic Toolbox**. The original converted InTouch window is backed up. A new InTouch window is created in the InTouch application, which embeds the newly created Industrial Graphics.

If only a subset of the window is converted, then the user must manually verify that the converted windows will work with the non-converted window.

The name of the converted window is assigned by default as the name of the new Industrial Graphic. If the InTouch window name contains unsupported characters, an underscore (_) replaces each unsupported character. If the symbol already exists, a numerical suffix is appended to the name of the converted symbol. For example, Main_001.

The special characters dollar ($), pound (#) and underscore (_) are the only exceptions.

If an application is migrated, editing the window or modifying any of the window properties will replace any special characters in the window name to underscore (_). This is applicable for all types of windows including application, frame and, template.

A new toolset is created and assigned the InTouch application name for all converted symbols. The InTouch folder hierarchy is maintained by the toolset after the windows conversion.

- A converted symbol from an unassigned InTouch window is added to a toolset assigned the InTouch application name.
- If the window belongs to an assigned area, the original folder structure is created in both toolboxes and the name of the InTouch application is assigned as the top root folder name.

Complete the window conversion procedure

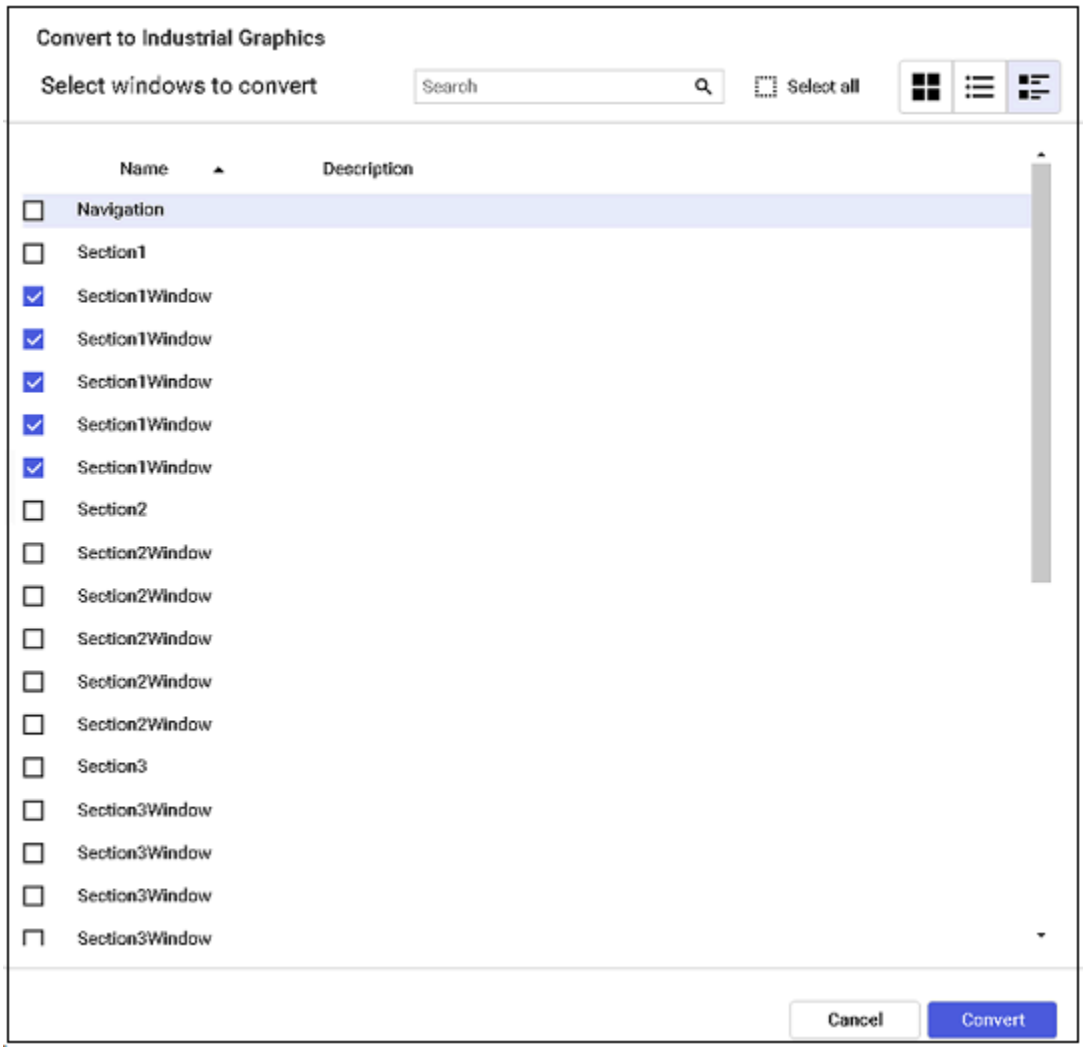
Convert windows to Industrial Graphics

1. Close all InTouch windows that will be converted to symbols.
2. Select the window to be converted from the **Windows** pane.
3. Right-click to show the shortcut menu and select **Convert To Industrial Graphic...**.

The **Select Windows to Convert** screen appears.

4. Select the windows that you want to convert.

   By default, the window selected in the previous screen will be checked. You can now select additional windows.



5. Select **Convert**.

   A message appears and indicates the windows are being converted in succession. After the windows are converted, a succession of **Check In** dialog boxes appear to enter an optional comment for each window that was converted.

6. Observe the WindowMaker **Industrial Graphic Toolbox** and the System Platform IDE **Graphic Toolbox.**

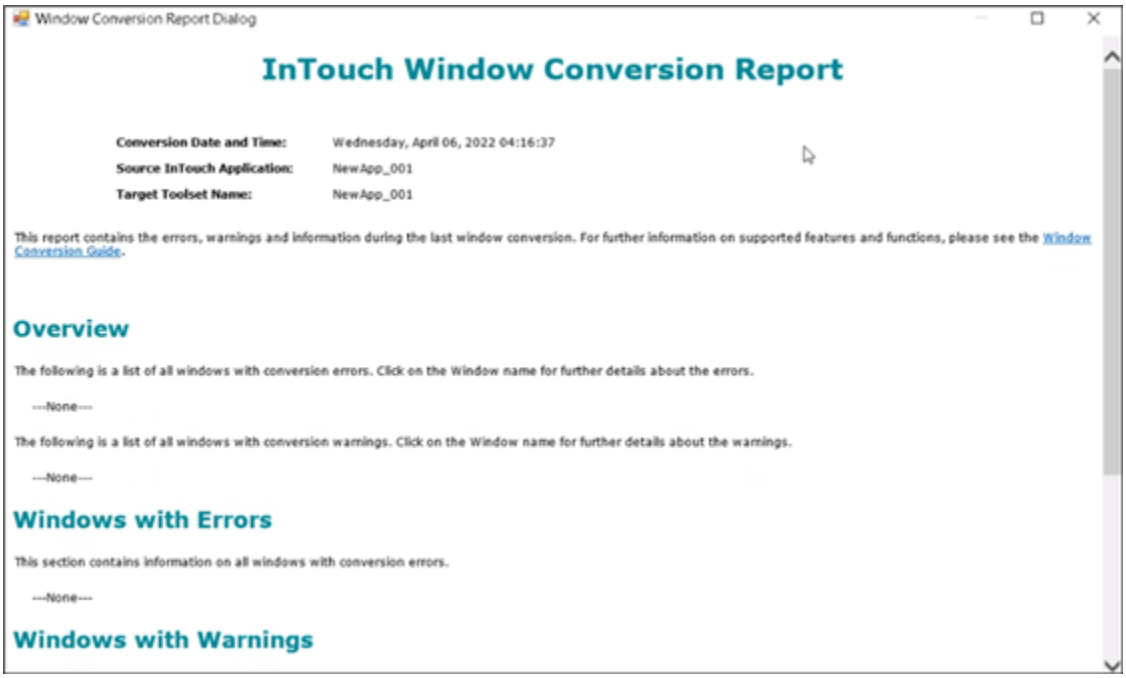
   The converted windows should appear as Industrial Graphics in both tool boxes.

Diagnose window conversion errors

During conversion, a progress bar shows warning or error messages that occur during the window conversion. A custom log flag is created containing useful diagnostic information for every element, animation, or script while converting windows.

After conversion, select **Window conversion Report** from the conversion progress bar to export all messages to

an HTML conversion report that can be viewed in a message window.

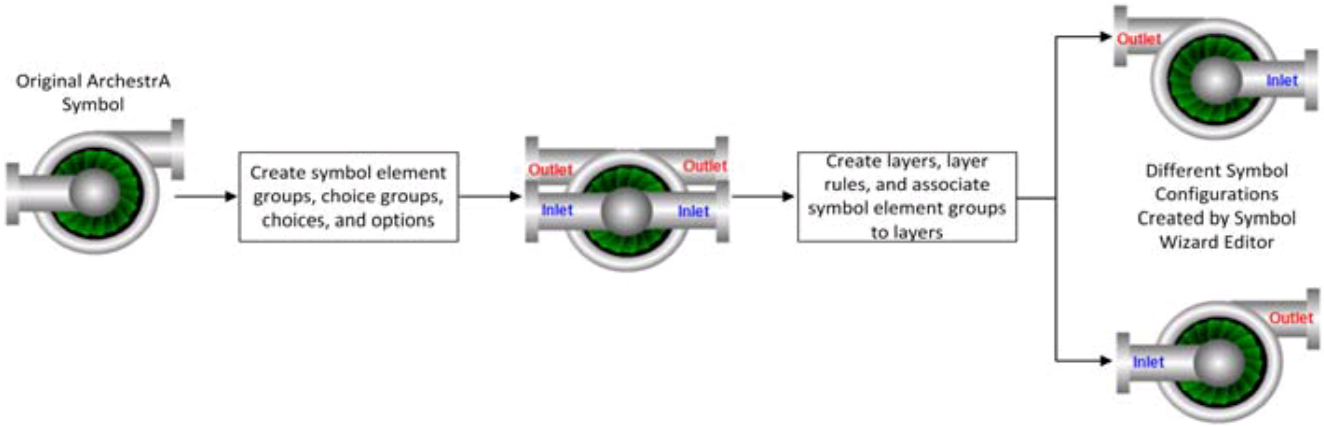


Both the conversion progress dialog box and the conversion report include the following information about a window conversion:

- Window name
- Conversion status that indicates if the window was converted successfully or if errors occurred.
- Warning messages that occurred during the conversion.
- Error messages that occurred during the window conversion.

Create Symbol Wizards with the Symbol Wizard Editor

The Industrial Graphic Editor includes the Symbol Wizard Editor, which can be used to create different visual and functional configurations of a symbol. These multi-configuration symbols are called Symbol Wizards. An example of creating a Symbol Wizard is a pump symbol that shows the outlet pipe on the left or right of the central blade housing. Using the Symbol Wizard Editor, both configurations can be included in one Symbol Wizard.



Symbols Wizards are not associated with any specific Application Server object template or Application Server object instance. Except for the ability to select a specific symbol configuration, Symbol Wizards behave like standard Industrial Graphics.

The ability to create Symbol Wizards with different configurations reduces the number of required symbols that need to be created for an application. Also fewer symbols results in lower maintenance and fewer design time issues.

Create Symbol Wizards

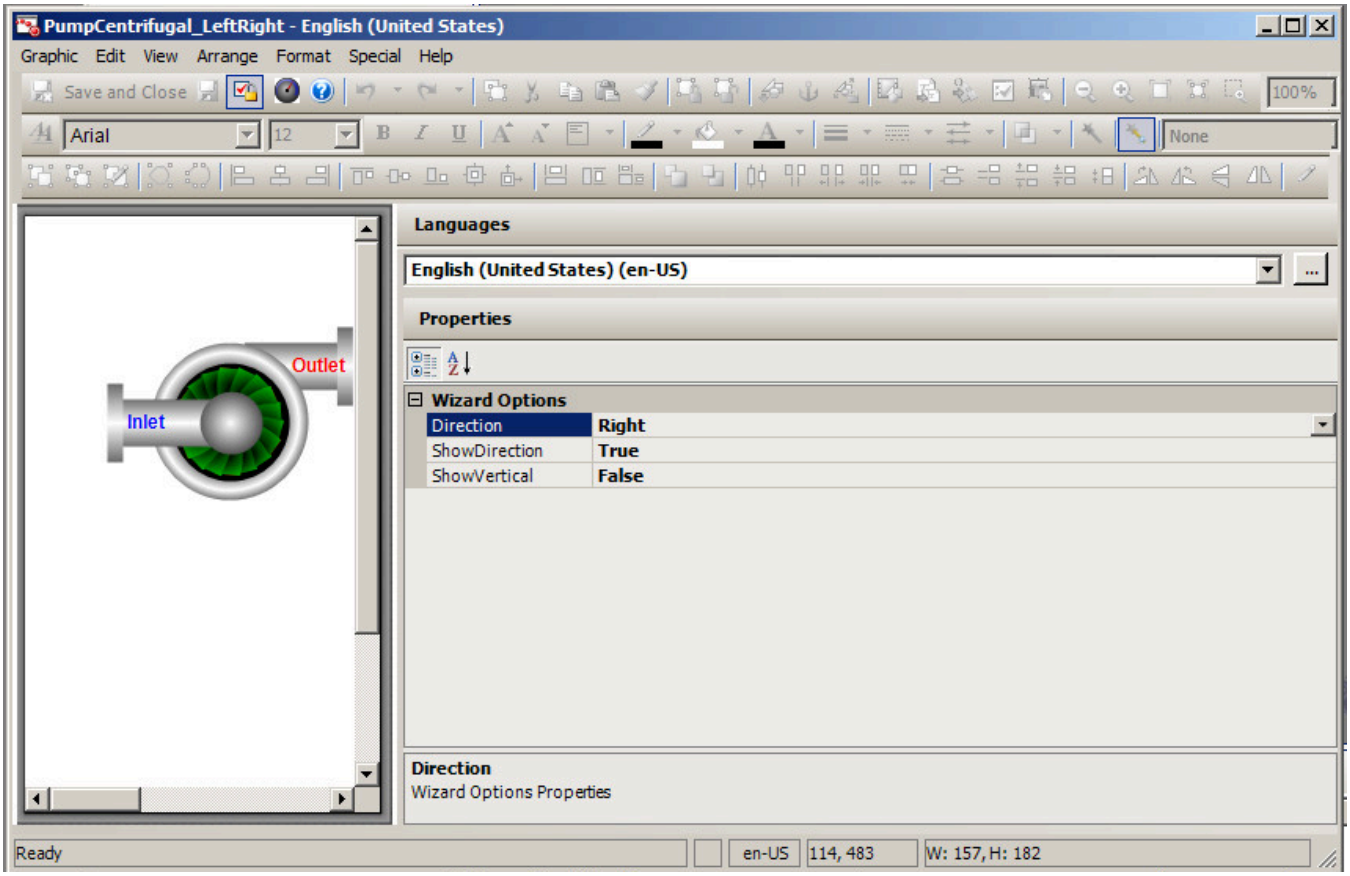
The process of creating and implementing a Symbol Wizard has two work flows, a Designer workflow and a Consumer work flow:

- A Designer creates multiple configurations of a symbols using the Symbol Wizard Editor.
- A Consumer selects the appropriate configuration of a Symbol Wizard and embeds the symbol into managed InTouch applications.

Symbol Wizard Designer workflow

The Designer uses the Symbol Wizard Editor to define the various symbol configurations and assign graphic elements, custom properties, and scripts to symbol configurations in the form of layers. Rules are assigned with Choice Groups, Choices, and Options that determine when a layer is included in a symbol configuration. The Designer selects a configuration to be the symbol default that appears when the symbol is embedded in a managed InTouch application.

After creating all symbol configurations, the Designer verifies how each configuration of a symbol will be presented to a Consumer using the Symbol Wizard Preview. Designers use the **Wizard Options** to verify that each configuration appears as designed based on the layer rules set for the symbol.



When a Symbol Wizard is finished, the Designer saves it to the Galaxy library so that it is available to embed into managed InTouch applications.

For more information about creating Symbol Wizards, see Industrial Graphic Editor User's documentation.

Symbol Wizard Consumer workflow

When a Consumer embeds an instance of a Symbol Wizard into a managed InTouch application, the symbol's default configuration is selected. The Consumer can change the configuration by selecting options from the Symbol Wizard's **Wizard Options** view. Depending on the selected configuration, there can be additional configuration-related properties that can be selected by the Consumer.

While the InTouch application is running, the Symbol Wizard appears as the configuration selected by the Consumer. A symbol's configuration cannot be changed during run-time.

For more information about how to embed Symbol Wizards into a managed InTouch applications, refer to Embed symbol wizards.

Understand trend pen historical data retrieval

When an application containing a Trend Pen starts running, a Historian query retrieves data for the entire Trend Pen period. Real-time data is plotted on the trend line during the period that historical data is being retrieved. After retrieving historical data, it is added to real-time data to back fill the trend line for the entire period.

The following procedure shows the steps to configure a Trend Pen. Typically, configuring a Trend Pen includes

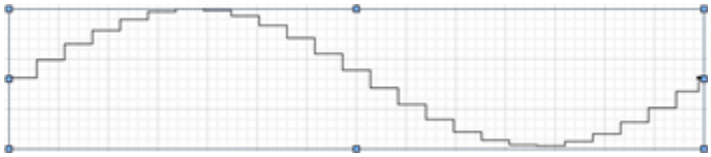
several steps to place the Trend Pen next to a meter graphic to visually indicate the Trend Pen plot shows the changes in the graphic's process value over time.

To configure Trend Pen

1. Select **Trend Pen** from the Visualization folder.

   The cursor changes to a cross hair when placed over the Industrial Graphic Editor canvas area.
2. Place the cursor over the canvas at the position you want to set as a horizontal boundary of the Trend Pen plot.
3. On the canvas, drag to select the rectangle for the Trend Pen control.

   The horizontal and vertical boundaries of the Trend Pen rectangle represent the drawing area of Trend Pen plot during run time. The horizontal axis of the graphic rectangle represents the time period of the trend. The vertical axis represents the range of the trend's possible values.

   The **Trend Pen** dialog box appears. You can also show the **Trend Pen** dialog box by double-clicking on the Trend Pen graphic or selecting **Edit Animations** from the **Special** menu.
4. Enter a reference in the **Reference** field.

   The reference is the data source that appears as the value shown by the trend, which can be an external reference like an object's attribute, an analog tag value, or a custom property. Constants and expressions are not allowed.
5. Select **Historian** or **InTouch Log History/LGH** for the Historical Source. If you select Historian proceed with Step 7.
6. On selecting **InTouch Log History/LGH** you can use the icon against the **UNC Path** to toggle input to the field as an Expression or Static text.
   - If you select the Expression mode, you can select the ellipsis button to launch your HMI's attribute/tag browser and select a Custom property, attribute or tag.
   - If you select the Static Text mode, you can select the ellipsis button to launch the file browser where you can specify the LGH file name.
7. Select **Auto-Detect** or **Expression** for the method to identify the location of the Historian.
   - **Auto-Detect**:The Historian server is auto-detected from the AppEngine on which the reference attribute is running. For example, if the **Reference** field is set to UDO.UDA1, then **Auto-Detect** is set to the Historian server name configured for the AppEngine on which UDO is running. Auto-Detect is only valid for Application Server References.
   - **Expression:** When an expression or reference is entered in the **Server Name** field, the Trend Pen connects to the specified Historian Server.

   The icon to the left of the **Server Name** field toggles input to the field as an expression or Static Text mode.

   A Trend Pen only shows live data if the **Server Name** field is left blank in **Expression** mode.
8. Select **Moving** or **Fixed** as the type of trend time period.
   - **Moving:** The start time of a trend period is the current time, and the end time is the duration of the time period from the start time. The start time for the next period is set to the end time of the previous trend period.

- **Fixed:** In a Fixed trend time period, the StartTime and EndTime properties do not change automatically. The start time of a trend period is the current time initially. The StartTime property can be changed by a script.

The EndTime property of a trend period (both Moving and Fixed) is read-only. The end time of a trend period is calculated from the specified start time and duration of the period.

9. Set the X-axis time period of the trend line in the **Duration (Minutes)** field.

   The trend time period can be specified as a constant, an external reference, an expression, or custom property. If a floating point number is entered, the period is rounded up to the nearest minute.

   The minimum trend period is 1 minute and the maximum period is 10080 minutes (1week).

10. Select **Auto-Range** or **Clip out of Range Values** for the scaling method to place process values on a trend line.

    If **Auto-Range** is selected, the **Min Range** and **Max Range** fields are disabled. The Y-axis of the trend line is automatically adjusted to show the full range of trend values within the upper and lower boundaries of the Trend Pen graphic.

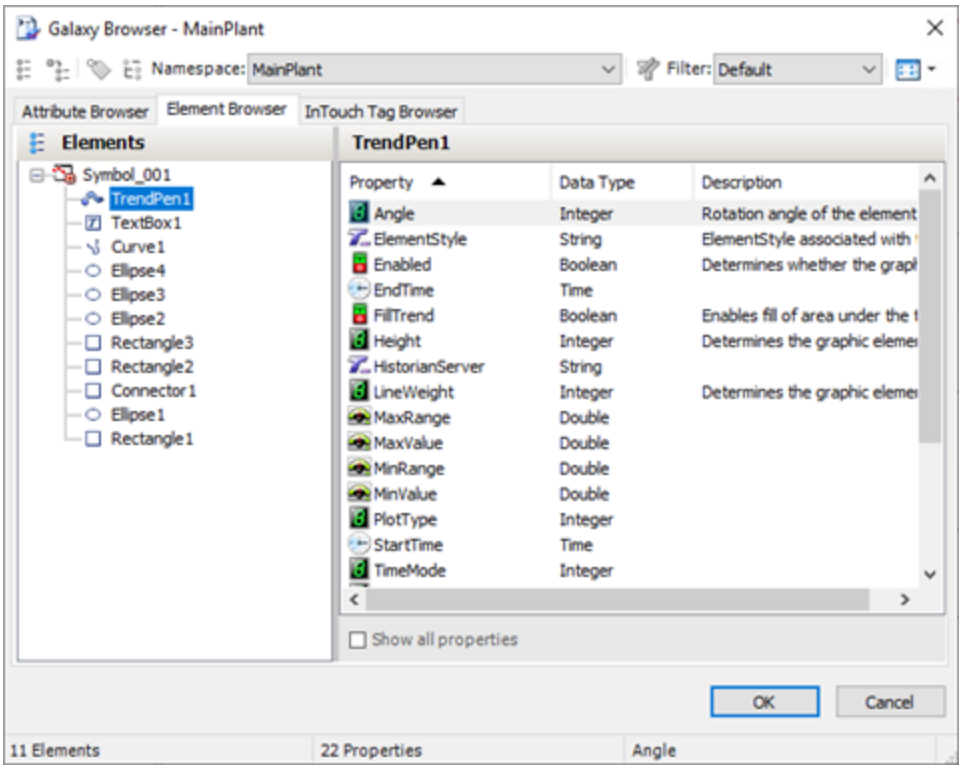
    If **Clip out of Range Values** is selected, the **Min Range** and **Max Range** fields are enabled. **Min Range** and **Max Range** set the lower and upper limits of the trend's Y-axis value range. Both fields can be set to constants, external references, or custom properties.

    When a value exceeds the trend's minimum or maximum limits using **Clip out of Range Values**, the trend line is truncated at the limit of the value range and appears as a horizontal line for the period when the process value is out of trend's value range.

11. From **Plot Type**, select **Step Line** or **Line** as the type of trend plot.
    - A **Step Line** plot draws a horizontal line from a trend data point to the time of the next data point on the trend's X-axis, and then draws a vertical line to the data point.
    - A **Line** plot draws a line directly to each successive point within the trend period.

Change trend pen properties during runtime

A Trend Pen contains a collection of properties whose values can be modified during run time to change the visual and functional characteristics of a trend. The following illustration is one example of a browser or explorer typically available in HMI software to search for attributes or tags, the values of which can be changed at run time.



The subtopics below describe properties that are typically modified during run time to change the visual and functional characteristics of a Trend Pen.

MinValue property

During run time, the value of the MinValue property can be modified to change the minimum measured value shown by a trend. During run time, MinValue can be a read/write or a read only property based on the value assigned to the Trend Pen's **Y Axis Range** option during design time.

- When **Y Axis Range** is set to **Auto-Range** during design time

  The minimum measured value shown by a trend is set to the minimum value of data received from the Historian or from current data during the trend period. MinValue is read only.
- When **Y Axis Range** is set to **Clip out of range** values during design time.

  The minimum measured value shown by a trend is set to a minimum limit set from the **Min Range** option during design time.

  MinValue is read/write and can be changed during run time. When MinValue is changed during run time, the trend line is redrawn based on the values assigned to the MinValue and MaxValue properties.

  When the values assigned to MinValue and MaxValue properties are the same, the trend's Y Axis Range automatically changes to Auto-Range.

MaxValue property

During run time, the value of the MaxValue property can be modified to change the maximum measured value shown by a trend. During run time, MaxValue can be a read/write or a read only property based on the value

assigned to the Trend Pen's Y Axis Range option during design time.

- When **Y Axis Range** is set to **Auto-Range** during design time

  The maximum measured value shown by a trend is set to the maximum value of data received from the Historian or from current data during the trend period. MaxValue is read only.
- When **Y Axis Range** is set to **Clip out of range** values during design time

  The maximum measured value shown by a trend is set to a maximum limit set from the **Max Range** option during design time.

  MaxValue is read/write and can be changed during run time. When MaxValue is changed during run time, the trend line is redrawn based on the values assigned to the MinValue and MaxValue properties.

  When the values assigned to MinValue and MaxValue properties are the same, the trend's Y Axis Range automatically changes to Auto-Range.

StartTime property

During run time, the value of the StartTime property can be modified to change the start time of a trend period based on the value set to the Trend Pen **Time Period** during design time.

- When **Time Period** is set to **Fixed** during design time

  The default value assigned to the StartTime property is the time when the Trend Pen first appears in the WindowViewer, also the start time changes as time passes. StartTime is read/write and can be changed during run time. When value of StartTime changes, the Trend Pen re plots the trend using new StartTime value.
- When **Time Period** is set to **Moving** during design time

  The value set to the StartTime property is the current system date and time. StartTime is read only.

EndTime property

During run time, the value of the EndTime property can be modified to change the end time of a trend period based on the value set to the Trend Pen **Time Period** during design time.

- When **Time Period** is set to **Fixed** during design time

  The default value assigned to the EndTime property is the end time set during design time. EndTime is read/write and can be changed during run time. When value of EndTime changes, the Trend Pen re plots the trend using new EndTime value.
- When **Time Period** is set to **Moving** during design time

  The value set to the EndTime property is the current system date and time. EndTime is read only.

PlotType property

During run time, the value of the **PlotType** property can be modified to change the type of trend plot.

- When **PlotType** is 0, the Trend Pen plot type is Step Line. The default.
- When **PlotType** is 1, the Trend Pen plot type is Line.

The value of **PlotType** is ignored if the value is neither 0 nor 1.

When the value of **PlotType** changes in run time, trend data is retrieved again before drawing the trend.

TimeMode property

During run time, the value of the **TimeMode** property can be modified to change the type of trend time period.

- When **TimeMode** is 0, the trend time period mode is **Moving**. The end time of the trend is the current time. The default.
- When **TimeMode** is 1, the trend time period mode is **Fixed**. The start time of the trend is when the Trend Pen first appears in the WindowViewer, also the start time changes as time passes.

The value of **TimeMode** is ignored if the value is neither 0 nor 1.

When the time period changes from Moving to Fixed during run time, the trend's start time and end time remain the same before switching, and the data remains as well. When the time period changes from Fixed to Moving during run time, data is retrieved again before drawing the trend. The trend's start and end times are adjusted automatically by Moving mode.

FillTrend property

During run time, you can use the FillTrend property to change the appearance of the area below the trend pen curve.

The FillTrend property can also be modified using scripts. `For example TrendPen6.FillTrend = not TrendPen6.FillTrend;`

The Fill Property in Fill Style animation, Element Style animation, or Element Style will now apply to the FillTrend property according to the general element style precedent rules.

For more information on the Fill Style animation, see Setting Fill Style.

For more information on Element Style animation, see Configuring an Animation Using Element Styles.

Note: The FillTrend property can be enabled during design or runtime. In runtime the color will default to white, if no color is selected in design time.

Configure a Multi Pens Trend

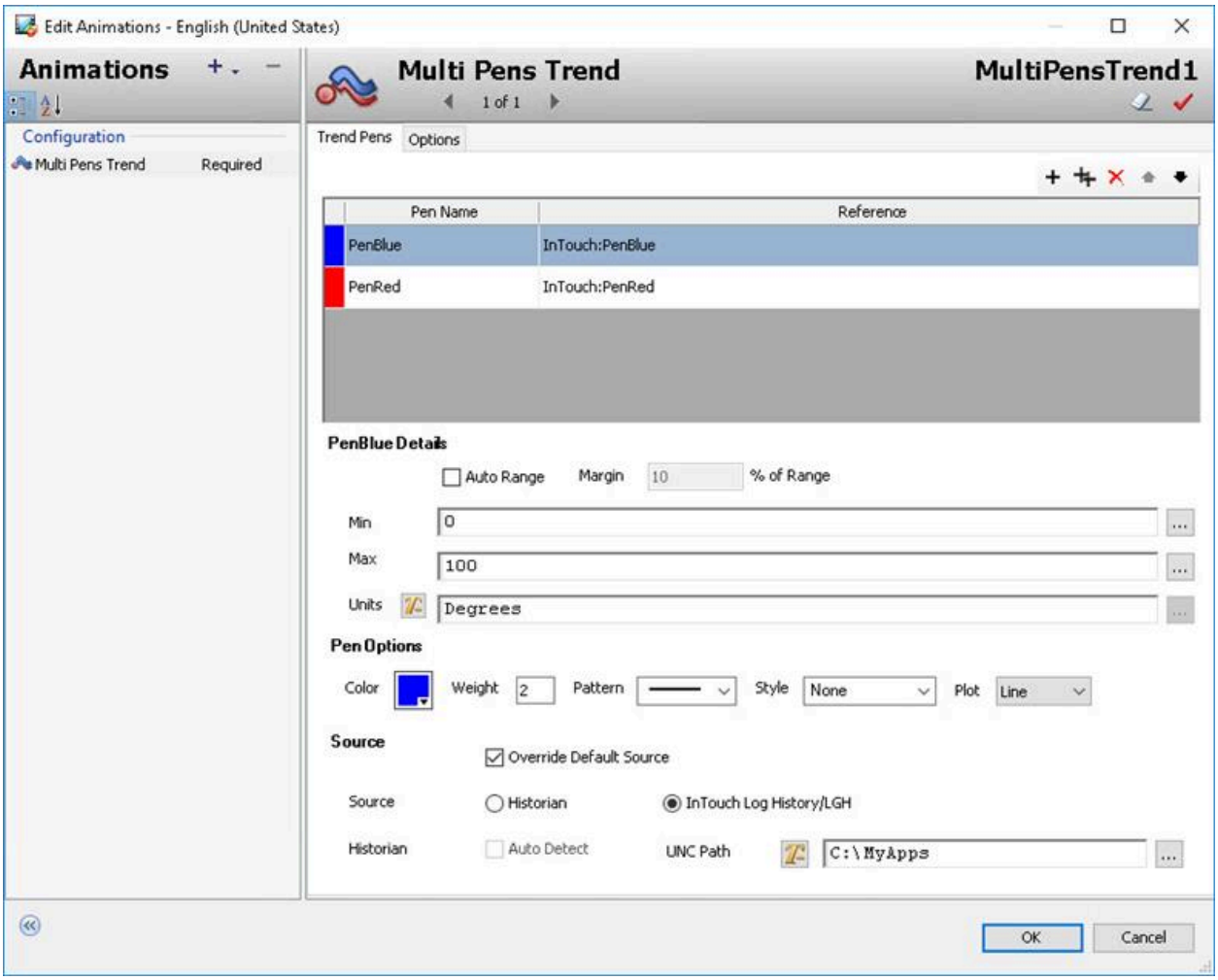
A Multi Pens Trend is like a Trend Pen and displays a succession of process values but with multiple trend lines. See the sections under Configuring a Trend Pen for more information on the types of pen trends.

Note:
- Multi Pens Trend has not been tested on a non-English Language Operating System.
- Performance Index calculation for Multi Pens Trend is not included in the GPI.
- AVEVA OMI ViewApps do not support Multi Pens Trend.
- Web Client does not support Multi Pens Trend.

To configure a Multi Pens Trend:

1. Select the **Multi Pens Trend** element.
2. Click over the Graphic Editor canvas.

3. The **Edit Animations** dialog box appears. For each pen, configure the following sections in the **Trend Pens** tab:
   - Pen Name and Reference
   - Pen Details
   - Pen Options
   - Source



Configure the pen name and reference

In the Trend Pens tab, for each pen:

1. Enter a name of the pen in the **Pen Name** field.

   The pen name field must meet the following rules:
   - Only static text strings are allowed.
   - Must contain one letter.
   - The maximum number of characters is 32.
   - Valid characters are alphanumeric and special characters ($, #, _).

- Two trend pens cannot have the same name.
- The pen name is case-insensitive.

  A tooltip will show the configured Pen name.

2. Enter a reference in the **Reference** field. Use the Ellipsis button next to the field to select the reference.

   The reference is the data source that appears as the value shown by the trend, which can be an external reference like an object's attribute, an InTouch analog tag value, or a custom property. Constants and expressions are not allowed. For more information, see Setting the Input Mode.
3. By default, two empty rows will be created in the grid, and you can create a maximum of 8 pens.
4. Use the + icon to add more trend pens. Use the ✕ icon to remove any trend pens. Use the ▲ and ▼ arrows to change the order in which the pens are displayed. Use the ⧉ icon to duplicate an existing pen configuration.

Configure the pen details

Under the Pen Details section, you can configure the range and units of each individual pens.

1. Select **Auto Range** for the scaling method to place process values on a trend line.

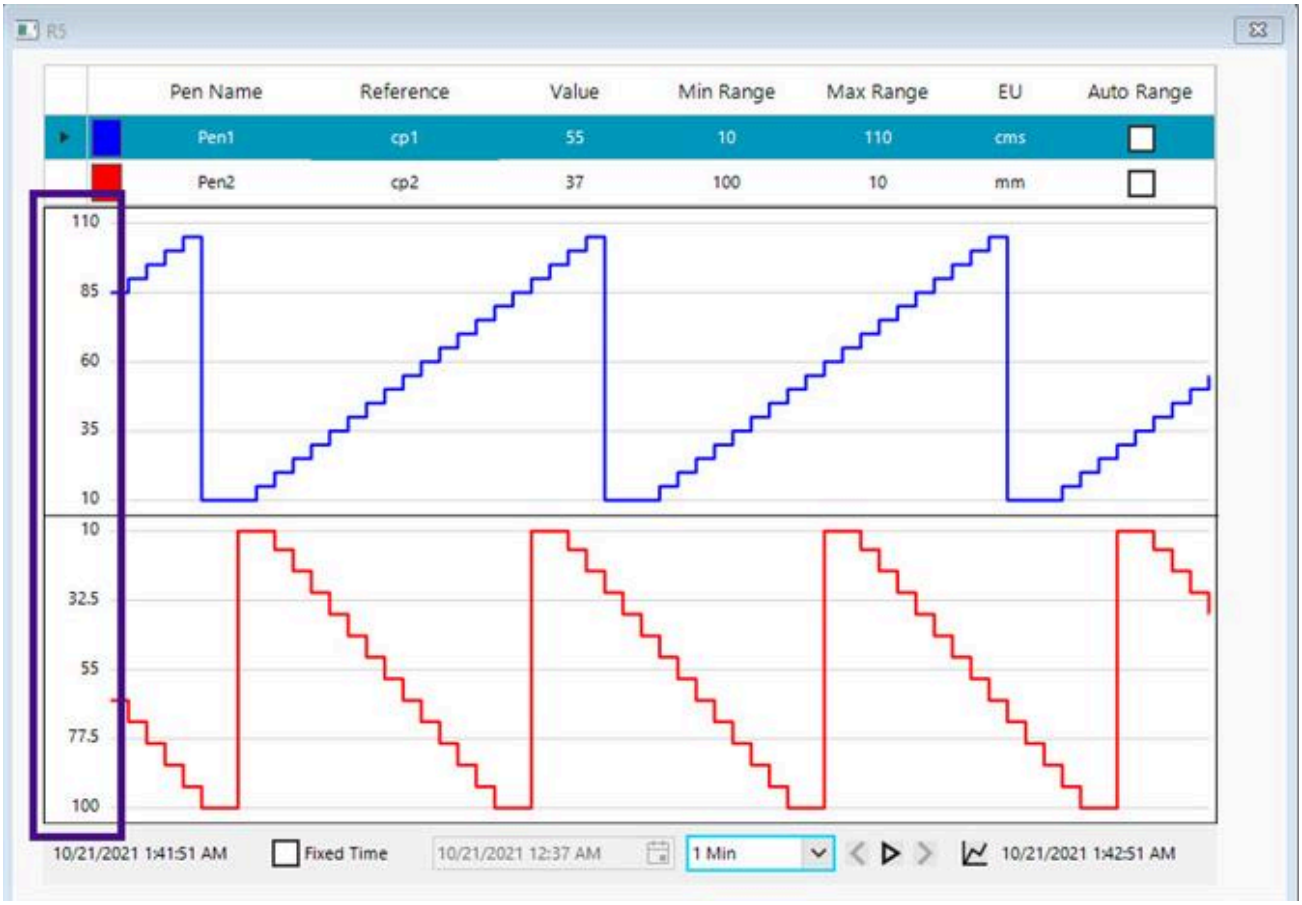
   If you select **Auto Range**, the **Min** and **Max** fields are disabled. The Y-axis of the trend line is automatically adjusted to show the full range of trend values within the upper and lower boundaries of the trend.

   a. Specify the **Margin** percentage value. In runtime a margin for the trend area will be dynamically displayed based on the trend values and the margin% specified. This is only for display purposes and does not affect the values of the axis.

      For example: If the minimum value is 10 and the maximum value is 20. The range is 10 (20-10).
      If you specify a margin of 20%, then the margin will be calculated as Range*Margin%; 10*20%=2.
      The maximum value of the Y-axis will be Maximum Value + Margin; 20+2=22. The minimum value of the Y-axis will be Minimum Value - Margin; 10-2=8.
2. If **Auto Range** is not selected, specify the **Min** and **Max** values for the Y-axis of the selected pen.

   You can specify the values directly, or select **...** . The Galaxy Browser is displayed.

   If the Min value is greater than the Max value then in runtime the trend pen will be rendered differently and the scale will be opposite to when the Max value is greater than Min value.



3. Browse and select an attribute or tag.
4. Enter the **Units**. Use the Toggle button to enter a static text or select a reference value from the Galaxy Browser. For example: Degrees.
5. Select **OK**.

Configure the pen options

Under the Pen Options, you can configure the style of each individual pen.

1. **Color**: Select the color in which the pen will be displayed.
   a. You can use the Color Picker to select a color from an part of the screen.
   b. You can also use Load Custom Palette, by clicking **Load Palette**.
   c. You can also add colors to the Custom Palette by pressing the **+** and remove colors by pressing the delete button.
   d. After developing the custom palette you can save it for future use. Select **Save Palette**.
2. **Weight**: Enter the weight of the pen. The weight determines the thickness of the trend pen line.
3. **Pattern**: Select the display pattern for the trend pen line from a list of predefined options.
4. **Style**: You can select from a list of predefined Style options. The Styles are configured under Element Styles in the IDE. See Change the Visual Properties of an Element Style for more information.
5. **Plot**: This is the type of trend line to be displayed. Select between a Step Line and Line.

6. Select **OK**.

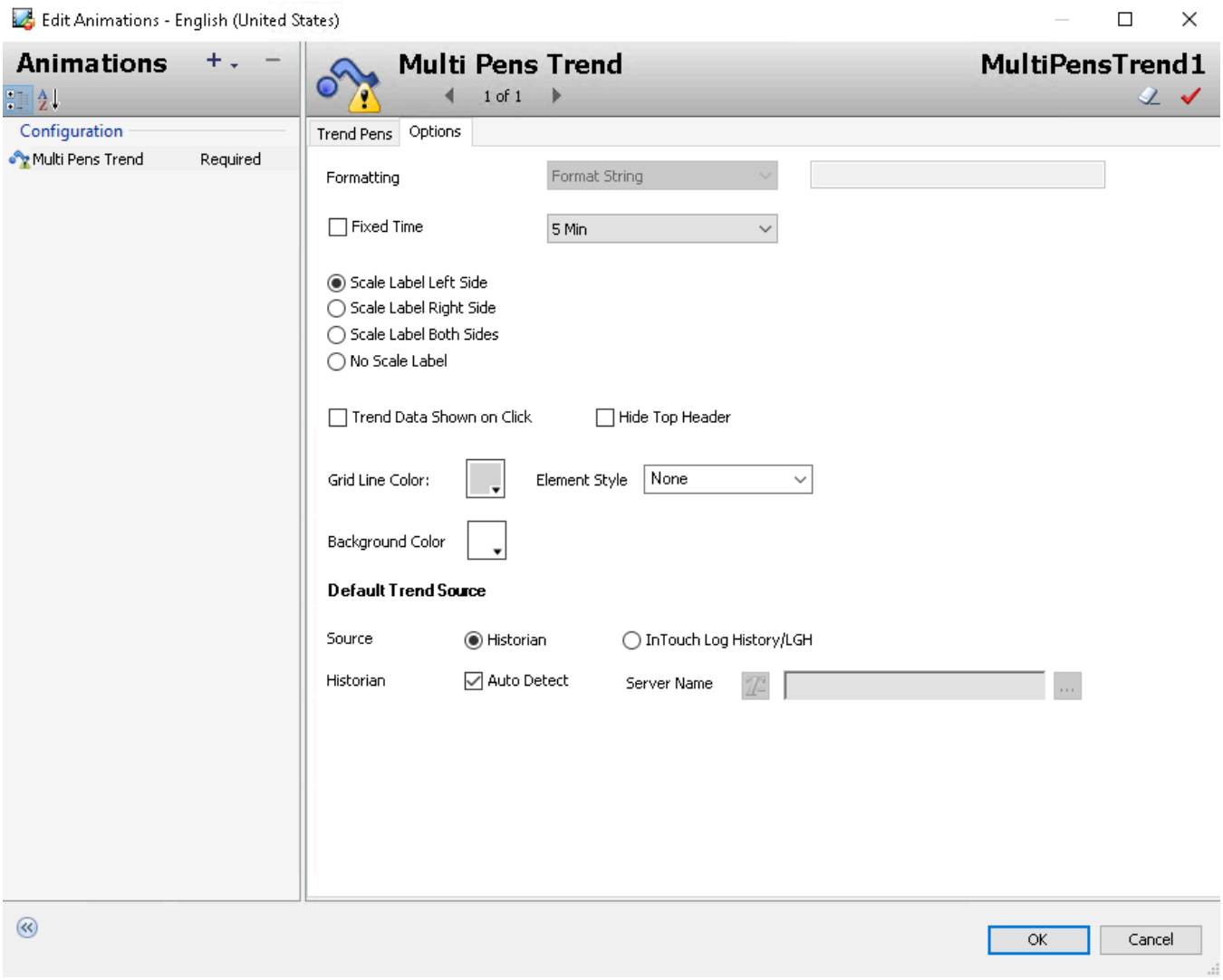
Configure the source

You can override the Historical source of data configured in the **Options** tab. See Customizing the Multi Pens Trends for more information.

- Select **Override Default Source** to change the source of Historical data.

Customize the Multi Pens Trend

You can customize the display of the Multi Pens Trend in runtime using the **Options** tab.



1. The **Formatting** options are disabled.
2. The **Fixed Time** setting can toggle between the Fixed Time Mode (when selected) and Moving Mode (when not selected). When the Fixed Time checkbox is selected the trend area will display trend data for a specific duration, and will not move.

3. Select the Time Interval from the list of predefined options in the drop down list. The time interval selected here will be set as the initial value in runtime.
4. Select where in the Trend Area the Scale Label should be displayed; On the left, right, on both sides or display no scale label.
5. Select **Trend Data Shown on Click** to customize the behavior of the cursor readout dialog in runtime.

   By default, the Cursor Readout dialog will be displayed when cursor hovers over the Trend Area. When this option is selected, you can show and hide the Cursor Readout dialog by clicking in the Trend Area.
6. To hide the top header, select **Hide Top Header**. Only the trend area and toolbar will be displayed.
7. Select the required color for grid lines using the **Grid Line Color** option.
8. Select a pre-defined style from the **Element Style** drop down list.
9. You can also choose a color for the background using the **Background Color** option.
10. Under the **Default Trend Source**, select Historian or InTouch Log History/LGH for the **Source** field.

    By default, Historian source and Auto Detect options will be selected. When **Auto Detect** is checked, the Server Name will be disabled.

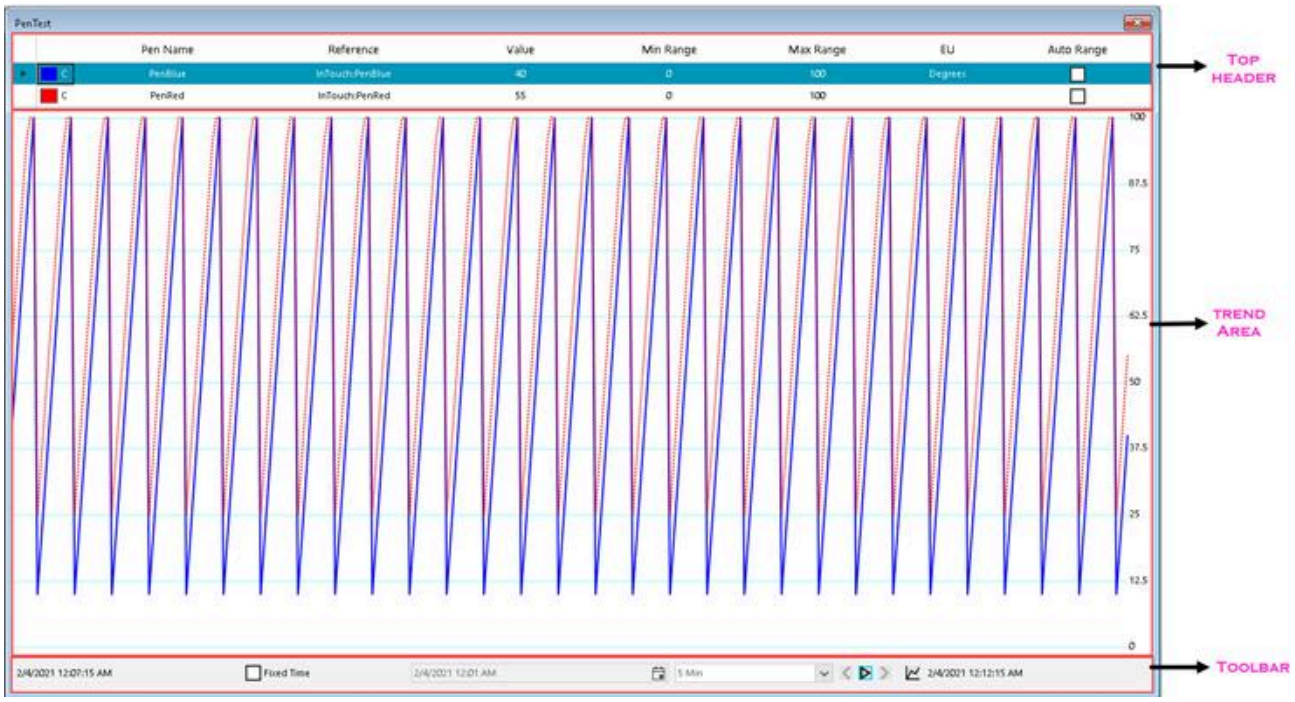
    When an expression or reference is entered in the Server Name field, the Multi Pens Trend connects to the specified Historian Server. The button to the left of the **Server Name** field toggles input to the field as an expression or Static Text mode.

    The Historian server is automatically detected from the AppEngine on which the reference attribute is running. For example, if the Reference field is set to UDO.UDA1, Auto Detect is set to the Historian server name configured for the AppEngine on which UDO is running. Auto Detect is only valid for Application Server References. A Trend Pen only shows live data if the Server Name field is left blank in Expression mode.
11. If you select **InTouch Log History/LGH**, use the button next to the **UNC Path** field to toggle input to the field as an Expression or Static text.
    a. If you select the Expression mode, you can select the ellipsis button to launch your HMI's attribute/tag browser and select a Custom property, attribute or tag.
    b. If you select the Static Text mode, you can select the ellipsis button to launch the file browser where you can specify the LGH file location.

Use the Multi Pens Trend at runtime

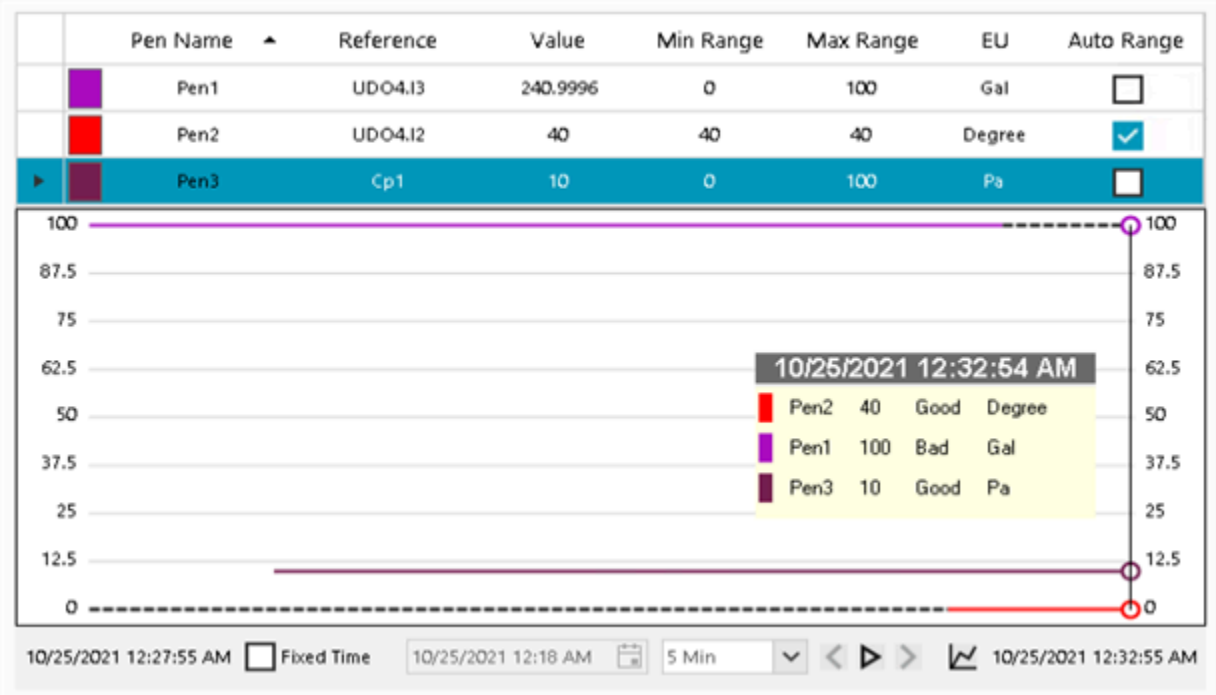
You can change the behavior of the Multi Pens Trend in runtime in WindowViewer. For example; adding pens, deleting pens or playing historical trend data. The Multi Pens Trend can be organized in 3 areas:

- Top Header: Lists the pen details in a table format
- Trend Area: The pens are visually represented in this area
- Toolbar: Contains the playback and stacking options



Cursor Readout Dialog

The cursor readout dialog displays the following fields; *Pen Name*, *Value, Quality* and *Engineering Units* when the cursor is placed at a specific point on a pen; i.e. at a specified time. To view the cursor readout dialog, hover the cursor on the Trend Area.



Displaying Data Quality

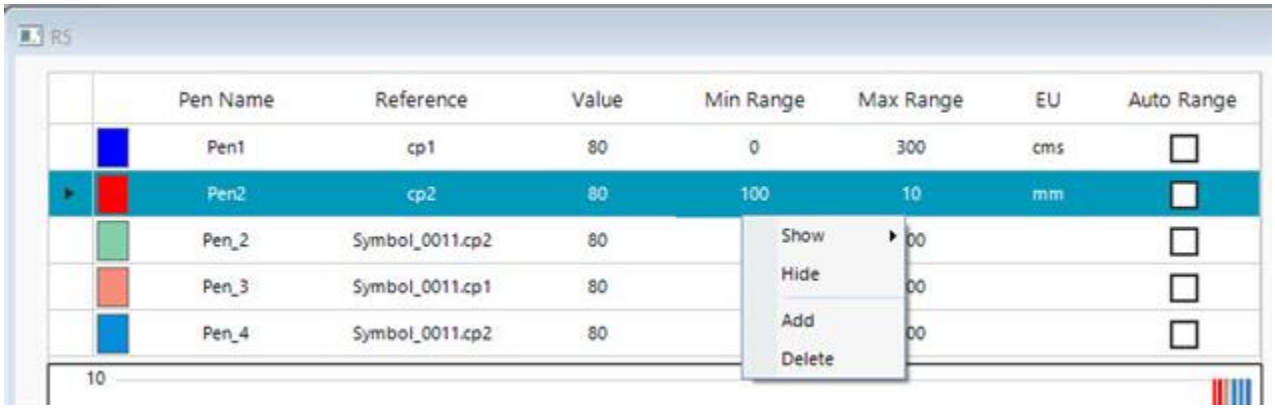
The style of the trend pen changes to a Bad Quality style when the reference does not have good quality data. The trend will continue to display using the last known good value. The style can be changed using Galaxy Styles in System Platform IDE.

Note: In a show graphic window, only the trend area will zoom in or zoom out and the status control bar will not move when zoomed using ctrl+ mouse wheel option.

Top header options

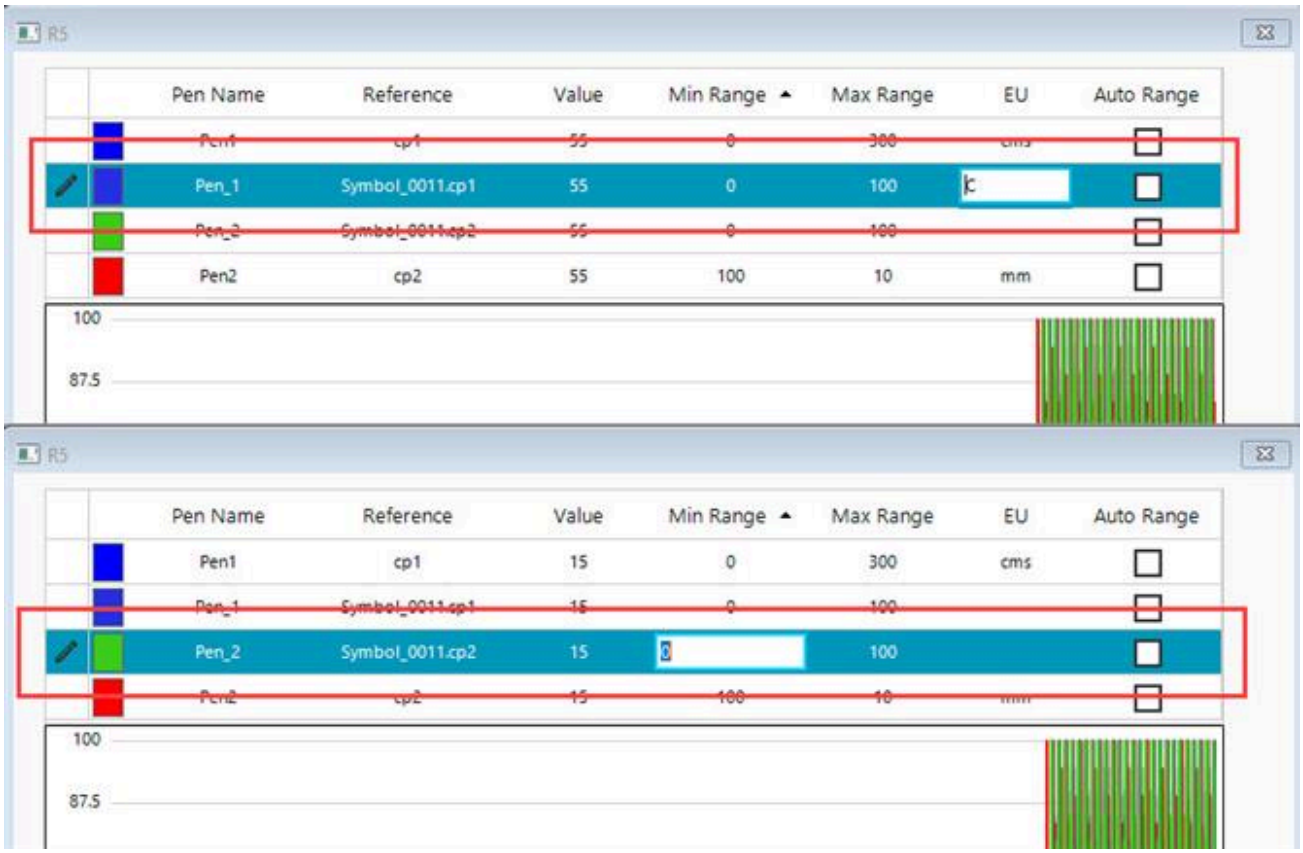
You can use the Top Header area to edit pen details in runtime. You can also use top header to:

- Hide and show individual pens
- Add a pen
- Delete a pen



Editing pen details in runtime

1. Double-click on the field to edit it.
2. Update the value.
3. Select outside the row to save the new value.



Hiding and showing individual pens

1. Right-click on a row.
2. Select **Hide** from the context sensitive menu.
   The pen is hidden from the top header area and the pen is not rendered in the trend area.
3. To show a pen, right-click in the top header area.
4. From the context sensitive menu, select **Show** and then select the pen name.
   The pen will be displayed in the top header and trend area.

Adding a pen

1. Right-click in the top header area, and select **Add**.
   A new row is included in the top header area.
2. Provide pen details including name and reference.
3. Press Enter.
   The trend area will start rendering the pen, if data is available.

Deleting a pen

- Right-click in the top header area, and select **Delete**. Alternatively you can press the delete key.
  The pen is deleted from the multi pens trend control.

Sorting the top header area

1. Select the heading name in the heading row to sort pen details.

   For the following heading names the top header will be sorted in ascending and descending order: Pen Name, Reference, EU.

   All other heading names will be sorted from smallest to largest.

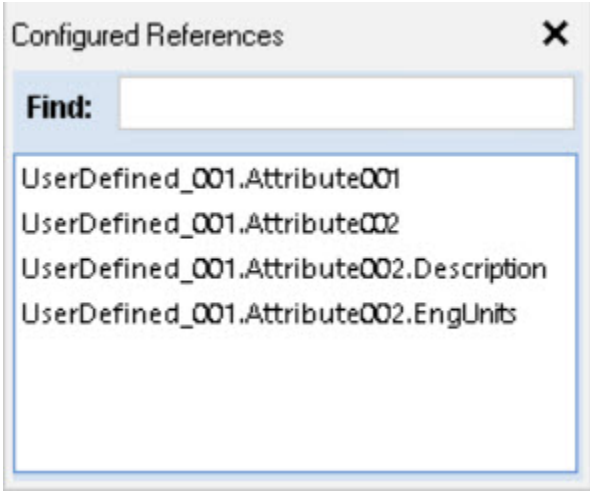
   All headings except Color can be used for sorting the top header area.
2. Select the heading name again to change the directionality of the sort.
3. Select the heading name a third time to return to the original state pre sorting.

Configured References dialog box

The Configured References dialog box allows you to view the attributes configured on the graphic or window. The dialog box can be resized and re-positioned on the window. WindowViewer will maintain the position and dimensions of the dialog box even on restart.

Launch the Configured References dialog box

- Right-click on the window. The Configured References dialog box appears.



Find an attribute

- Enter the first few characters name of the attribute in the Find field and press enter. The list of attributes matching the text will be filtered and displayed.

Copy an attribute name

- From the Configured References dialog box, right-click on the attribute and select **Copy**.

  The name of the attribute is saved to the operating system clipboard.

Add pens using drag and drop

You can add trend pens in runtime by selecting the reference and using drag and drop.

1. Right-click on the graphic or element, the Configured References dialog box appears with a list of all the available attributes for that graphic.
2. Select the reference and drag and drop it on the Top Header of the Multi Pens Trend. This will create a new pen and the trend area will start rendering the data for the selected pen.

Trend area scale

The scale of the trend area is calculated and displayed in runtime, depending on the values of the trend pens selected. The number of grid lines displayed depends on the height of the Trend Area. If the height of the Multi Pens Trend is changed the scale will change accordingly. Depending on the height, the number of intervals will be either 2, 4, or 8.

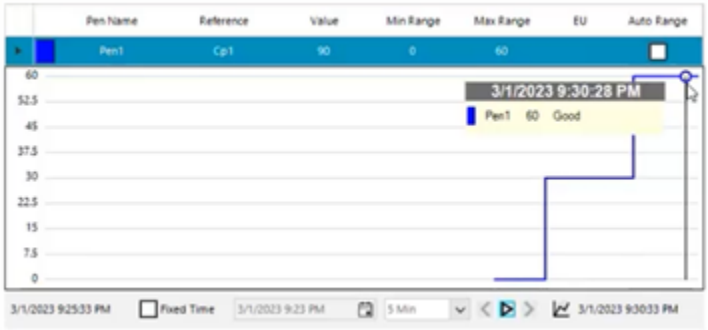
Change the minimum and maximum range

If the Auto Range checkbox is not selected, you can change the minimum and maximum range. The trend area will update the scale and the trend will be plotted with the new scale.

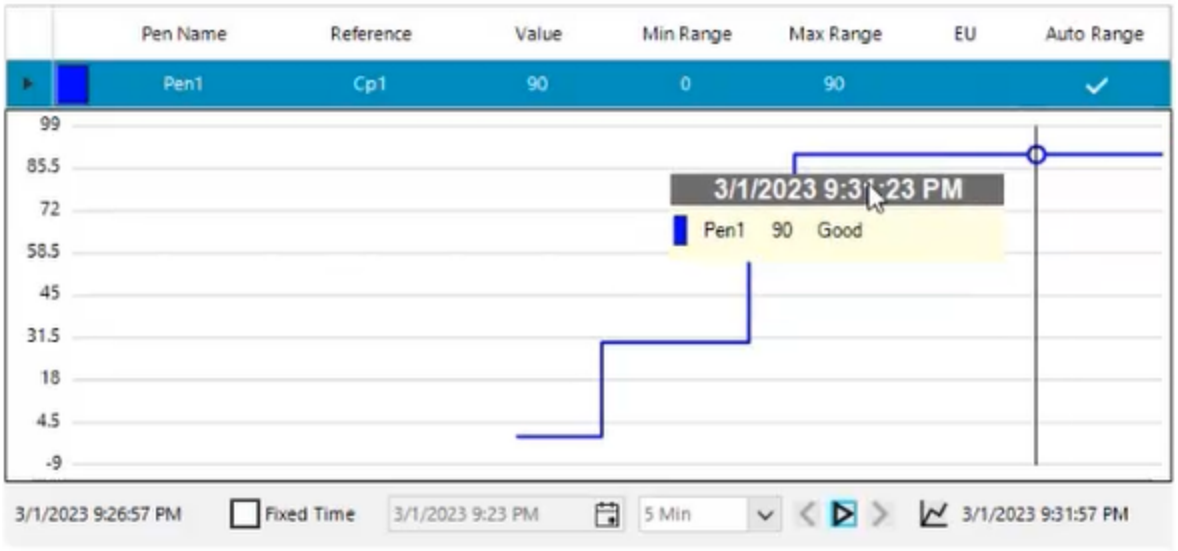
- For a selected row, select the Min Range and Max Range fields in the top header and enter the new value.

The scale of the pen in the trend area will be updated.

When the tag value is beyond the minimum and maximum range the trend line will be truncated to the minimum and maximum range value. The cursor readout always shows the value based on the trend line. In the below image even though the tag value is 90, since the maximum range is 60, the trend is truncated to 60 and the cursor readout is also showing 60.



To view the trend line without truncation and see the actual value in the cursor readout, select the Auto Range checkbox. If the minimum and maximum range values specified for a pen are equal, then it is considered as an Auto Range condition and it will ignore the values specified for a pen. In the below image since the Auto Range checkbox is selected, the trend line is not truncated and cursor readout is showing the actual value.



For more information on Auto Range, refer to the Enable auto range topic.

Enable auto range

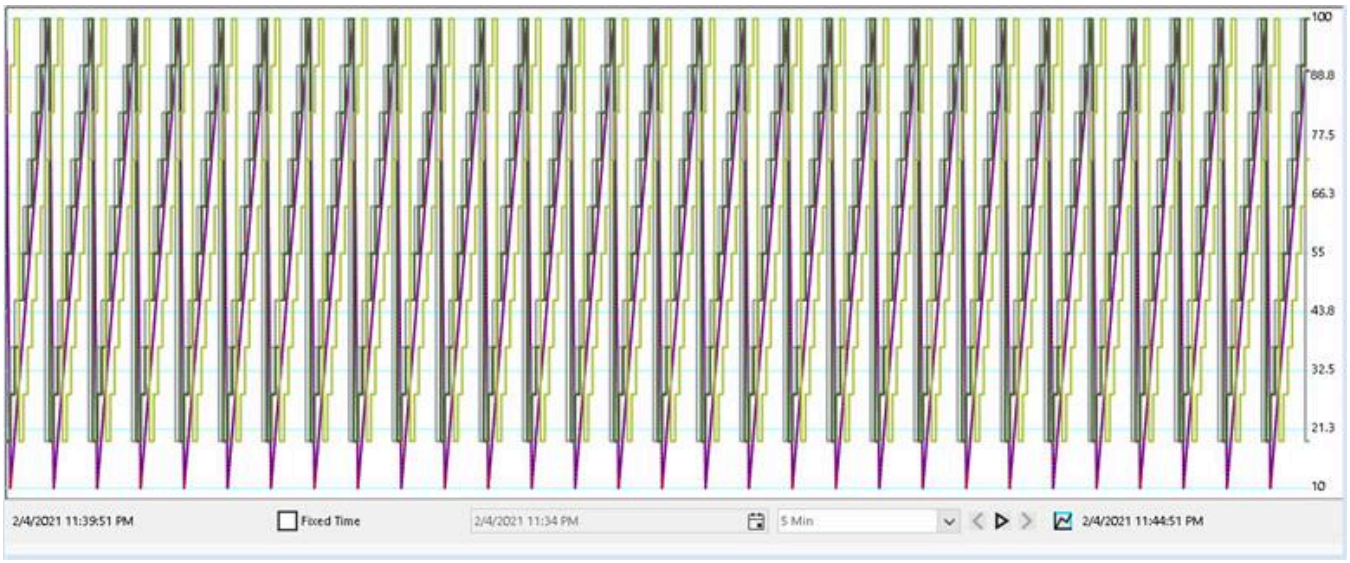
The Auto Range will update the scale of the Trend Area dynamically based on the trend pen values. It will ignore the minimum and maximum range value specified for a pen.

- Select the **Auto Range** checkbox against the required pen.

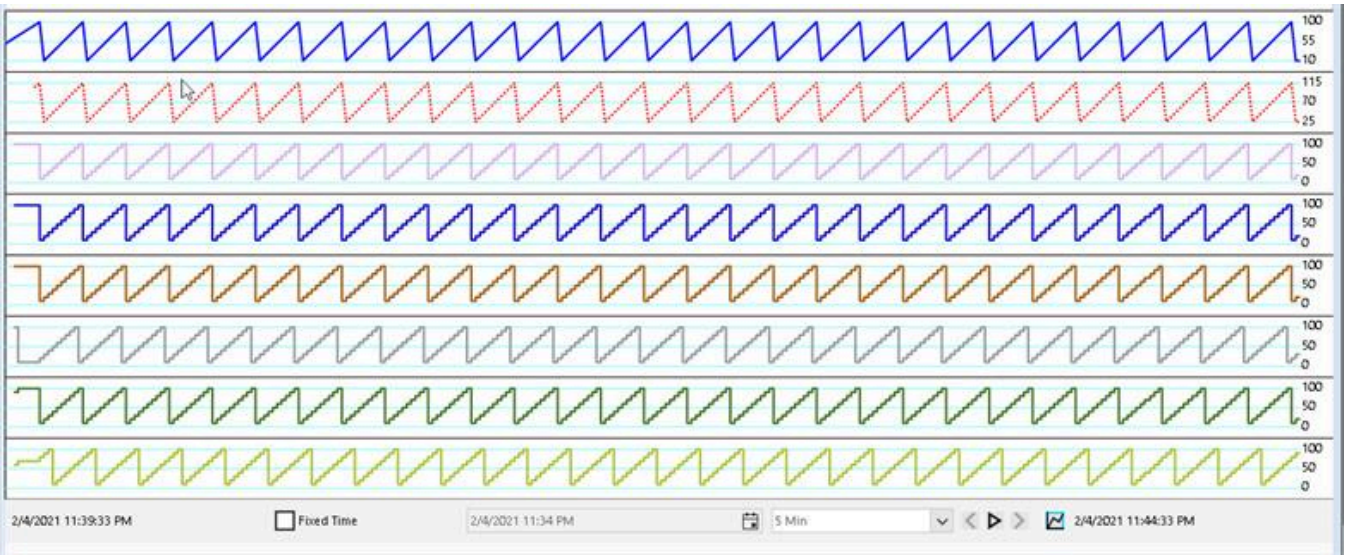
The scale of the pen in the trend area will be updated, when the pen row is selected.

Stack and unstack trends

In runtime, you can use the Stack/Unstack option to view the individual pens. By default, the pens are displayed together.



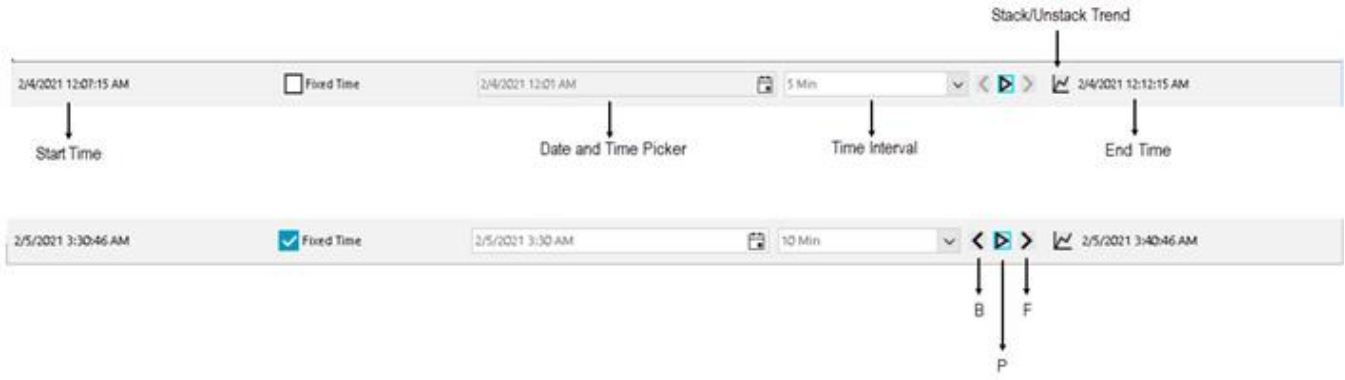
To display the pens individually, click the **Stack the trends** button in the toolbar. The pens are shown individually stacked one top of the other. Select again, to return to the unstacked view.



Note: If pens are added in runtime, the individual scales will overlap when the trend area is small. To view results more clearly, you can hide some pens.

Play back the Trend Data

Using the playback options in the toolbar you can traverse the trend data between a specified start and end time.



Moving Mode

By default, the Trend area will display the current trend data for the value specified in the Time Interval field. The relation of Start Time and End Time is End Time = Start Time + Time Interval. In Moving Mode, the End Time is always current time and will change according to the input. In the Fixed Mode, the Start Time and End Time are frozen until the playback controls are used.

You can select the Time Interval from a list of predefined values. The Trend area will refresh the trend pen data and only show data for that time period.

Fixed Mode

1. Select the **Fixed Time** checkbox if you want to view trend pen for a particular period in the past or future.
2. Select the **Time Interval**.

   The Trend Area will reflect the pen data for the time interval selected between the Start Time and End Time.
3. Use the **Move the trend backwards** (B), **Pause the trends/resume the trends** (P) and **Move the trends forward** (F).

   The Pause button freezes the trend area for the selected time interval.

The < and > will move the trend area for half of the Time Interval within the block of time between the Start Time and End Time.

For example: Time Interval = 10 min

Mode	Start Time	End Time
Live	2/8/2021 3:15:16 AM	2/8/2021 3:25:16 AM
Fixed: Move Backward	2/8/2021 3:10:16 AM	2/8/2021 3:20:16 AM
Fixed: Move Forward	2/8/2021 3:20:16 AM	2/8/2021 3:30:16 AM

Using the Date and Time Picker

You can select a particular Start date and time using the Date and Time Picker. This will only work when the Fixed Time checkbox is selected.

The Start Time will display the date and time selected. The End Time will display the selected date and time + the time interval.

The Trend Area will display the data for the Date and Time selected + the Time Interval.

About SmartSymbols

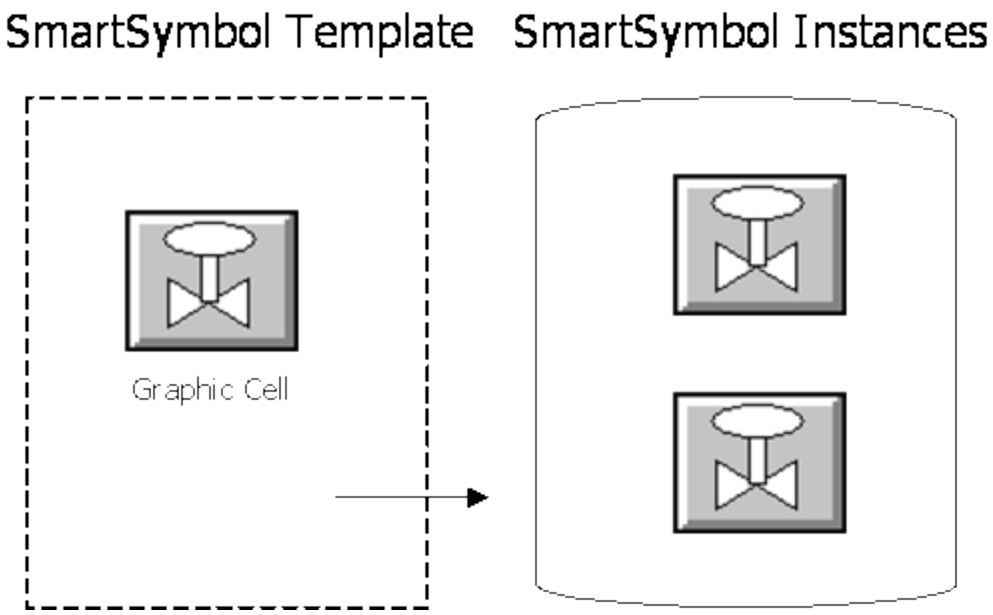
SmartSymbols are InTouch graphics that are converted into reusable templates. Changes that you make to a SmartSymbol template propagate to all instances of the SmartSymbol throughout the application. This relieves you of the duplicate effort for creating, modifying, validating, and re-validating graphics used repetitively throughout an application.

SmartSymbols can reference InTouch tags and application server attributes. A SmartSymbol that references no tags or only InTouch tags is called an InTouch SmartSymbol.

A SmartSymbol that references an attribute of one or more Automation Objects or Automation Object instances is called an Industrial Graphics SmartSymbol. Industrial Graphics SmartSymbols may also reference InTouch tags.

InTouch and Industrial Graphics SmartSymbols are different from Industrial Graphics. Industrial Graphics are designed and managed using the System Platform Integrated Development Environment (IDE).

If you are using InTouch version 10 or later and are creating new graphics, Industrial Graphics may be more suitable than SmartSymbols for your application.



SmartSymbol Manager and Library

The SmartSymbol library contains the SmartSymbols for an InTouch application. You use the SmartSymbol Manager to import, export, and organize the contents of the SmartSymbol library.

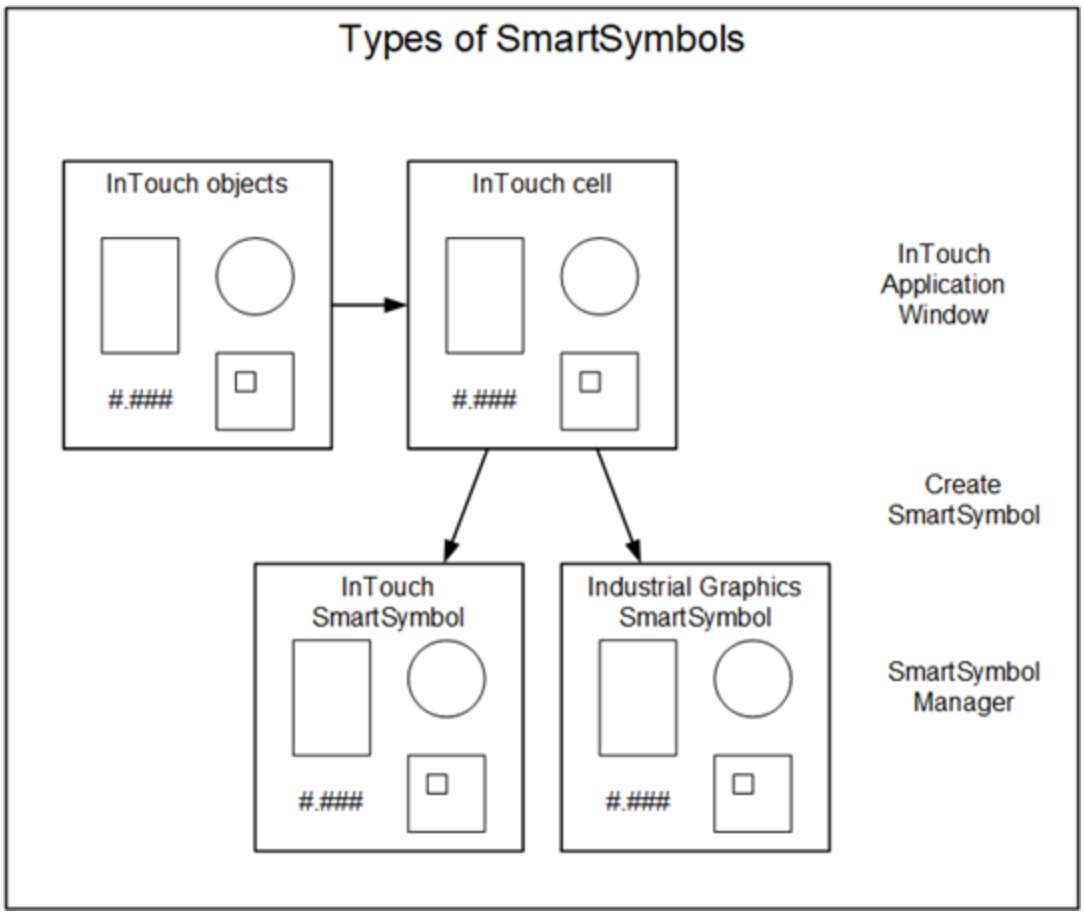
SmartSymbols are stored in a \Symbols folder of the application folder of the user's InTouch application.

Reference information about the SmartSymbols in the library is stored in an XML file. You should not edit the XML file.

InTouch SmartSymbols and Industrial Graphics SmartSymbols

InTouch SmartSymbols are not the same as Industrial Graphics SmartSymbols. Industrial Graphics supersede Industrial Graphics SmartSymbols and are developed using the System Platform IDE. InTouch SmartSymbols reference InTouch tag data. Industrial Graphics SmartSymbols reference Galaxy object or template attributes.

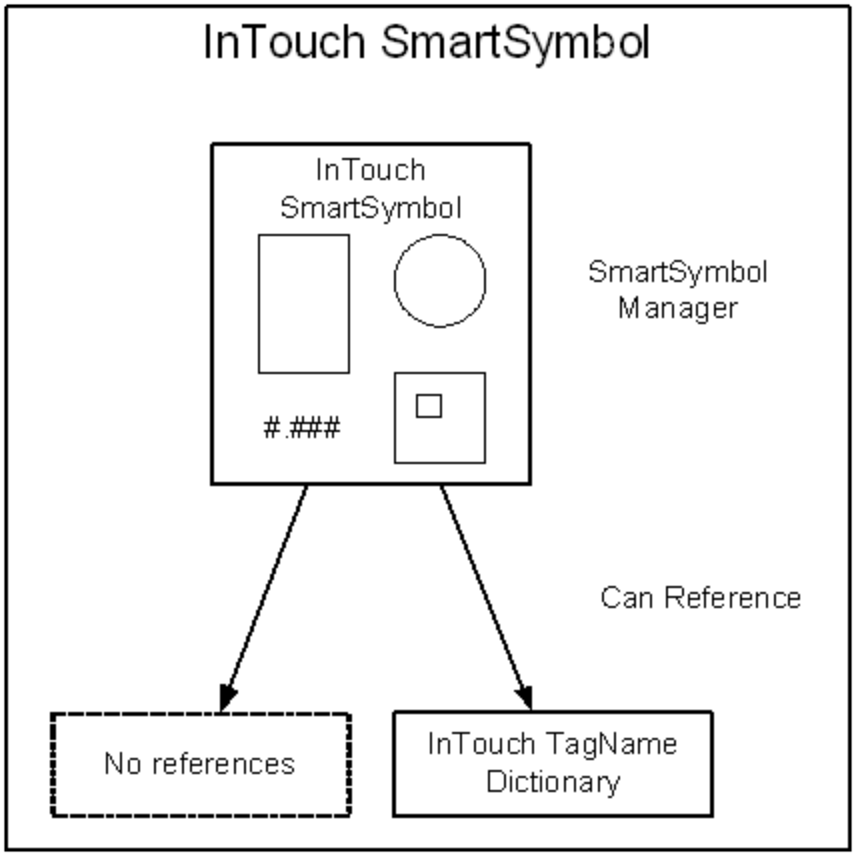
The following figure shows the types of SmartSymbols.



InTouch SmartSymbols

InTouch SmartSymbols are stored under the InTouch Symbols folder in the SmartSymbol Manager.

You can configure animation for the graphical elements in an InTouch SmartSymbol using references to local and remote InTouch tags. For more information, see Create SmartSymbol templates and instances.

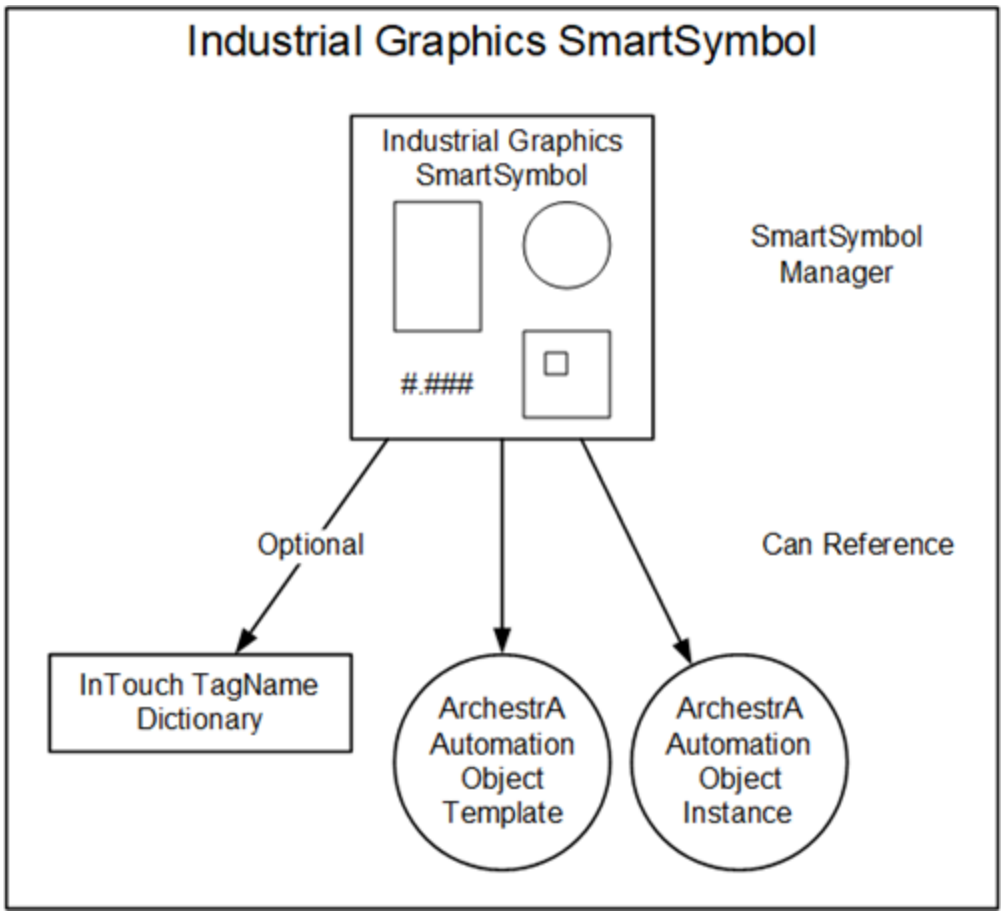


Industrial Graphics SmartSymbols

Industrial Graphics SmartSymbols are stored under the Industrial Graphics folder in the SmartSymbol Manager.

Note: Even though the folder is labeled Industrial Graphics, the folder contains Industrial Graphics SmartSymbols, not Industrial Graphics.

To create Industrial Graphics SmartSymbols, you select one or more Galaxy object templates to define the animation references for the various graphical elements. For more information, see Create SmartSymbol templates and instances.



Limitations of SmartSymbols

The following are known limitations of SmartSymbols.

- SmartSymbols cannot contain a trend object. If you try to create a SmartSymbol that contains a trend object (historical or real-time), an error message appears.
- SmartSymbols cannot contain Distributed Alarm Display controls, Windows Controls, InTouch ActiveX controls like AlarmViewer, or third party ActiveX controls that are configured in an InTouch application.
- You cannot browse for instances in a galaxy created with Application Server version 1.5. Install Application Server version 2.0 or later to browse for instances.
- Generating SmartSymbols with an SPC Chart wizard is not supported.
- SmartSymbols cannot reference local script variables.
- The **Attribute Browser** does not show derived object instances. To address this issue, create the derived template in the SmartSymbol Manager or create a custom filter in the browser.

Create SmartSymbol templates and instances

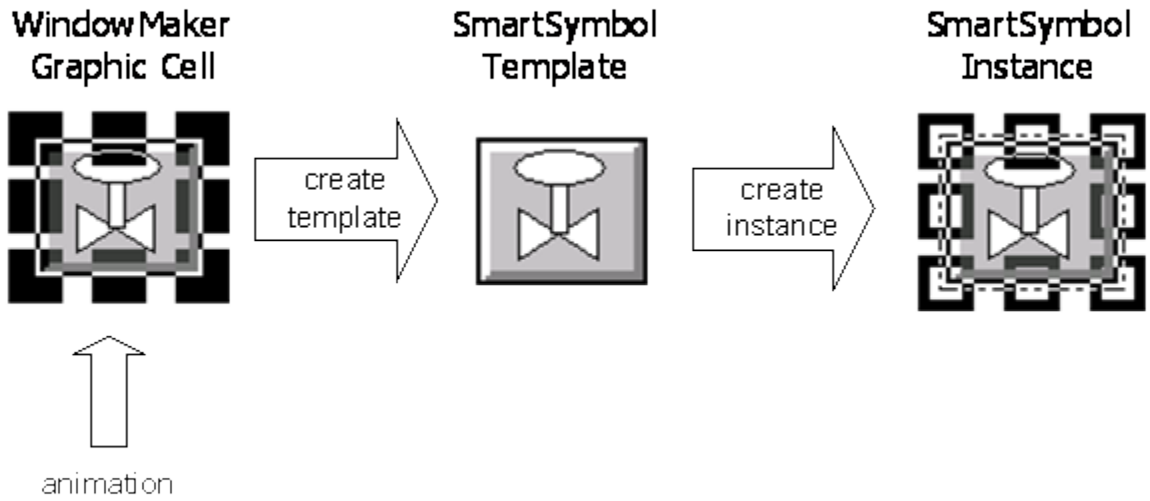
Using WindowMaker, you can create SmartSymbol templates and instances.

You create SmartSymbol templates by drawing one or more graphics in WindowMaker, combining them into a

cell, and then converting the cell into a SmartSymbol. You do not have to connect or link templates to InTouch tags or application server objects.

After you create a SmartSymbol template, you can create an instance of the SmartSymbol in an application window.

You can also create application server object instances from an existing Industrial Graphics SmartSymbol instance to avoid switching between InTouch WindowMaker and the System Platform IDE.



Create SmartSymbol templates for use with InTouch data

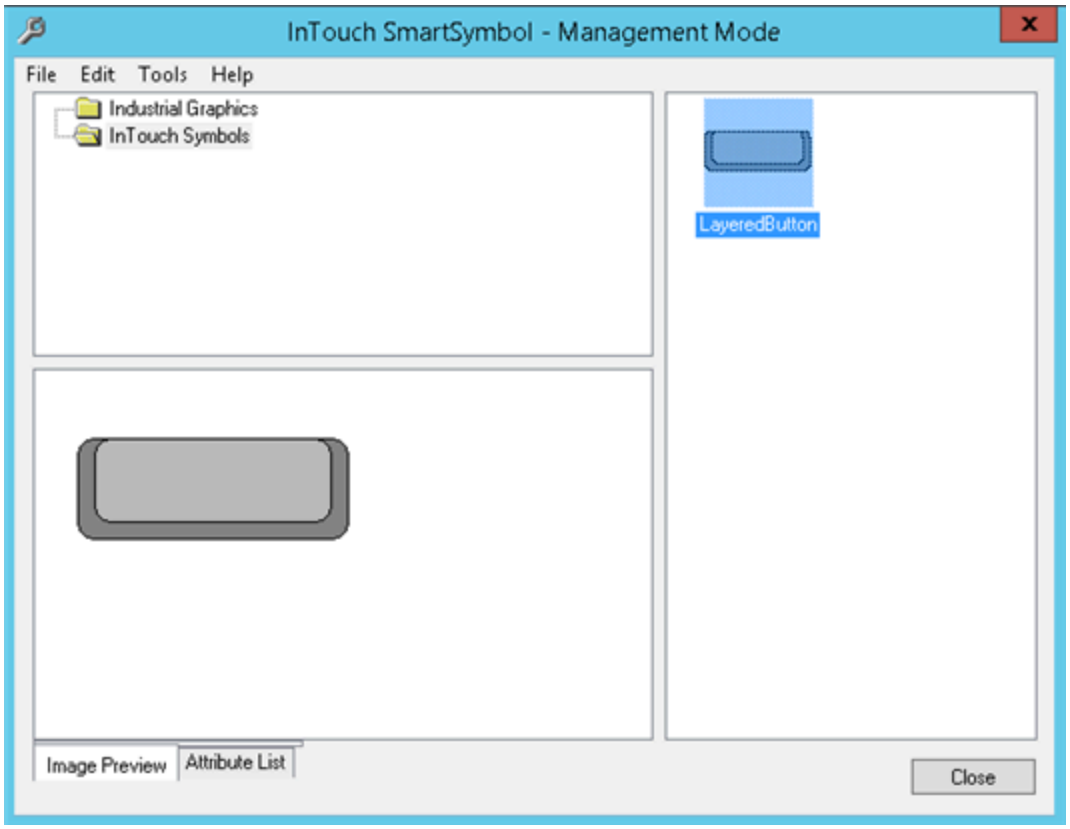
InTouch SmartSymbols are created from cells, which can contain graphical elements, animation, and references to InTouch tags.

The following procedure specifies that a new window, graphics, and cell be created, but you can use an existing window, graphics, or cells to create a SmartSymbol.

Create a new InTouch SmartSymbol template

1. Create a new window in WindowMaker.
2. Using the graphics drawing tools and/or Wizards, create a graphic or set of graphics that you want to include in your SmartSymbol.
3. Configure the animation link(s) for the graphic(s).
   For instructions, see Animate objects.
4. Select all objects to be included in the SmartSymbol template.
5. On the **Animation** menu, in the **Cell** group, select **Make Cell**.
6. Select the cell you just made.
7. On the **Draw** menu, in the **SmartSymbols** group, select **Generate**.

   The **InTouch SmartSymbol - Management Mode** dialog box appears with the new SmartSymbol highlighted.



By default, the new SmartSymbol is placed in the InTouch Symbols top level folder. A default name is automatically assigned to the symbol (for example, New Symbol1).

8. Type a new name or accept the default. You can change the name of the SmartSymbol at any time. For more information on renaming SmartSymbols, see Rename SmartSymbol templates.
9. Select **Close**. A message appears prompting you to replace the graphic cell with new SmartSymbol. Select **Yes** or **No**. If you select **Yes**, the graphic cell is replaced by the SmartSymbol. If you select **No**, the graphic cell is unchanged. In either case, the new SmartSymbol is stored in the SmartSymbol library and is available for future use.

Create Industrial Graphics SmartSymbol templates

When you create a SmartSymbol from an InTouch graphic cell that contains at least one reference to an automation object template or instance, you end up with an Industrial Graphic SmartSymbol.

A reference to an automation template contains a "$" sign.

You can generate SmartSymbol templates that are associated with object templates and/or instances.

When you create a SmartSymbol instance on the InTouch window, you can instantiate its referenced object template.

Generate a new Industrial Graphic SmartSymbol template

1. Create a new window in WindowMaker.
2. Using the drawing tools and/or Wizards, create graphics that you want to make into a SmartSymbol.
3. Configure the animation link(s) for the graphic(s).

   For instructions on configuring Galaxy source names, see Access Application Server data from InTouch. For

instructions on configuring creating links to application server attributes, see Animate objects.

4. Select the graphics to make into a cell.
5. On the **Animation** menu, in the **Cell** group, select **Make Cell**.
6. On the **Draw** menu, in the **SmartSymbols** group, select **Generate**.

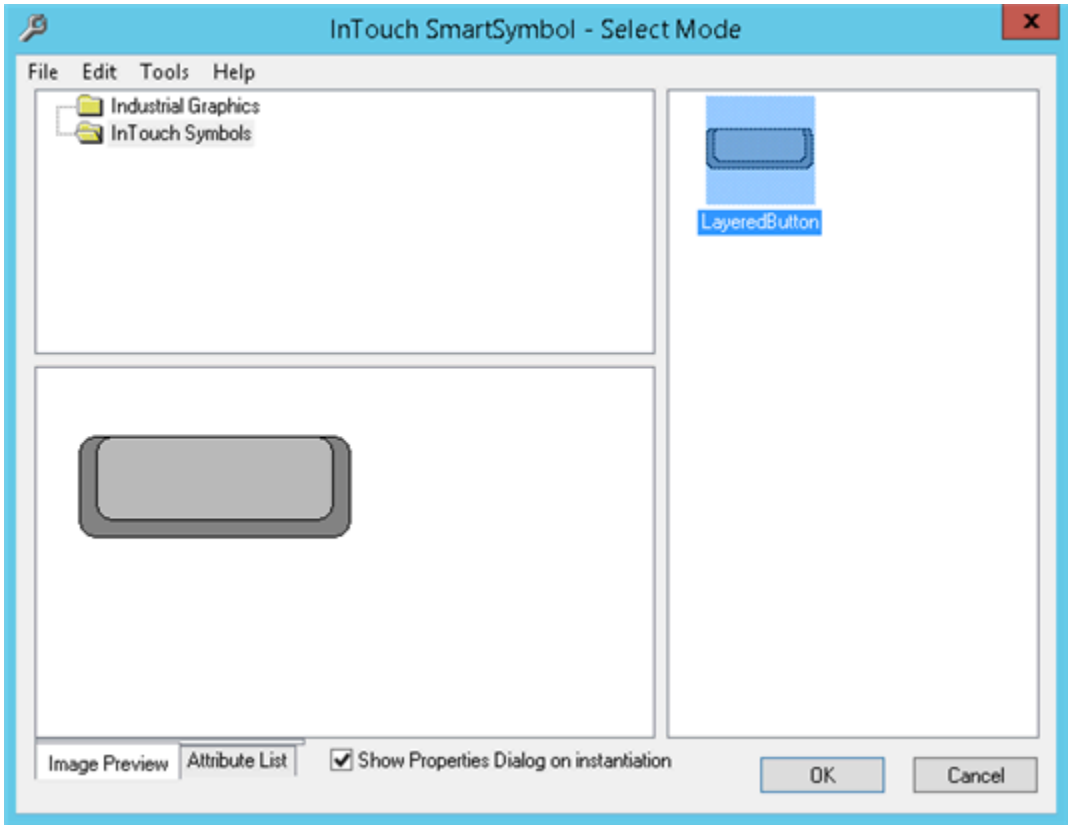
   The SmartSymbol Manager creates the new SmartSymbol for Galaxy data.
7. Type in a new name or accept the default name and change it later.
8. Select **Close**. A message appears prompting you to replace the graphic cell with new SmartSymbol. Select **Yes** or **No**. If you select **Yes**, the graphic cell is replaced by the SmartSymbol. If you select **No**, the graphic cell is unchanged. In either case, the new SmartSymbol is stored in the SmartSymbol library and is available for future use.

Create SmartSymbol instances from InTouch SmartSymbol templates

You can create multiple SmartSymbol instances from a single SmartSymbol template. Each instance inherits all references and text labels. Before the instance is placed on the InTouch window, you can change the references and text labels.

Create a SmartSymbol from an InTouch SmartSymbol template

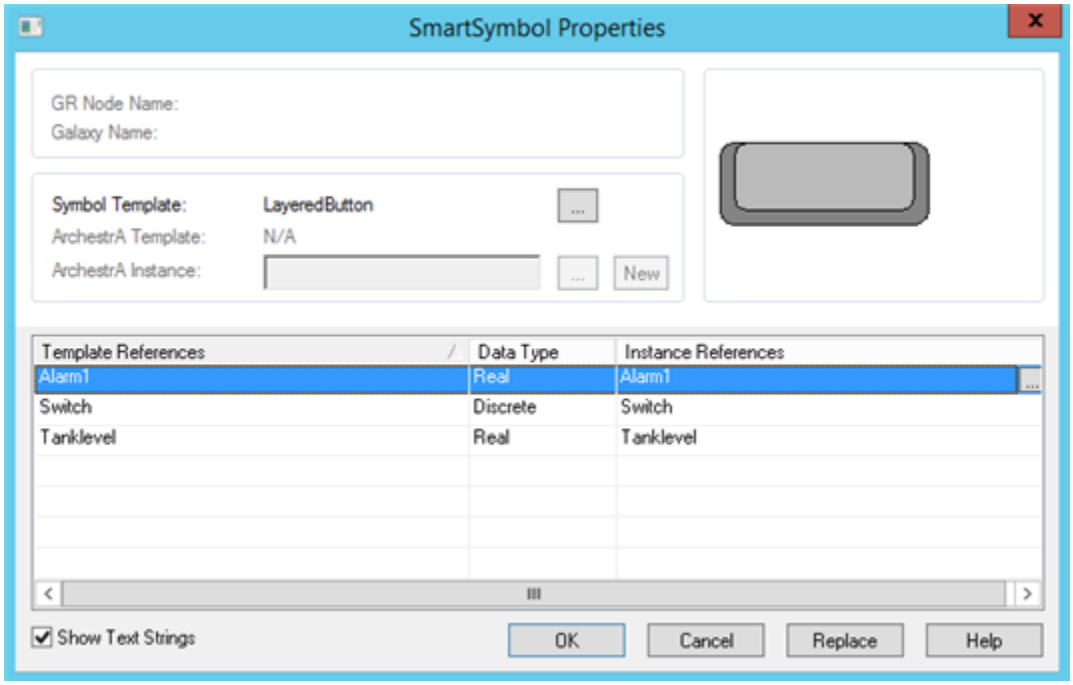
1. Open WindowMaker and open a window where you want to use a SmartSymbol.
2. On the **Home** menu, in the **Insert** group, select the **SmartSymbol** icon.
3. Select the location within WindowMaker window where you want to place the symbol. The **InTouch SmartSymbol - Select Mode** dialog box appears.



Note: By default, the **Show Properties Dialog on instantiation** checkbox is selected. Clear the checkbox if you don't want to change any references or text labels for the new SmartSymbol instance.

4. In the **InTouch Symbols** folder, double-click the SmartSymbol. The new symbol appears in the application window.

   If the **Show Properties Dialog on instantiation** checkbox is selected in the previous step, the **SmartSymbol Properties** dialog box appears.



5. In the **Instance References** column, select the **Ellipsis** button. The **Select Tag** or **Tagname Dictionary** dialog box appears.
6. Select the tag to link to the SmartSymbol. Close the window and the **SmartSymbol Properties** dialog box appears.

   Note: If you enter a new tagname that is not defined yet, then the **Tagname Undefined** dialog box appears, select **OK** and define a new tag from the Tagname Dictionary.

7. Select **OK**. The new symbol appears in the application window.

Create SmartSymbol instances from ArchestrA SmartSymbol templates

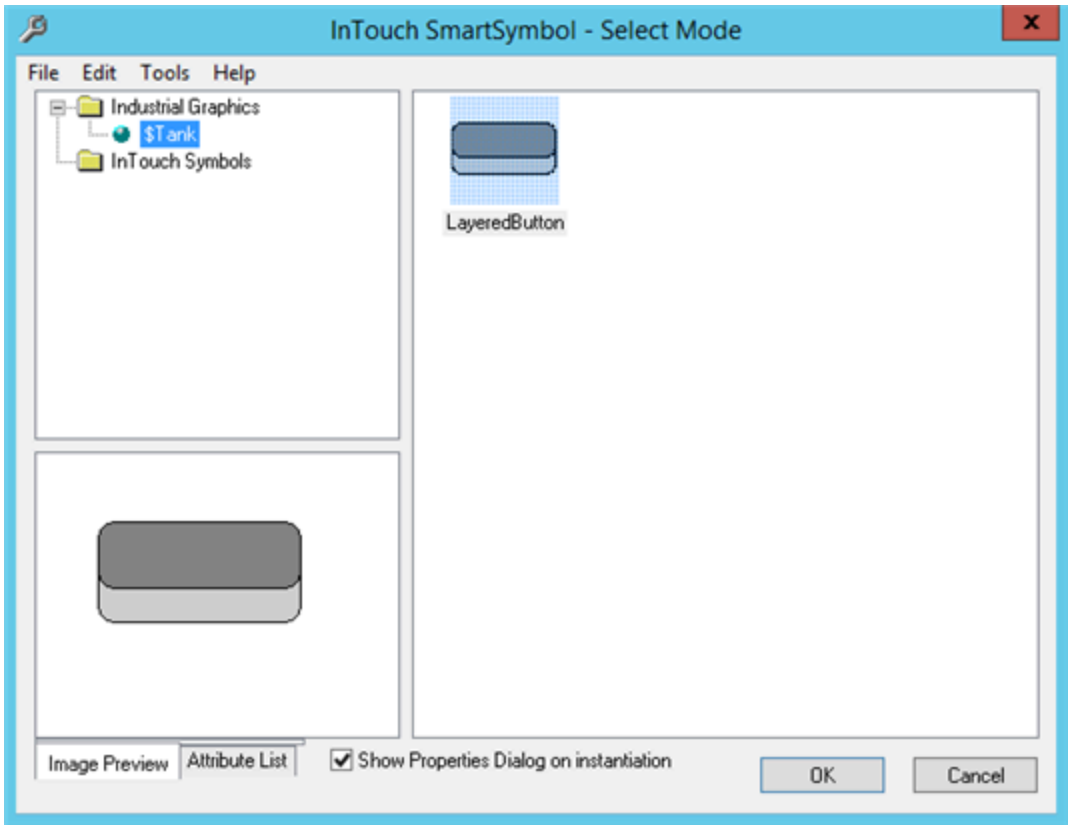
You can create multiple ArchestrA SmartSymbol instances from a single ArchestrA SmartSymbol template.

Create a SmartSymbol from an ArchestrA SmartSymbol template

1. Select the **SmartSymbol** icon and select in the WindowMaker window where you want to place the symbol. The **InTouch SmartSymbol - Select Mode** dialog box appears.

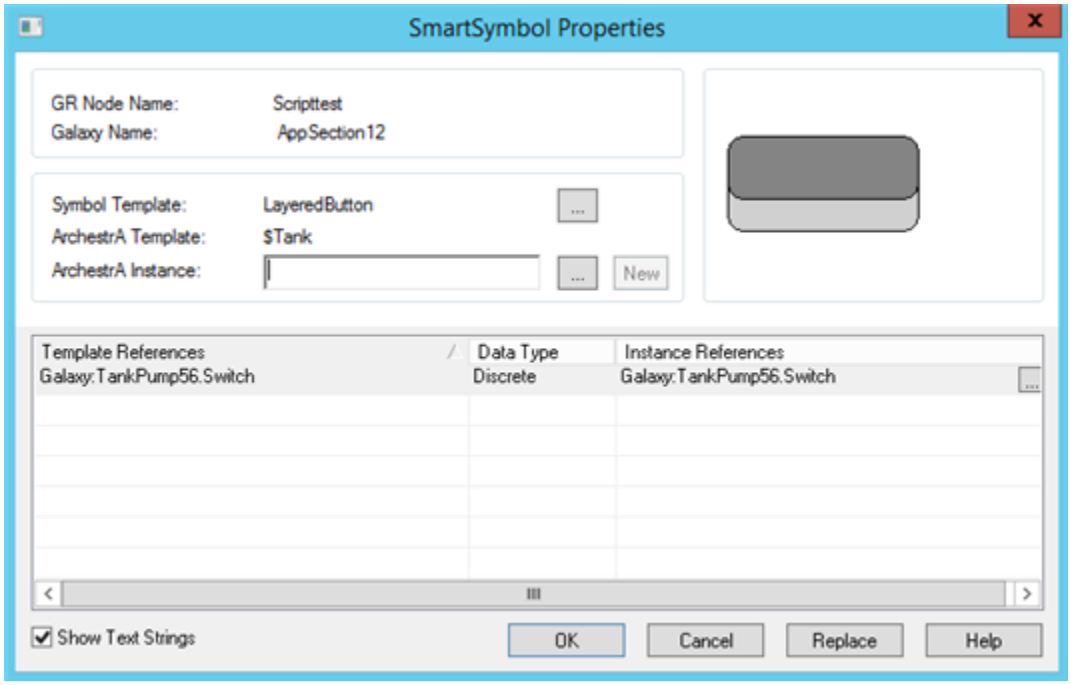
   Note: By default, the **Show Properties Dialog on instantiation** checkbox is selected.

2. Select the **Industrial Graphics** folder. The Industrial Graphics appear in the right pane.



3. Select the SmartSymbol and select **OK**. The new symbol appears in the application window.

   If the **Show Properties Dialog on instantiation** checkbox is selected in the previous step, the **SmartSymbol Properties** dialog box appears.



4. In the **ArchestrA Instance** text box, you can either:
   - Browse for and select an ArchestrA object.

- Create a new ArchestrA object instance derived from associated object template. Enter a name for the instance and select **New**.

  The instance attribute references appear in the **Instance References** column.

  Note: If you did not connect the Galaxy yet, a dialog box appears prompting you to enter node name and Galaxy name.

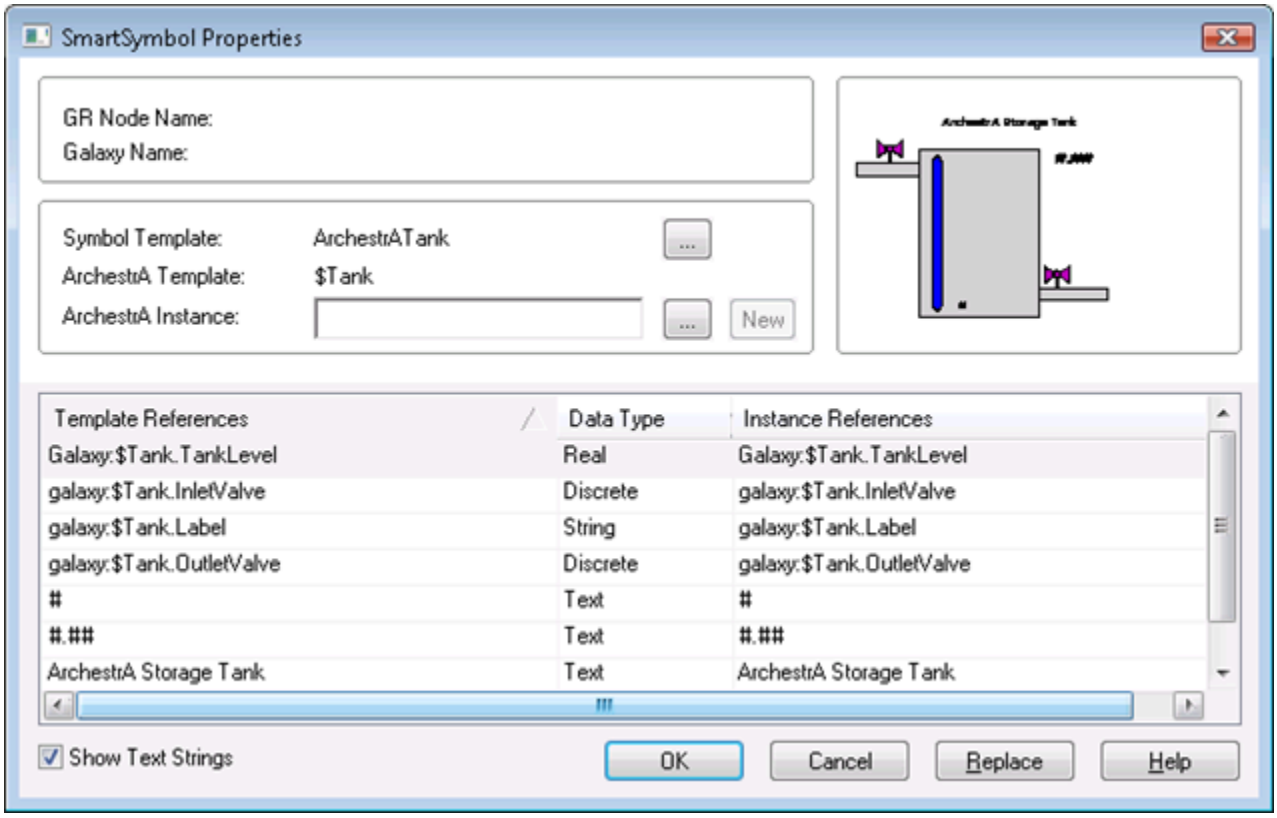
5. In the **Instance References** column, change the references if needed. You can manually type in the references with correct syntax or select the **Ellipsis** button to use the **Attribute Browser**.
6. Select **OK**. The new symbol appears in the window.

Create an ArchestrA object instance from an ArchestrA SmartSymbol instance

You can create a new ArchestrA object instance from an existing ArchestrA SmartSymbol instance. By doing this, you do not need to switch between WindowMaker and the IDE.

Create a new ArchestrA object instance

1. In WindowMaker, open the window in which the SmartSymbol instance is located.
2. Double-click the SmartSymbol instance in the application window. The **SmartSymbol Properties** dialog box appears.

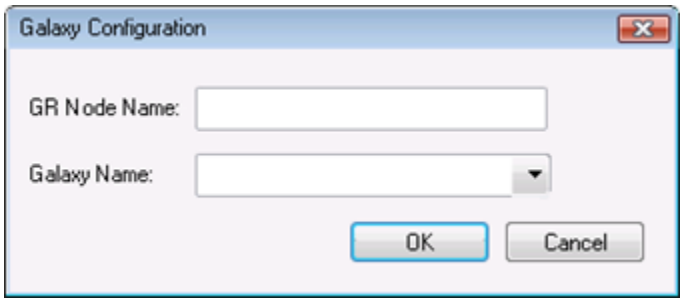


3. In the **ArchestrA Instance** box, type a valid name for the new Automation object.

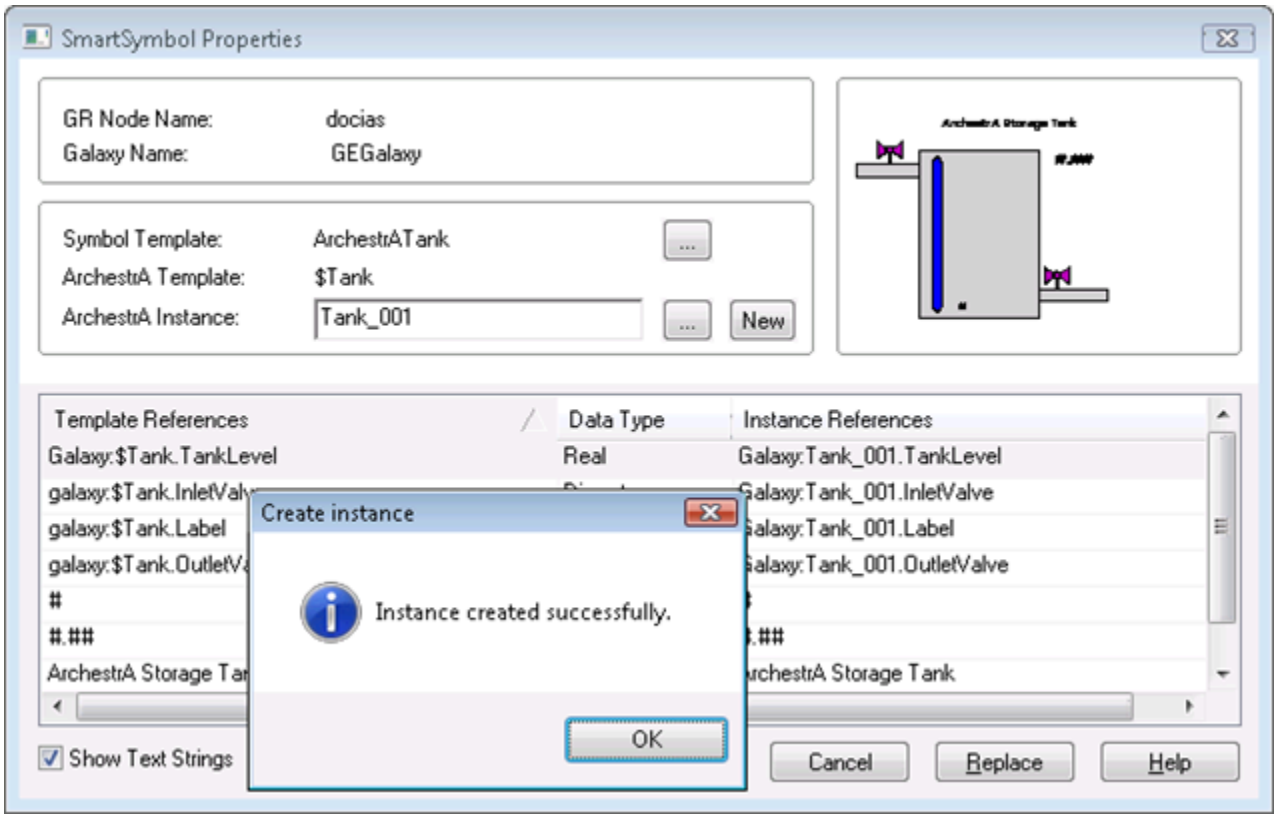
   Note: If this is the first time you specify an object, you are prompted to log in. Provide a valid user name, password, and domain name. If the Application Server security is set to a mode other than None, a domain name is required only for OS User or OS Group Based security.

4. Select **New**. When the message appears prompting you to select a valid Galaxy in which to create the new

object, select **OK**. The **Galaxy Configuration** dialog box appears.



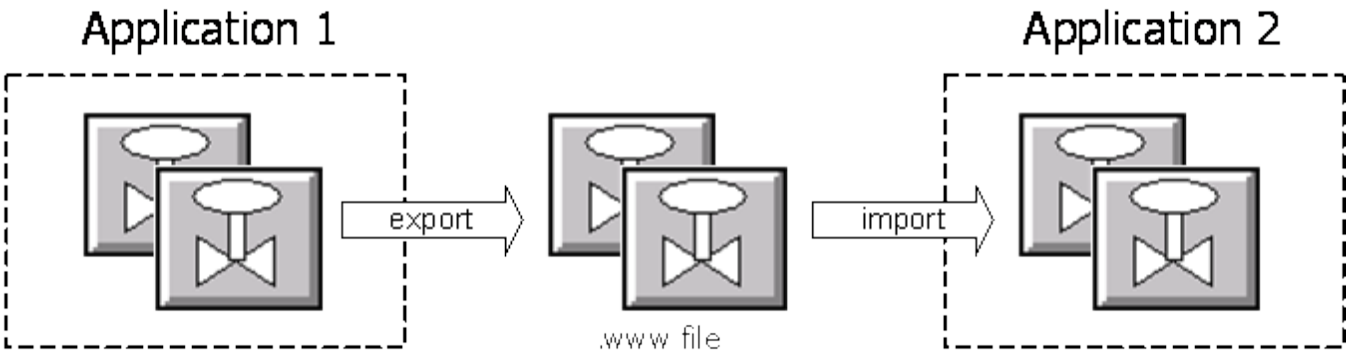
5. Specify the Galaxy. Do the following:
   a. In the **GR Node Name** box, type the name of the computer that the Galaxy is running on.
   b. In the **Galaxy Name** list, select the Galaxy.
   c. Select **OK**. The ArchestrA object instance is created and the instance references point to the new instance.



   a. Select **OK** again to close the **Create Instance** dialog box.
6. Select **OK** to close the **SmartSymbol Properties** dialog box. The new SmartSymbol instance appears in the application window.

Manage SmartSymbols

Using the SmartSymbol Manager, you can import and export SmartSymbols among multiple InTouch applications and across different physical systems. Exporting and importing SmartSymbols is the best way to move SmartSymbols between InTouch applications.



You can also import windows with SmartSymbols and the graphics will be imported—but not the template information—resulting in orphaned instances of SmartSymbols. For more information, see Recover SmartSymbols.

With the SmartSymbol Manager, you can rename, duplicate, delete, and save SmartSymbol templates.

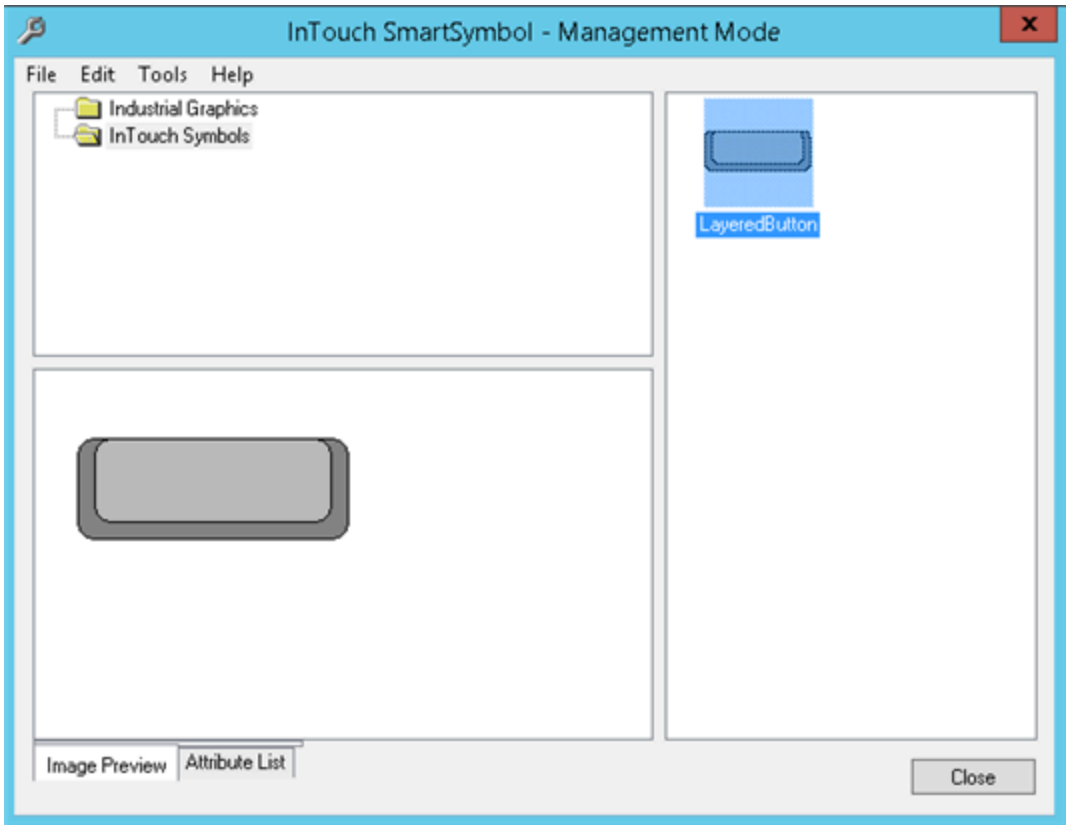
Import SmartSymbols

You can import SmartSymbols from other InTouch applications into your application's SmartSymbol library. Importing symbols from other applications allows you to reuse symbols instead of creating those symbols again.

Import SmartSymbols into the SmartSymbol library

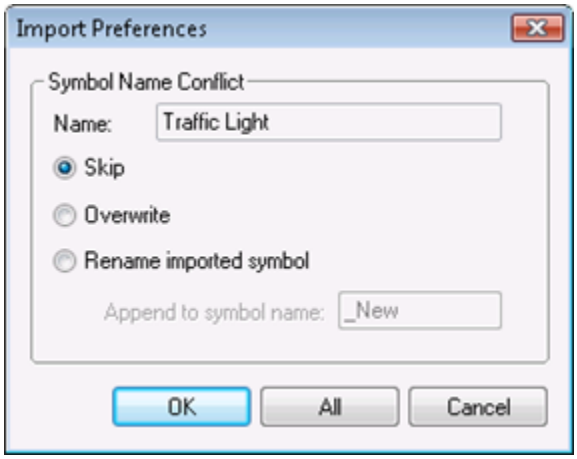
1. Close all application windows.
2. On the **Draw** menu, in the **SmartSymbol** group, select **Manage**.

   The **InTouch SmartSymbol - Management Mode** window appears.



3. On the **File** menu, select **Import**. The **Import Symbol** dialog box appears.
4. Browse for the file that contains the SmartSymbols to import. Symbol export files have a .www file extension.
5. Select the file, and select **OK**. The SmartSymbols in that file appear in the SmartSymbol Management Mode window.

   If there is a name conflict, the **Import Preferences** dialog box appears.



6. Do one or more of the following:
   - To skip the import of this symbol, select **Skip**. If you are importing multiple symbols, the rest of the symbols are imported.
   - To overwrite the existing symbol with the new one, select **Overwrite**.
   - To rename the new symbol with an unused name, select **Rename imported symbol**. In the **Append to**

symbol name box, type the name.

7. Do one of the following:
   - Select **OK** to apply the selected option to the SmartSymbol.
   - If you selected **Rename imported symbol**, select **All** to apply the text in the **Append to symbol name** box to all SmartSymbols with name conflicts in the package file.

     The imported SmartSymbol(s) appear in the InTouch **SmartSymbol Management Mode** window.

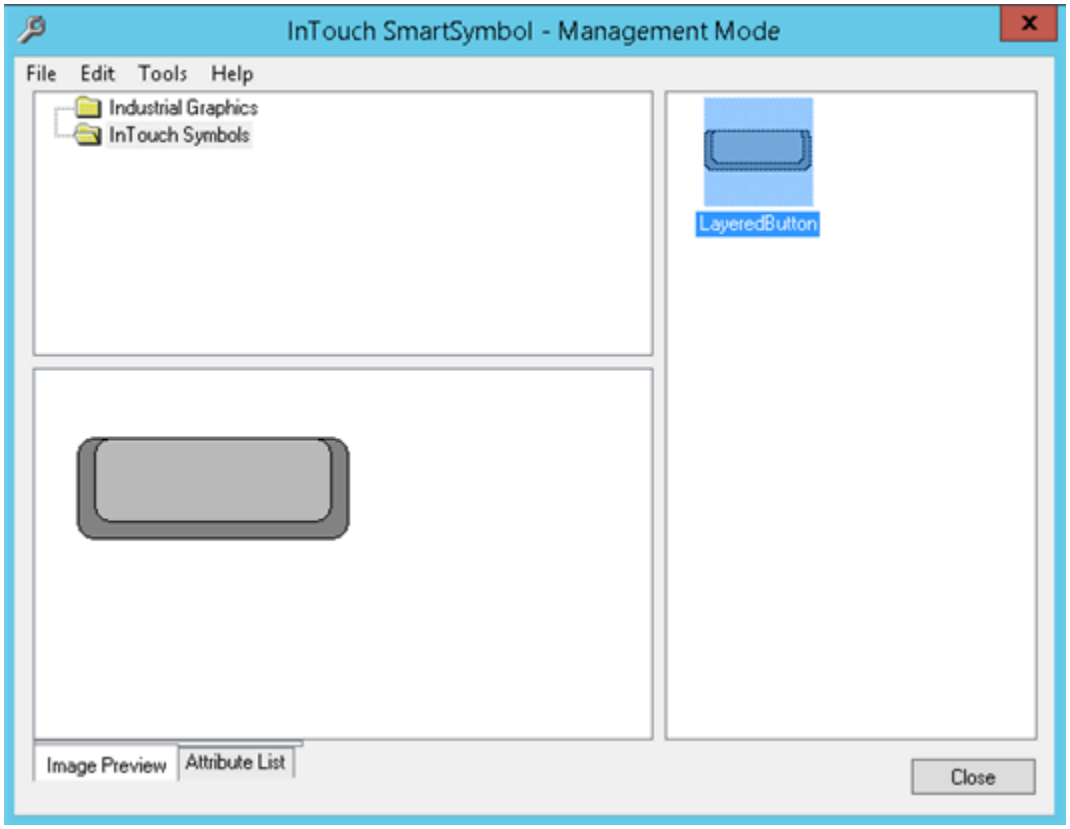
Export SmartSymbols

After you create or import SmartSymbols in your application SmartSymbol library, you can export one or more SmartSymbol templates to other InTouch applications. Exporting SmartSymbol templates is the recommended way to move SmartSymbols between InTouch applications.

Export a SmartSymbol

1. On the **Draw** menu, in the **SmartSymbol** group, select **Manage**.

   The **InTouch SmartSymbol - Management Mode** window appears.



2. From the list of SmartSymbols and folders, select the SmartSymbol(s) or folder(s) that you want to export.
3. On the **File** menu, select **Export**. The **Export Symbol** dialog box appears.
4. Browse to the folder to export the symbol to.
5. Type in a file name, with a .www extension, and select **Save**. The SmartSymbol(s) and/or folder(s) is(are) exported to the folder you specified.

Rename SmartSymbol templates

Using the SmartSymbol Manager, you can rename SmartSymbol templates. Renaming a SmartSymbol template has no impact on any SmartSymbol instances.

Rename a SmartSymbol template

1. In the SmartSymbol Manager, select the SmartSymbol template you want to rename.
2. On the **Edit** menu, select **Rename**.
3. Type in a new name for the symbol and then press **Enter**. The SmartSymbol template appears with the new name.

Duplicate SmartSymbol templates

After you create a SmartSymbol template, you can create a copy of it. For example, you can duplicate a template and modify and edit it to be a new template with similar features.

For more information on editing SmartSymbol templates, see Editing SmartSymbols.

Duplicate a SmartSymbol template

1. In the SmartSymbol Manager, select the SmartSymbol that you want to duplicate.
2. On the **Edit** menu, select **Copy**.
3. Select the folder for the new SmartSymbol.
4. On the **Edit** menu, select **Paste**. The new SmartSymbol appears. If placed in the same folder as the original, the new SmartSymbol is named Copy of <original name>.

Delete SmartSymbol templates

If you delete a SmartSymbol template, you can no longer open, edit, or view the properties of SmartSymbol instances based on the template. The run time state of those SmartSymbol instances is not affected by the deletion.

You can recover a deleted SmartSymbol from its instance. For more information, see Recover SmartSymbols.

Delete a SmartSymbol template

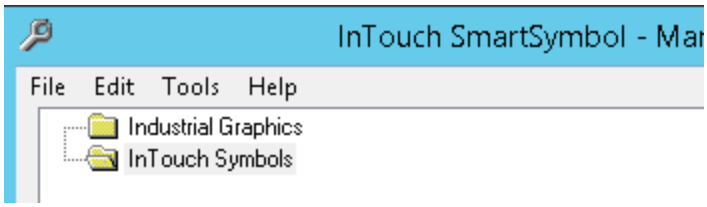
1. In the SmartSymbol Manager, select the SmartSymbol to delete.
2. On the **File** menu, select **Delete**. When the message appears, select **Yes**.

   The SmartSymbol template is deleted from the SmartSymbol library. All instances of this SmartSymbol become orphaned.

Save SmartSymbols in a folder hierarchy

SmartSymbols are stored in the SmartSymbol library in a standard hierarchical folder structure. You can see two standard folders are included to simplify the organization of SmartSymbols in the SmartSymbol library:

- A top-level folder for Industrial Graphics SmartSymbol templates
- A top-level folder for InTouch SmartSymbol templates



You can create sub-folders for templates using the SmartSymbol Manager. Store Industrial Graphics SmartSymbol templates in the template folder with which they should be associated with when they are created. For example, if you create a SmartSymbol to use with $Valve objects, store the symbol template in the "$Valve" template folder.

You cannot drag Industrial Graphics SmartSymbols into an InTouch Symbols folder, and you cannot drag InTouch SmartSymbols into an Industrial Graphics Symbols folder.

Move a SmartSymbol to a different folder

1. Select the SmartSymbol you want to move.
2. Drag the SmartSymbol into the new folder.

Support for SmartSymbols and language switching

Language switching works for SmartSymbols if the SmartSymbol template exists in the application.

If a SmartSymbol contains translatable text objects, when you export the dictionary a separate XML is generated, for example SSD_<SymbolName>_<LangID>_<ID>.xml. This XML file contains all translatable strings contained in the SmartSymbol. You can open it in Excel and translate the text strings like you would for any InTouch application.

When you import the translation for an InTouch application, the translations for each SmartSymbol are imported as well.

When you switch languages in WindowViewer, any SmartSymbols containing translatable strings that are translated in this way appear translated.

When exporting SmartSymbols that have dictionary files, the dictionary files are exported along with the .www file. For more information about language switching, see Switch a language at runtime in *AVEVA™ InTouch HMI Application Run Time*.

Recover SmartSymbols

When you delete a SmartSymbol template from the library, all instances of that SmartSymbol are considered "orphaned" instances. You can recover a deleted SmartSymbol from an orphaned instance. If an orphaned instance does not exist in an application window, you cannot recover the SmartSymbol.

If you try to open the properties of an instance after the SmartSymbol template is deleted, a warning message appears telling you that the SmartSymbol no longer exists in the library.

You can also have orphaned instances if you import a window containing SmartSymbols. You must recover the SmartSymbol from the orphaned instance and then rename the SmartSymbol.

Recover a deleted SmartSymbol

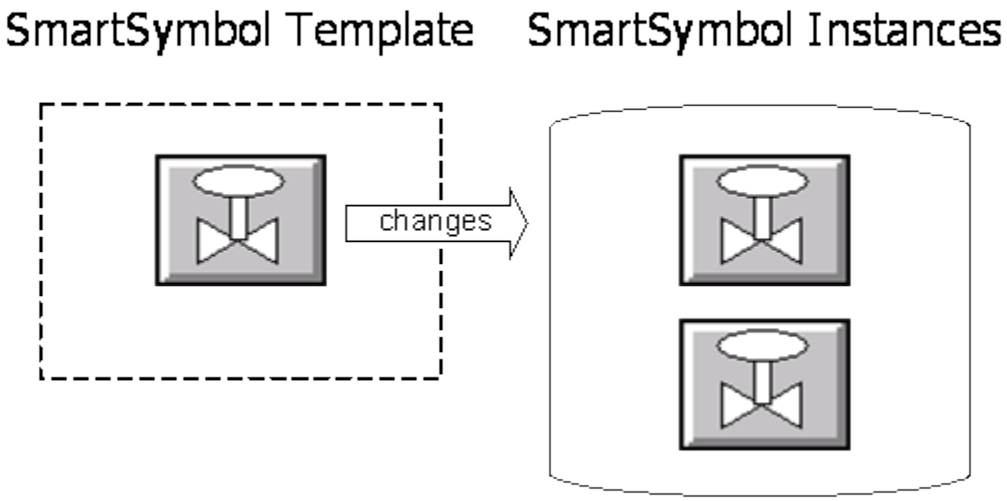
1. Select an orphaned instance of the deleted SmartSymbol in an InTouch HMI application window.

2. On the **Draw** menu, in the **SmartSymbol** group, select **Recover**.

   The SmartSymbol appears in the **SmartSymbol Management Mode** window, with a name of **New Symbol**.
3. Rename the SmartSymbol as appropriate.

Edit SmartSymbols

After you create a SmartSymbol, you can edit it by changing and modifying the template or an instance of the SmartSymbol.



Change SmartSymbol templates

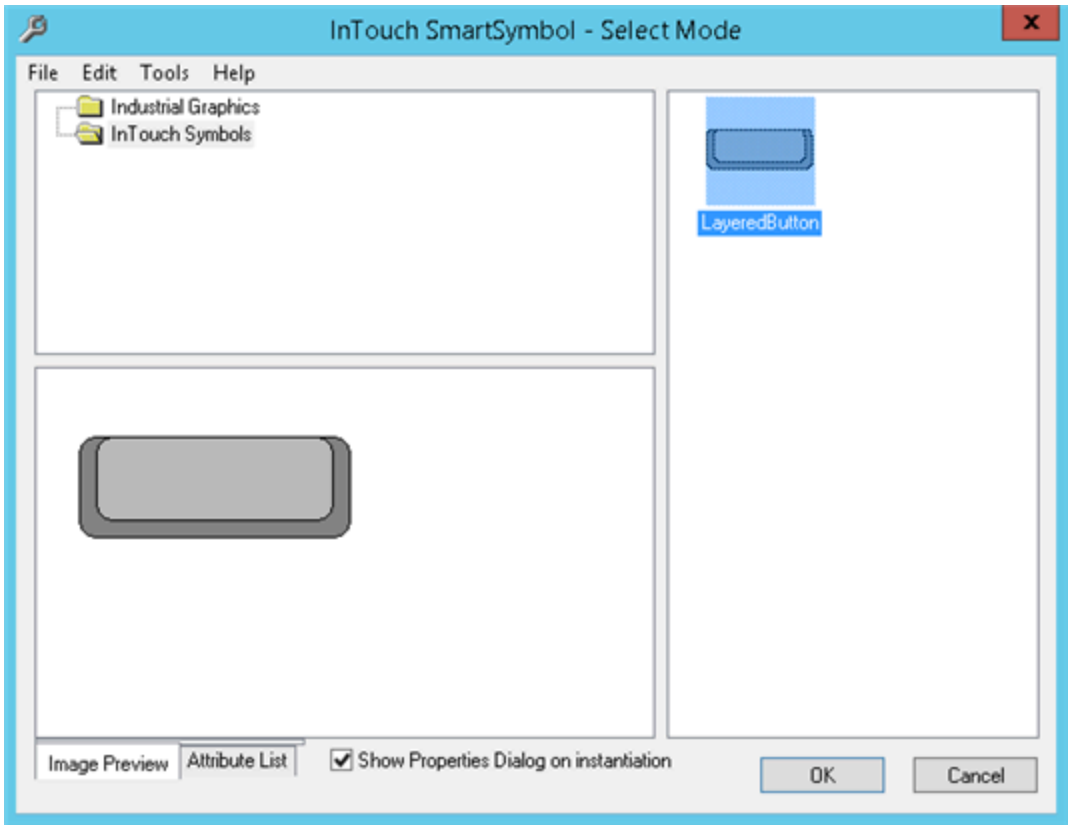
To edit a SmartSymbol, break the cell and then use the drawing tools to make changes. You can also change the animation that is associated with the SmartSymbol. Template changes affect all instances of the SmartSymbol.

Note: Edit SmartSymbols in a temporary window rather than in an application window.

Edit an existing SmartSymbol template

1. On the **Draw** menu, in the **SmartSymbol** group, select **Start Edit**.
2. Select the location within the window where you will edit the SmartSymbol.

   The InTouch **SmartSymbol - Select Mode** window appears.



3. Select a SmartSymbol to edit and select **OK**.

   An instance of the SmartSymbol is placed in the application window.

4. On the **Animation** menu, in the **Cell** group, select **Break**.

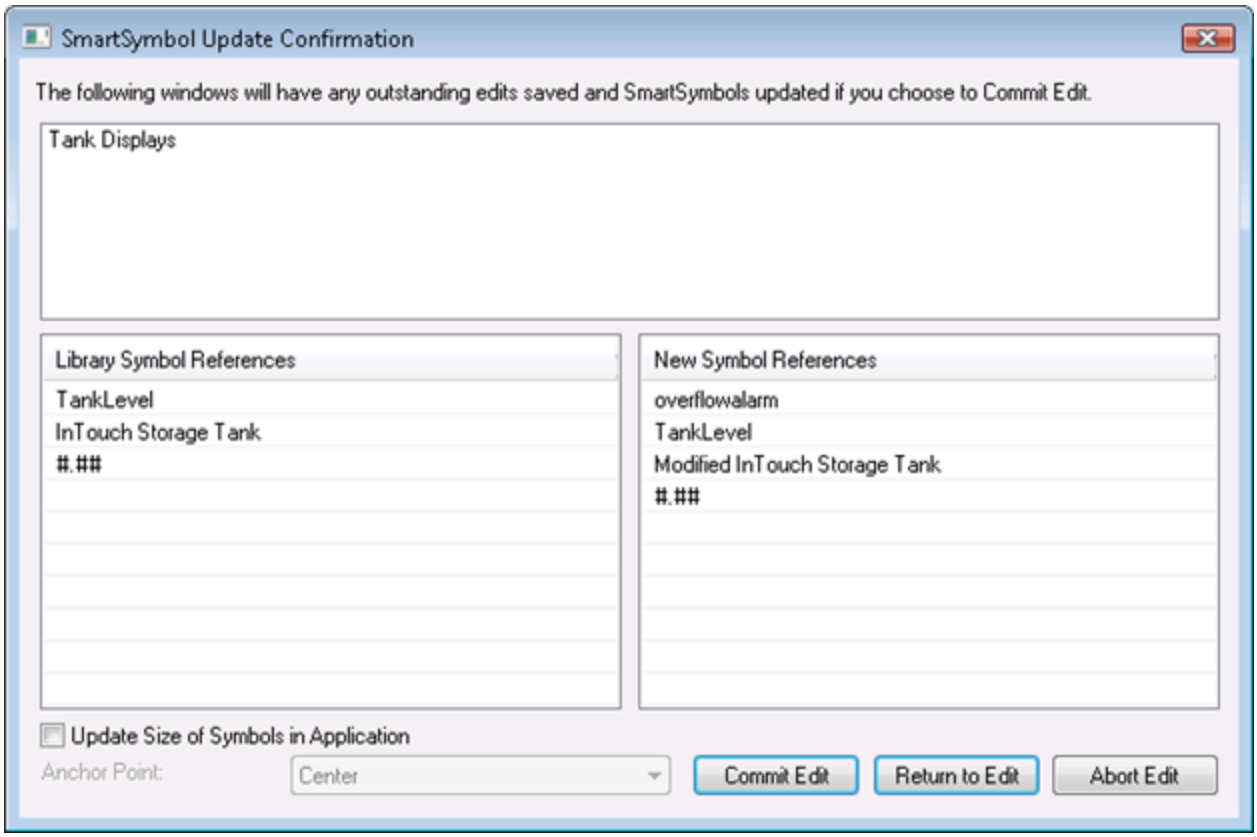
   The symbol is broken into its component elements.

5. You can now edit one or more elements.

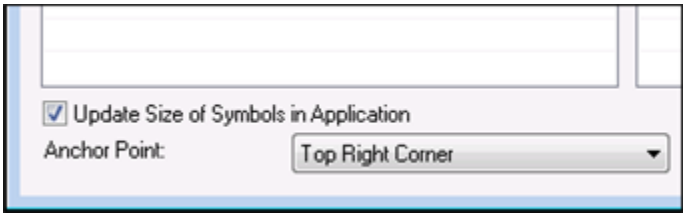
   Note: If you add elements to a cell that is part of a SmartSymbol, this can result in a spatially larger cell. When you propagate the changes to the SmartSymbol instances, you can select whether or not to propagate the size change.

6. When you are done editing, select all elements of the symbol.
7. On the **Animation** menu, in the **Cell** group, select **Make Cell**.
8. On the **Draw** menu, in the **SmartSymbol** group, select **End Edit**.

   The **SmartSymbol Update Confirmation** dialog box appears.



9. If the size of the edited SmartSymbol changed, you can configure the size propagation. Do one of the following:
   - To not affect the size of existing SmartSymbol instances, clear the **Update Size of Symbols in Application** checkbox.
   - To propagate the change in the size of the template to the SmartSymbol instances, select the **Update Size of Symbols in Application** checkbox, and in the **Anchor Point** list, select the part of a SmartSymbol instance to be "anchored" to the screen when the resizing is done.



10. Do one of the following:
    - To apply the changes you made, select **Commit Edit**. The SmartSymbol Manager updates the SmartSymbol template and all instances.
    - To continue editing the SmartSymbol, select **Return to Edit**. The application window reappears for further editing.

Change SmartSymbol instances

You can change any references and static text in a SmartSymbol instance. You can search and replace the static text in the instance.

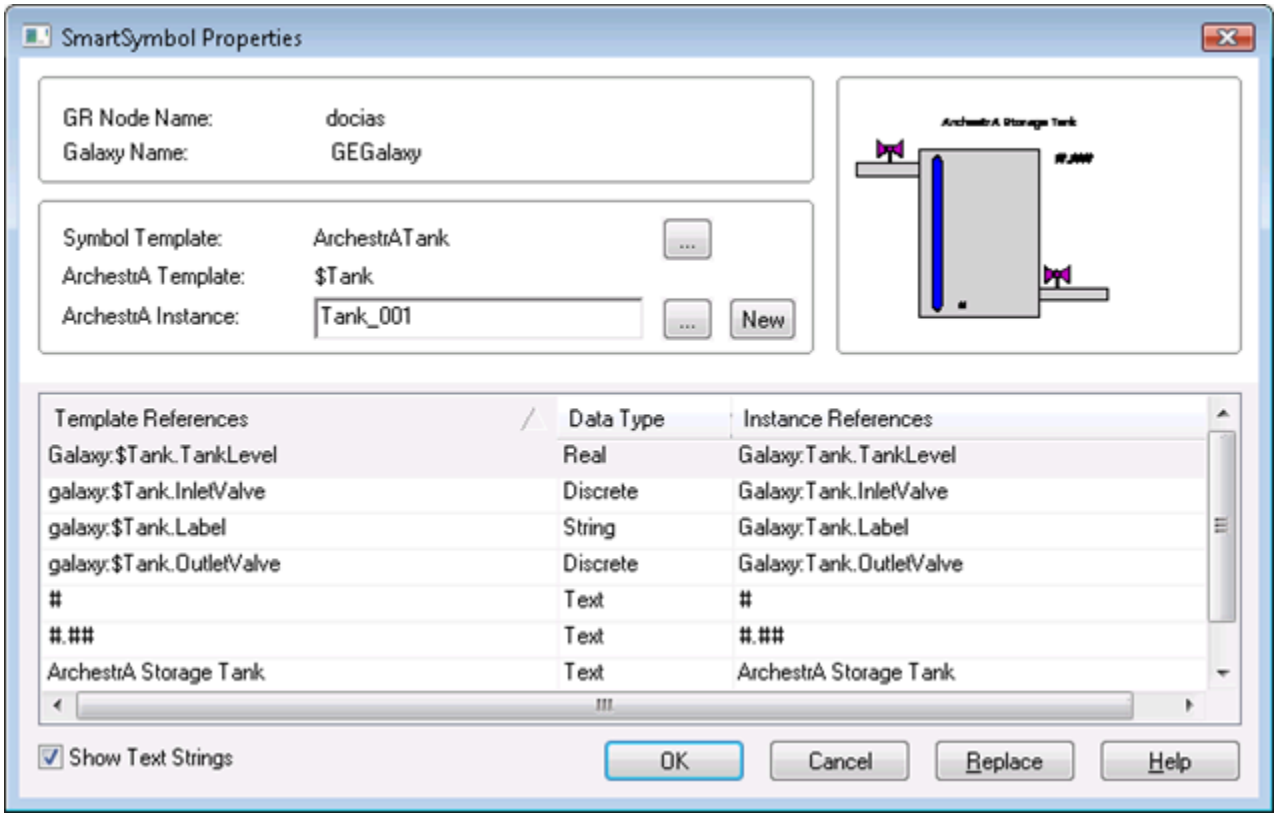
Select a different reference for a SmartSymbol instance

After you place a SmartSymbol in a window as an instance, you can change its references to point to something different, such as another object or a different tag. The SmartSymbol template is not affected.

At run time, you can change the tags a SmartSymbol instance references by using the IOSetRemoteReferences() script function. For more information, see Redirect remote references during run time.

Edit references in a SmartSymbol instance

1. Double-click the SmartSymbol instance. The **SmartSymbol Properties** window appears.



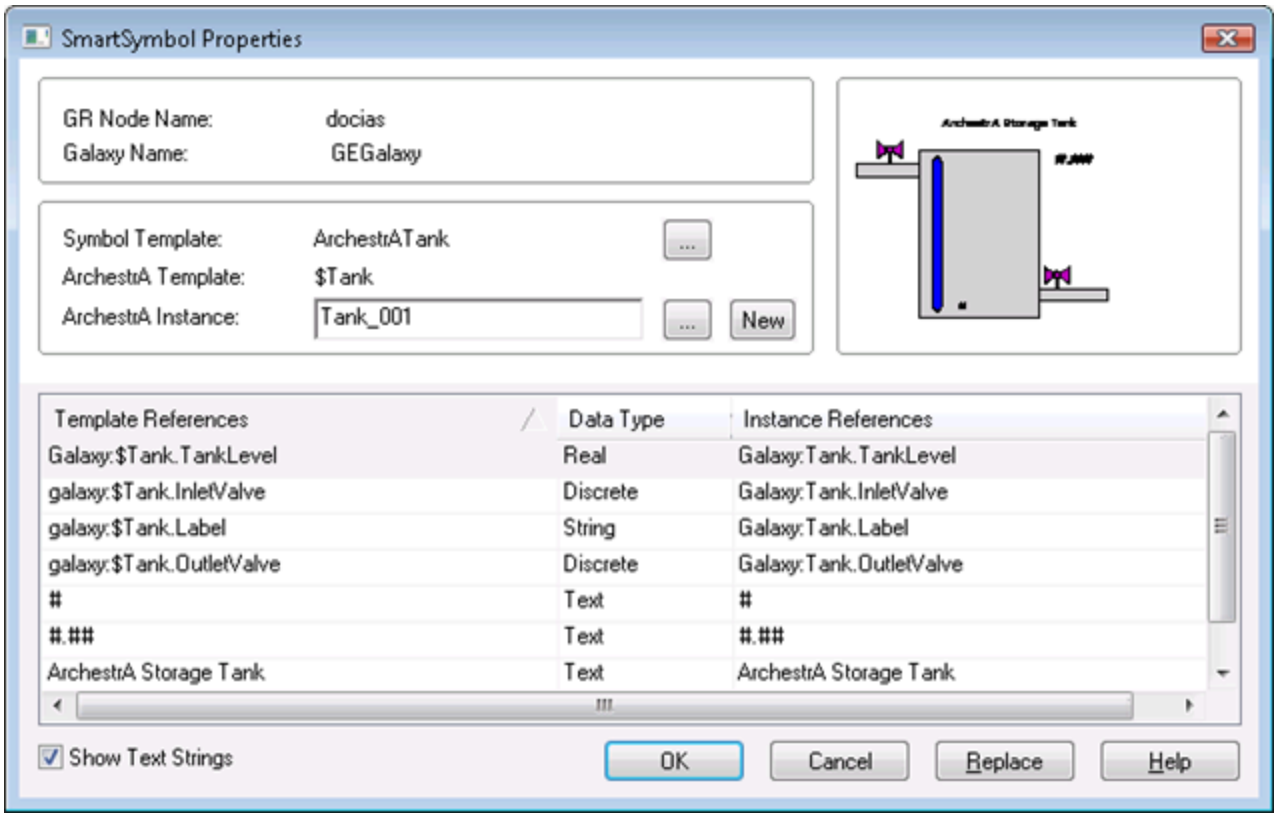
2. Do any of the following:
   - Select the **Ellipsis** button next to **Symbol Template** to select a new SmartSymbol template.
   - Select the **Ellipsis** button next to the **ArchestrA Instance** text box to browse for an ArchestrA object instance.
3. Select a different object instance to map to the SmartSymbol and select **OK**.

Manually edit text and references of a SmartSymbol instance

After you create a SmartSymbol instance in an application window, you can change the static text in the instance.

Change the static text in a SmartSymbol instance

1. Double-click the SmartSymbol instance. The **SmartSymbol Properties** window appears.



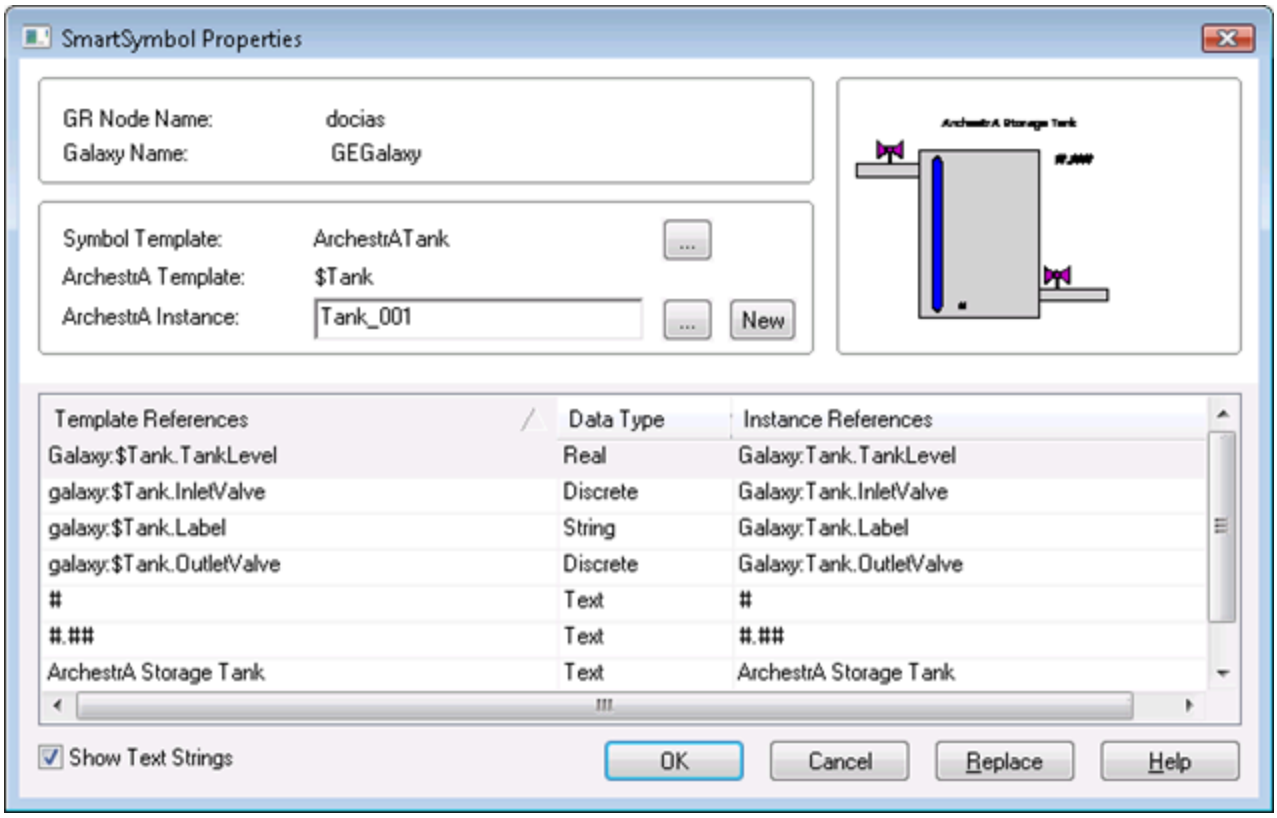
2. In the **Instance References** column, select the text box and modify the text.
3. Select **OK**.

Replace SmartSymbol instance tagnames and text Strings

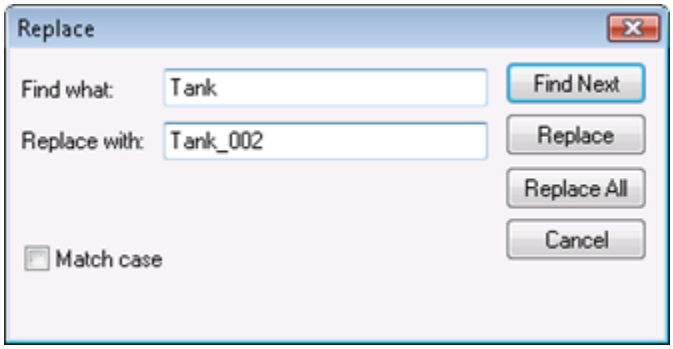
If several references and text strings of a SmartSymbol instance require the same change, you can use the replace feature.

Replace SmartSymbol instance references

1. Double-click the SmartSymbol instance. The **SmartSymbol Properties** window appears.



2. Select **Replace**. The **Replace** dialog box appears.



3. Configure the replacement text strings. Do the following:
   a. In the **Find what** box, type the text you want to replace. Select the **Match case** checkbox make the search case-sensitive.
   b. In the **Replace with** box, type the replacement text. The replacement text is always used exactly as typed.
4. Do one of the following:
   - To replace all text, select **Replace All**.
   - To find and replace the text one instance at a time, select **Find Next** and select **Replace** to replace that one instance.
5. Select **OK**. The SmartSymbol instance in the application window appears with the changed tags and text strings.

Migrate InTouch SmartSymbols

You can use InTouch SmartSymbols in Industrial graphics by importing (migrating) them. The SmartSymbol appearance and configuration is imported and converted to animation configuration.

The imported SmartSymbol can:

- Be added to the existing elements on the canvas.
- Replace the existing elements on the canvas.

Generally, you can import any InTouch SmartSymbol into an Industrial Graphic.

Note: The SmartSymbol can contain objects that cannot be imported, or can be imported but have limited functionality. For a full list of these objects, see Restrictions for SmartSymbol import.

Import InTouch SmartSymbols into an Industrial Graphic

Import an InTouch SmartSymbol into an Industrial Graphic

1. Open the Industrial Graphic Editor.
2. On the **Special** menu, click **Import InTouch SmartSymbol**. The **Find InTouch Application Wizard** appears.
3. If your InTouch application is in a different folder than the default, click the browse button to browse for the path to the InTouch application.
4. Select **Find Applications**. The **Search Root** box shows the path under which all applications are to be searched.
5. If your InTouch application is not under the specified **Search Root** path, change the **Search Root** path by typing a new start folder for the search or browsing for one.
6. Select **Find**. All InTouch applications contained in the specified **Search Root** folder are found and listed.
7. Select the application from which you want to import SmartSymbols and click **Next**. The **Select InTouch SmartSymbol** dialog box appears.
8. Browse to the location of the SmartSymbol in the SmartSymbol hierarchy, select the SmartSymbol that you want to import, and then click **OK**.

   If you already have elements on the canvas, a dialog box appears prompting if you want to replace the existing elements.

   - Select **Yes** if you want to delete the existing elements and import the SmartSymbol on an empty canvas.
   - Select **No** if you want to keep the existing elements and import the SmartSymbol.
9. If the SmartSymbol contains fonts that are currently not installed on the operating system, the **Edit Font Mapping** dialog box appears.

   You can click **Continue** to accept the suggested font mapping, or change the font mapping for each individual font. To do this:

   a. Select the font name in the **Mapped Font** column. A browse button appears.
   b. Select the browse button. The **Supported Font Selection** dialog box appears.
   c. Select a font from the list and click **OK**. The selected font appears in the **Mapped Font** column.
   d. Repeat the steps for any other font you want to map to another font.

Note: If you want to save the mapping for the next time you import a SmartSymbol, check **Save mapping**.

10. The SmartSymbol is imported and appears on the canvas.

Restrictions for SmartSymbol import

When you import an InTouch SmartSymbol, the following configuration is imported:

- InTouch graphics
- Graphical animations
- Scripts
- References

Import InTouch graphics

The following tables shows you InTouch graphics that:

- Can be imported without any problem.
- Can be imported but are changed in their functionality or lose some functionality in the process.
- Cannot be imported.

  The following InTouch graphics can be imported without any problem:

InTouch Graphic	Industrial Graphic Element	Notes
Rectangle	Rectangle	
Rounded Rectangle	Rounded Rectangle	
Ellipse	Ellipse	
Line	Line	
H/V Line	Line	Smart Symbols convert H/V lines to Lines. Therefore, ArchestrA can only generate lines.
Polyline	Polyline	
Polygon	Polygon	
Text	Text	
Bitmap	Bitmap	
Cell	Group	ArchestrA property "Treat as Icon" = false.

InTouch Graphic	Industrial Graphic Element	Notes
Symbol	Group	ArchestrA property "Treat as Icon" = true.
Button	Button	

The following InTouch graphics can be imported, but are changed in their functionality or lose some functionality in the process:

InTouch Graphics	Industrial Graphic Element	Notes
Wizard	Elements	When grouped in a SmartSymbol, it appears as a group of elements.
SmartSymbol	Elements	When grouped in another Smart Symbol, it is broken down into a cell, losing its SmartSymbol properties.

The following InTouch graphics cannot be imported, as they cannot be added to a SmartSymbol:

InTouch Graphic	Industrial Graphic Element	Notes
RealTime Trend	n/a	Cannot be added to a SmartSymbol.
Historical Trend	n/a	Cannot be added to a SmartSymbol.
ActiveX Controls	n/a	Cannot be added to SmartSymbol. This would include all ActiveX alarm controls (Alarm DB View, Alarm Viewer, and so on)

Import graphical animation

When you import an InTouch SmartSymbol, all data configured in InTouch animations is imported to ArchestrA animations. InTouch animations and ArchestrA animations often have a different name, but perform the same function.

The following table shows you which animations correspond to each other:

InTouch Animation Link	ArchestrA animation
User Inputs - Discrete	User Input
User Inputs - Analog	User Input
User Inputs - String	User Input
Sliders - Vertical Slider	Vertical Slider
Sliders - Horizontal Slider	Horizontal Slider
Touch Pushbuttons - Discrete Value	Pushbutton
Touch Pushbuttons - Action	Action Scripts
Touch Pushbuttons - Show Window	not supported
Touch Pushbuttons - Hide Window	not supported
Line Color - Discrete	Line Style
Line Color - Analog	Line Style
Line Color - Discrete Alarm	Line Style
Line Color - Analog Alarm	Line Style
Fill Color - Discrete	Fill Style
Fill Color - Analog	Fill Style
Fill Color - Discrete Alarm	Fill Style
Fill Color - Analog Alarm	Fill Style
Text Color - Discrete	Text Style
Text Color - Analog	Text Style
Text Color - Discrete Alarm	Text Style
Text Color - Analog Alarm	Text Style
Object Size - Height	Height
Object Size - Width	Width
Location - Vertical	Location Vertical
Location - Horizontal	Location Horizontal

InTouch Animation Link	ArchestrA animation
Percent Fill - Vertical	% Fill Vertical
Percent Fill - Horizontal	% Fill Horizontal
Miscellaneous - Visibility	Visibility
Miscellaneous - Blink	Blink
Miscellaneous - Orientation	Rotation
Miscellaneous - Disable	Disable
Miscellaneous - Tooltip	Tooltip
Value Display - Discrete Value	Value Display
Value Display - Analog Value	Value Display
Value Display - String Value	Value Display

Import action scripts

When you import a SmartSymbol, all action scripts associated with objects in SmartSymbol are imported as well. An action script in a SmartSymbol becomes a script animation in an Industrial Graphic.

Most of the predefined InTouch functions (QuickScripts) are imported.

Mathematical functions

The following mathematical functions in InTouch WindowMaker are supported by the Industrial Graphic Editor:

```
Abs, ArcCos, ArcSin, ArcTan, Cos, Exp, Int, Log, LogN, Pi, Round, Sgn, Sin, Sqrt, Tan,
Trunc
```

String functions

The following string functions in InTouch WindowMaker are supported by the Industrial Graphic Editor:

```
Dtext, StringASCII, StringChar, StringCompare, StringCompareNoCase,
StringFromGMTTimeToLocal, StringFromIntg, StringFromReal, StringFromTime,
StringFromTimeLocal, StringInString, StringLeft, StringLen, StringLower, StringMid,
StringReplace, StringRight, StringSpace, StringTest, StringToIntg, StringToReal,
StringTrim, StringUpper, Text, wwStringFromTime
```

System functions

The following system functions in InTouch WindowMaker are supported by the Industrial Graphic Editor:

```
ActivateApp
```

Miscellaneous functions

The following miscellaneous functions in InTouch WindowMaker are supported by the Industrial Graphic Editor:

```
DateTimeGMT, LogMessage, SendKeys, WWControl
```

Import references

When you import a SmartSymbol, the following changes are made to tags and references:

InTouch SmartSymbol	Industrial Graphic	Example
Local Tags	Prefixed with "InTouch:" keyword	Real Memory Tag "TankLevel1" is converted to "InTouch:TankLevel1"
Local Tags with dotfields	Prefixed with "InTouch:" keyword	Discrete Memory Tag "TankLevel1.InAlarm" is converted to "InTouch:TankLevel1.InAlarm"
SuperTags	Prefixed with "InTouch:" keyword. You need to manually enclose the expression by the following syntax: attribute("...");	Real SuperTag member "Reactor1\Level" is converted to "InTouch:Reactor1\Level". You need to change the expression manually as follows: attribute("InTouch:Reactor1\Level");
I/O References	Prefixed with "InTouch:" keyword	Integer I/O Tag "Testprot:i00" is converted to "InTouch:Testprot:i00"
Galaxy References	"Galaxy:" prefix is removed	Galaxy Reference "galaxy:Pump1.Valve1" is converted to "Pump1.Valve1"

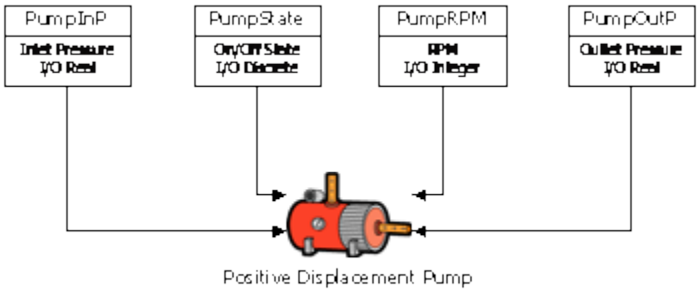
The following items are imported with functional change:

InTouch SmartSymbol	Industrial Graphic	Example
Galaxy:ObjectTagname.Property.#VString	"Galaxy:" prefix is removed but #VString is not supported. Applies also for #VString1, #Vstring2, #VString3 and #VString4	"Galaxy:Tank.PV.#VString4" is converted to "Tank.PV"
Galaxy:ObjectTagname.Property.#ReadSts	"Galaxy:" prefix is removed but #ReadSts is not supported	"Galaxy:Tank.PV.#ReadSts" is converted to "Tank.PV"
Galaxy:ObjectTagname.Property.#WriteSts	"Galaxy:" prefix is removed but #WriteSts is not supported	"Galaxy:Tank.PV.#WriteSts" is converted to "Tank.PV"
Galaxy:ObjectTagname.Property.#EnumOrdinal	"Galaxy:" prefix is removed but #EnumOrdinal is not supported	"Galaxy:Selection.Sel1.#EnumOrdinal" is converted to "Selection.Sel1"

Tags

An InTouch Human Machine Interface (HMI) application is a graphical representation of the components in a manufacturing environment. Plant operators work with this graphical interface to monitor and administer their manufacturing processes.

The figure below shows an example of a pump that is a component of a manufacturing process. The pump has properties with associated values. Pressure, RPM, and status are pump properties whose values are monitored through an HMI.

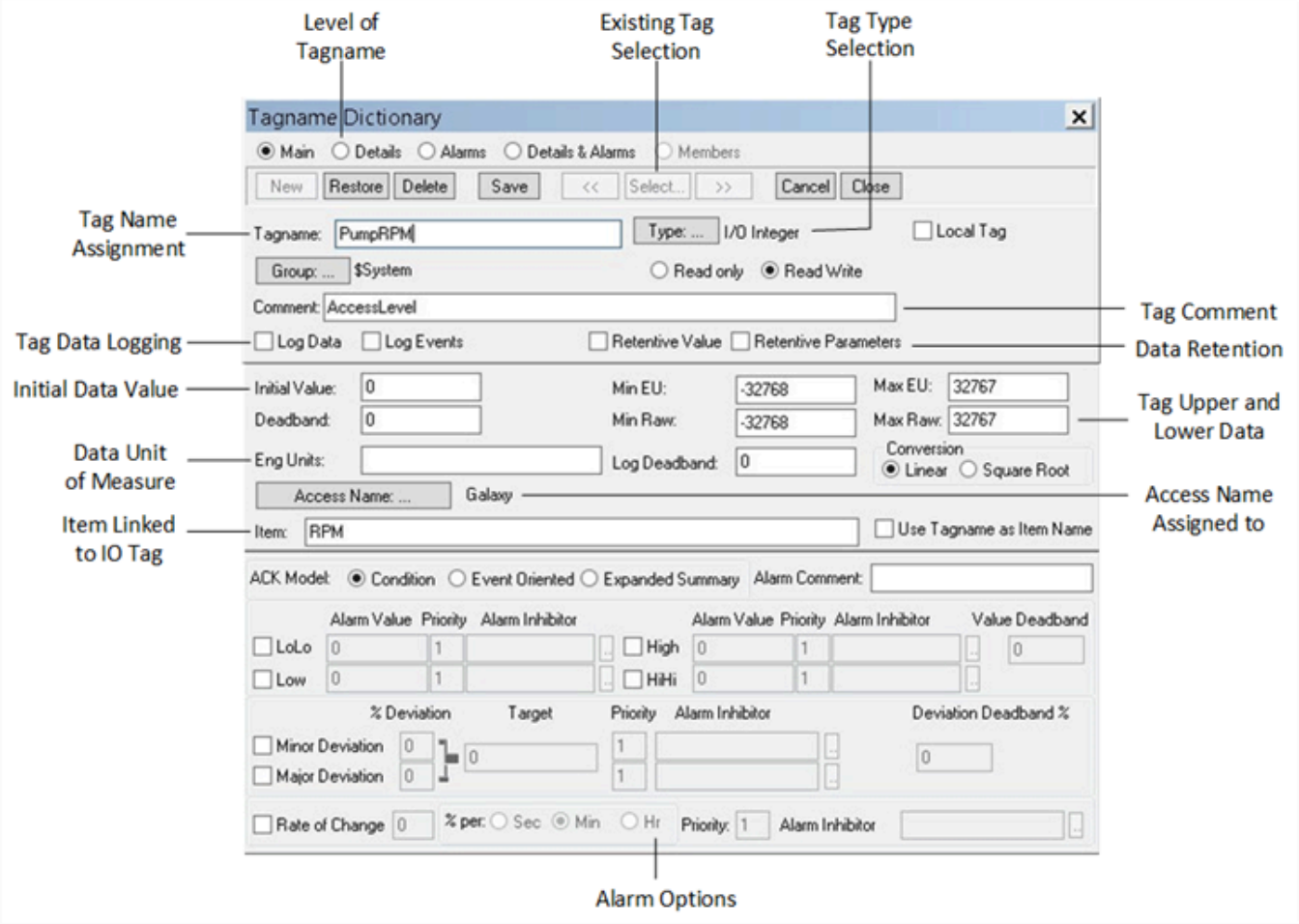


A *tag* represents a data item in an InTouch HMI application. You use tags to make specific component properties accessible as data items from a manufacturing environment. In the figure above, the PumpState tag indicates whether the pump is on or off. You create tags for components in your manufacturing environment whose properties you want to monitor or control in your InTouch application.

You can use different types of tags for the different types of data collected from a manufacturing component. For example, the PumpState tag returns a Boolean On/Off value to indicate if the pump is running or stopped. You assign the appropriate type of InTouch tag for the type of data that you want to be part of your application.

Manage tags with the Tagname dictionary

Using the Tagname dictionary, you create tags for an InTouch application. The figure below shows the **Tagname Dictionary** dialog box with all options to define the properties of an I/O tag.



Plan tag usage

You can reduce development time by identifying the key requirements of an application's tags in a preliminary planning phase. Thorough planning reduces the time you need to create InTouch applications.

Before creating tags:

- Identify all physical components of the process that need to be represented in the InTouch application.

  Create a list of component attributes that need to be represented as data sources in the application.
- Identify the type of data associated with each component attribute.

  Assign each tag a tag type based upon the data associated with the component attribute. For more information about assigning a data type to a tag, see Create new tags.
- Determine the characteristics of data that needs to be incorporated into your InTouch application.

  Assess the following data characteristics for each tag:

- Expected range of data values
- Units of measure assigned to data values
- Initial data value
- Deadband value to set a threshold when a tag's value is recognized as changed
- Messages to be shown when a tag's value changes state

  For more information about defining the characteristics of tag data, see Understand tag properties.

- Develop a tag naming convention and standard.

  Typically, complex applications require many tags. Develop a standardized naming convention that suggests the organization of the tags within the application. For more information about tag naming conventions, see Tag name conventions.

- Determine what process data needs to be saved.

  Selected data is saved to a log file. You can use logged data to create historical trends that show the changes in tag values over time. For more information about setting tag logging, see Tag logging.

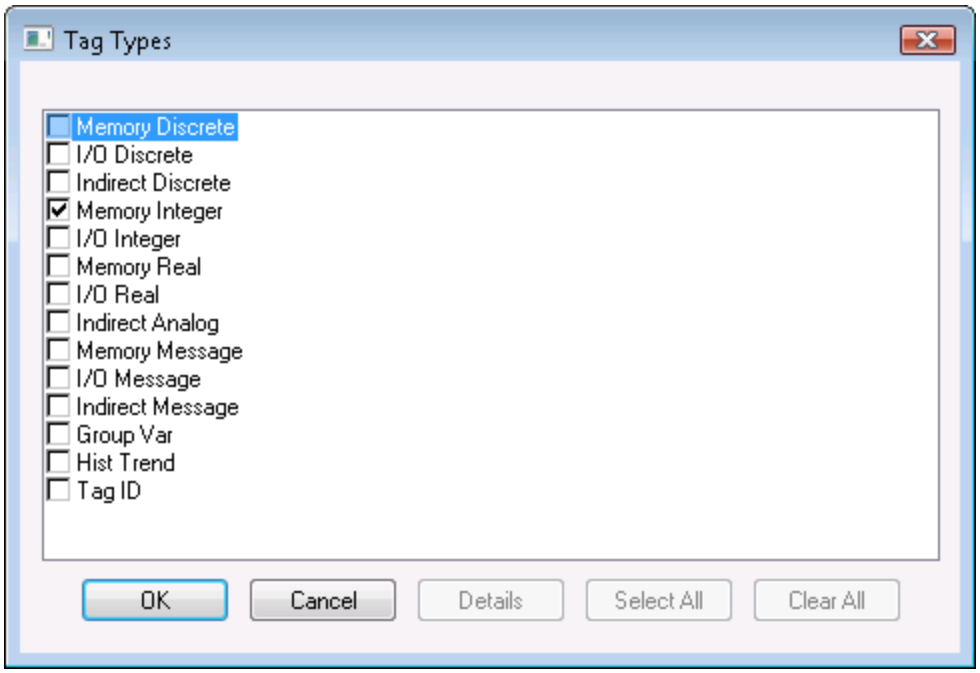
Create new tags

You create tags with the WindowMaker **Tagname Dictionary.** Before you start, analyze your plant process to determine the tags that you need to create in your InTouch application.

Create a new tag

1. Open the InTouch application in WindowMaker.
2. On the **Home menu**, in the **Tags** group, select **Tag Dictionary**.

   The first time you open the Tagname Dictionary, the definition for the **$AccessLevel system tag** appears in the **Tagname** box. After saving a new tag, the **Tagname Dictionary** shows the most recently saved tag definition.
3. Do the following:
   a. select **New**. The **Tagname** box clears.
   b. Enter a name for the new tag. For more information about tag naming requirements, see Tag name conventions.
   c. Optionally, type a comment about the new tag in the **Comment** box.
4. Select **Type**. The **Tag Types** dialog box appears with a list of supported InTouch tag types.

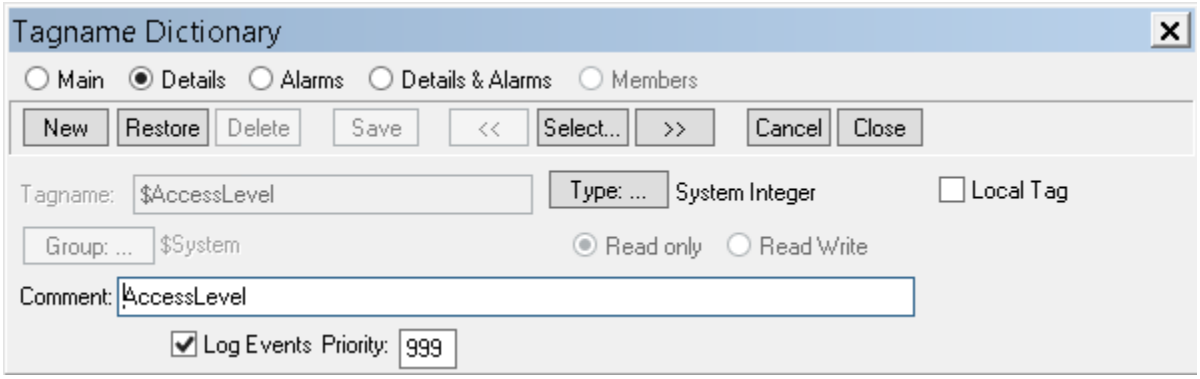


5. Highlight a type of tag from the list and select **OK**. The Tagname Dictionary reappears and shows the type of tag you selected.
6. If needed, select **Details** to see the additional Tagname Dictionary options for the selected tag type.
7. Specify further tag options in the **Tagname Dictionary** dialog box.

   For more information about specifying tag properties, see Configure tag properties.
8. Select **Save**. Select **Close** to close the **Tagname Dictionary** dialog box.

Configure tag properties

Using the **Tagname Dictionary** dialog box, you can specify common tag properties that are part of every tag definition. You must assign a name to each tag. You can add an optional comment. Both the tag name and comment are common properties of all tags.



Each type of InTouch tag has unique data properties. After you select a tag type, the **Tagname Dictionary** dialog box expands to show a set of options, based upon the selected tag type.

Common tag properties

You must assign a unique name to each tag. An optional comment can be part of the tag definition. It is good practice to assign an appropriate comment for each tag you define.

All tags belong to an alarm group, which is another common tag property. By default, all tags belong to the $System alarm group. For more information about assigning tags to other alarm groups, see Configure alarms.

Tag name conventions

When you name your tags, use a consistent naming convention if you need many tags with similar properties.

Follow these naming conventions for InTouch tag names:

- Use 128 characters or fewer in a tag name.
- Use an alphanumeric (**A-Z, a-z**, **0-9**) as the first character of a tag name.

  A best practice is to use only alphanumeric characters for tag names.
- Use at least one alphabetic character in the tag name.

  (Optional) Use the following special characters:

- Dash	! Exclamation point	# Number sign
$ Dollar sign	% Percent	& Ampersand
? Question mark	@ At sign	_ Underscore

  If possible, avoid using special characters in tag names unless absolutely required by your application.
- Avoid using a dash (-) in a tag name.

  A dash is a valid character for an InTouch tag name. But, InTouch evaluates a dash as a negation or subtraction operator in logical or arithmetic expressions. For example, the expression A=B-C can be interpreted as A=B minus C or the tag named B-C is assigned to tag A.

  Hyphen or minus sign (-) will not be allowed when Tag Names start with a numeric character.
- Do not use blank spaces in tag names.
- Do not use a number in the tag name that can be interpreted as an exponential number.

  For example, you cannot name a tag 125E4 because it could be interpreted as a base number with an exponent raised to the fourth power.
- Do not use a number in the tag name that can be interpreted as a hexadecimal number.

  For example, you cannot name a tag 0x123B because it could be interpreted as a hexadecimal number.

Automatically name tags

As you name your tags in the Tagname Dictionary, the InTouch HMI tracks the naming conventions you are using. If you name your tags Pump01, Pump02, the InTouch HMI suggests the next tag name as Pump03. You can accept or reject this name. This naming help is called Indexing.

Indexing is based on the last consecutive number in a tag name. For example, if your tag name is PumpInP04LotB99A, the InTouch HMI suggests the next tag name as PumpInP04LotB100A, not PumpInP05LotB99A.

Tag comments

You can enter an optional comment up to 160 characters in the Tagname Dictionary **Comment** box when you create a tag.

The first time you access the Tagname Dictionary, the default comment for the **$AccessLevel** system tag appears in the **Comment** box. Delete this comment to prevent it from being associated with any tags that you create.

Understand tag properties

After specifying common tag properties, you must define other properties that are specific to the type of tag you are creating. The following table shows the basic properties for memory tags by tag type.

Tag Type	Unique Properties
Discrete	**Initial Value**, **On Msg**, **Off Msg**, **Comment**
Integer	**Initial Value**, **Min Value**, **Deadband**, **Eng Units**, **Max Value**, **Log Deadband**, **Comment**
Real	**Initial Value**, **Min Value**, **Deadband**, **Eng Units**, **Max Value**, **Log Deadband**, **Comment**
Message	**Maximum Length**, **Initial Value**, **Comment**

I/O tags have additional properties to establish network communication and transform raw data from network devices to normalized values used by the InTouch application. For more information about defining I/O tags, see Configure I/O tag properties.

Value ranges, measurement units, and an initial value

Discrete, integer, real, and message tags are assigned an initial value when the InTouch application starts in WindowViewer. In the case of a discrete tag, the initial value is one of the possible binary states. For integer and real tags, the initial value is the number associated with the tag when the application starts. The initial value is specified in the Tagname Dictionary.

You can specify the initial tag value to be the last value of the tag when the application stops running in WindowViewer. By selecting the **Retentive Value** option from the Tagname Dictionary, the tag is assigned its last active value as the initial value when the application starts again.

Integer and real tags include properties that set the lower and upper boundaries of the range of possible numerical values assigned to a tag. Both integer and real tags include the **Min Value** and the **Max Value** properties that define the lower and upper limits of the range.

Integer and real tags also include the **Eng Units** property to assign an engineering units label that describes the unit of measure for the tag's value. For example, you can assign PSI as the **Eng Units** property for an integer tag associated with pump pressure.

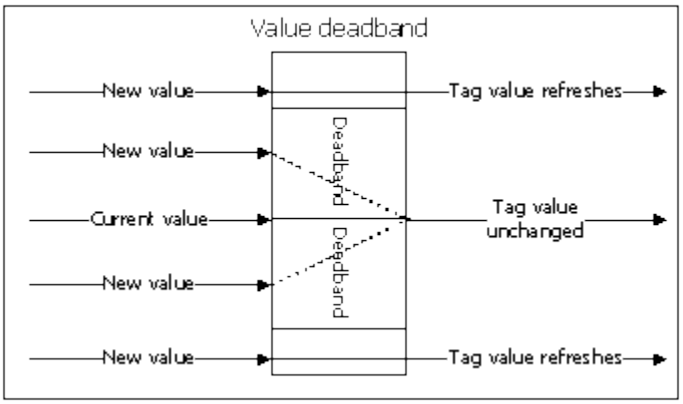
Tag deadbands

A deadband is a sensitivity setting for tag values. A deadband is usually associated with I/O tags whose values

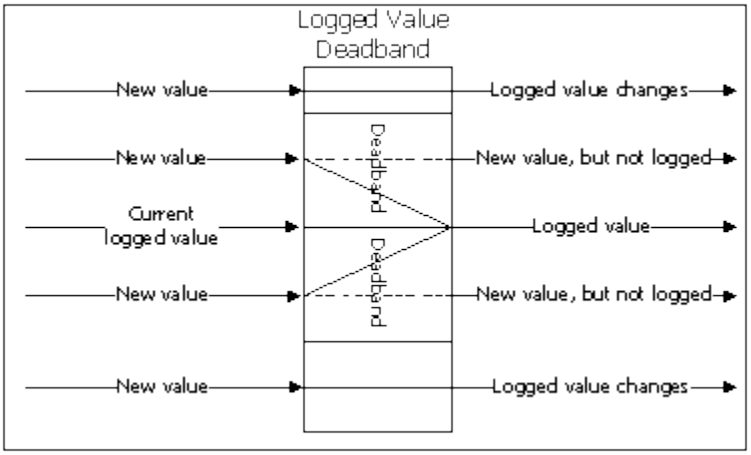
change constantly. A deadband filters out small momentary changes in a tag value to reduce the amount of InTouch data processing.

The Tagname Dictionary includes two deadband properties for tags associated with integer and real data.

- Value Deadband: The value deadband property sets a threshold that must be exceeded before WindowViewer refreshes the tag's value in run-time memory. The following figure shows the absolute deadband range around a current tag value.



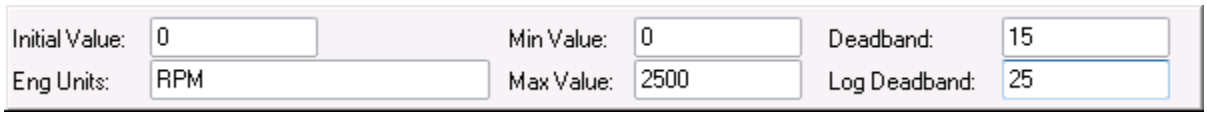
- Log Value Deadband: The log deadband sets a threshold that must be exceeded before the tag's value is written to the log file. The following figure shows the deadband around the current value of a logged tag.



Only new tag values outside of the deadband are written to the log file. Small value changes within the deadband range are ignored and not logged.

Set a tag deadband

1. Open the **Tagname Dictionary** dialog box.
2. Choose **Select**. The **Select Tag** dialog box appears. The tags currently defined for the application are listed.
3. Select an integer or real tag type from the list.
4. Select **OK**. The detail portion of the **Tagname Dictionary** dialog box shows additional options when you select a real or integer tag.



5. Set the value deadband by entering an integer or real number in the **Deadband** box based upon the selected tag type.

   The value deadband sets an absolute threshold level in engineering units.

6. Set the log deadband by entering an integer or real number in the **Log Deadband** box based upon the selected tag type.

   Like the value deadband, the log deadband sets an absolute threshold in engineering units.

7. Select **Save** to save your deadband changes.
8. Select **Close** to close the Tagname Dictionary dialog box.

Tag value retention

The detail portion of the **Tagname Dictionary** includes two properties to retain tag values and operator changes to alarm limits.

All tag types include a **Retentive Value** property. Select **Retentive Value** to retain the current value of the tag when the application stops. When you start the application again, WindowViewer uses the retained value as the initial value of the tag.

WindowViewer does not write retained values to I/O devices when WindowViewer starts the application again. I/O values are updated after the I/O Server initially scans the device providing data.

The **Retentive Value** option cannot be selected or cleared for new or existing tags if WindowViewer is running. When you select this option, the initial value of the tag is constantly updated to reflect its current value. When WindowViewer stops, the initial value is set to the last retained value. If this option is later cleared, the initial value of the tag is set to the last retained value.

Integer and real tags include the **Retentive Parameters** property. Select **Retentive Parameters** to retain any changes made by the operator to a tag's alarm limit while the application is running. WindowViewer uses the modified alarm limit as the initial value for the alarm limit when the application is restarted.

I/O connection

All types of I/O tags must identify the Access Name and Item Name of the external data source. For more information about specifying the Access Name and Item Name for I/O tags, see Set I/O access parameters.

Tag logging

During run time, WindowViewer can write an entry to the historical log file each time a tag's value changes more than the specified log deadband. WindowViewer also writes entries to the log at a fixed interval regardless of current tag values. By default, this fixed interval is one hour.

Note: For more controllable and versatile logging, consider using Historian to store InTouch historical data.

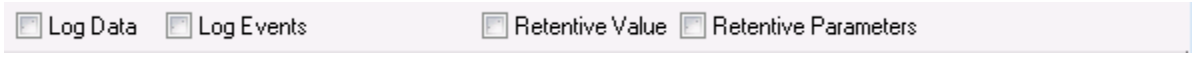
The **Tagname Dictionary** dialog box includes separate options to log data and events to the log file. You can set value logging options. For information about setting event logging, see Configure alarms.

For a tag's value to be written to the historical log file, historical logging must be enabled. For more information about setting general logging properties, see Configure historical logging.

For integer and real tags, you can set the Log Deadband in their respective details dialog boxes. The **Log Deadband** option specifies how many engineering units a tag's value must change to write a log entry.

Configure logging for a tag

1. Open the **Tagname Dictionary**.
2. Select the tag whose data you want saved to the log file.
3. Select **Log Data.**



4. Select **Log Events** if you want to log value changes to the tag initiated by an operator, an I/O, QuickScript, or operating system. The **Priority** box appears after you select **Log Events**.

   The **Priority** value determines the event priority for the tag. Valid values are 1 to 999, where 1 is the highest priority and 999 is the lowest.
5. Select **Save** and then close the Tagname Dictionary.

Create discrete tags

You can specify discrete tags to show the binary state of internal processes running in an InTouch application. A discrete tag must be assigned an initial value of on or off. Also, you can specify messages that appear in the alarm event window when the process associated with the tag transitions into or out of an alarm state.

The following steps show how to define a memory discrete tag. I/O discrete tags indicate the binary state of all inputs and outputs from programmable controllers, process computers, and data from network nodes.

For more information about setting the properties of an I/O discrete tag, see Specify a discrete I/O Tag.

Define an initial value and messages for a memory discrete tag

1. Select **Memory Discrete** as the type of tag in the **Tag Types** dialog box.
2. Select **Details** at the top of the **Tagname Dictionary** dialog box to show the detail options. The detail portion of the **Tagname Dictionary** dialog box appears.

3. Select **On** or **Off** as the initial value associated with the tag. The tag is set to this initial value when the application starts.
4. Enter messages in the **On Msg** and **Off Msg** boxes that appear when the tag transitions in and out of an alarm state.

   These messages are available for use in any animation link or script, regardless of whether the tag has alarms configured or not.

   - If you define a discrete alarm that is active when the tag value is equal to 1 (On, True), the message entered in the **On Msg** box appears in the **Value** and **Limit** columns of your ActiveX alarm displays.

     When the tag's alarm state returns to normal, the message entered in the **Off Msg** box appears in the **Value** column and the On message remains in the **Limit** column.
   - If you define a discrete alarm that is active when the tag value is equal to 0 (Off, False), the message entered in the **Off Msg** box appears in the **Value** and **Limit** columns of your ActiveX alarm displays.

     When the tag's alarm state returns to normal, the message entered in the **On Msg** box appears in the

Value column and the Off message remains in the **Limit** column.

5. Save your changes to the tag.

Create integer and real tags

You can specify integer and real tags to show numerical values of processes running in an InTouch application.

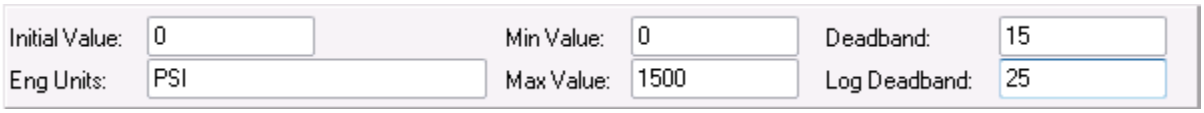
The following steps show how to define memory and I/O integer and real tags. Memory integer and real tags must be assigned an initial value. You also must set minimum and maximum data ranges.

For information about setting the I/O properties of an I/O integer and real tags, see Specify integer and real I/O tags.

Important: InTouch real numbers are limited to eight-digit precision. To avoid possible rounding errors, do not exceed eight-digit precision when you specify the properties of your real tags. For more information, see IEEE decimal units.

Define memory integer and real tag values

1. Assign memory integer or memory real as the type of tag in the **Tag Types** dialog box. The detail portion of the **Tagname Dictionary** dialog box appears.



2. Set the properties for integer and real tags. Do the following:
   - In the **Initial Value** box, type the integer or real number associated with the tag when the application starts.
   - In the **Min Value** box, type the minimum integer or real number for the tag.

     The **Min Value** sets the minimum possible value for numbers associated with memory integer and real tags.
   - In the **Max Value** box, type the maximum integer or real number for the tag.

     The **Max Value** sets the maximum possible value for numbers associated with memory integer and real tags.
   - In the **Eng Units** box, type the label you want to use for the tag's engineering units.
3. Save your changes to the tag.

For more information about setting the tag's deadband and log deadband properties, see Tag deadbands.

Create message tags

You can specify message tags for both internal and external processes. These tags include properties to specify an initial message that appears when WindowViewer starts the application and a comment that can be read from the Alarm Viewer.

For more information about defining an I/O message tag, see Specify a message I/O tag.

Define a memory message tag

1. Assign memory message as the type of tag in the **Tag Types** dialog box.

2. If needed, select **Details** to show the detail portion of the **Tagname Dictionary** dialog box.

3. Set the properties of a memory message tag. Do the following:
   - In the **Maximum Length** box, type an integer that is the maximum number of characters that can appear in the tag's message. Or, accept the default length of 131 characters, which is the maximum message length.
   - In the **Initial Value** box, type the text of the message that you want assigned to the tag when WindowViewer starts the application.
   - In the **Alarm Comment** box, type a message that can be read in the AlarmViewer control if the message tag has the **Log Events** option selected.
4. Save your changes to the tag.

Create I/O tags

I/O tags have a set of common properties that specify network connectivity between the InTouch application and external processes. The detail portion of the **Tagname Dictionary** dialog box includes options to set an I/O tag's external properties.

For more information about setting I/O properties, see Data access with I/O.

Modify tags

Modifying a tag is similar to creating a tag. You select the tag to be modified from the Tagname Dictionary. Then, you change the tag's properties following the same steps to create a tag.

Important: It is not easy to modify a tag's type after it is used in an InTouch application. The tag type selection may be limited to similar data types. You should carefully select the correct data type when you define your tags.

Modify a tag

1. Open the **Tagname Dictionary** dialog box.
2. Choose **Select**. The **Select Tag** dialog box appears. A list of tags currently defined for the application is shown.
3. Select the tag to be modified from the list.
4. Select **OK**. The **Tagname Dictionary** dialog box shows the values specified for the selected tag.
5. Make changes to the tag's properties.
6. Select **Save** to update the tag with your changes.
7. Select **Close** to close the Tagname Dictionary.

Create InTouch tags from OPCUA server

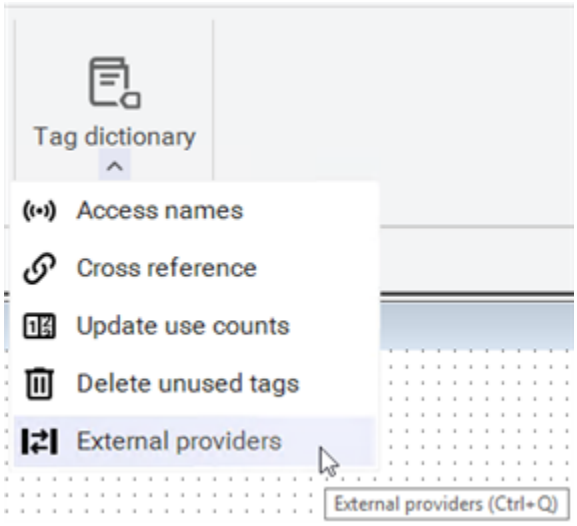
The **External providers** option under **Tag dictionary** lists all the OPC UA server connections configured in the Gateway Communication Driver that is present on the same machine. Using this option, you can avoid several manual steps and create access names and InTouch tags for OPC UA item references easily.

Create InTouch Tags from OPCUA Server configured in the Gateway Communication Driver:

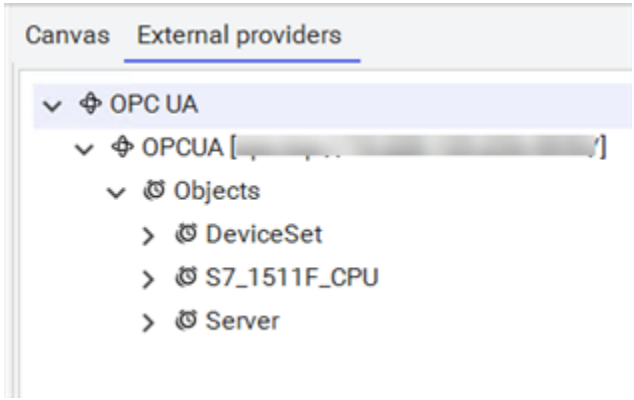
Prerequisites:

- Configure the OPC UA server connection in the Gateway Communication Driver. Refer to the Gateway Communication Driver help for more information.
- Both Gateway Communication Driver and the InTouch WindowMaker should be on the same machine.

1. Open the InTouch application in WindowMaker.
2. On the **Home** menu, in the **Tags** group, use the dropdown arrow under **Tag Dictionary** and select **External providers**. Alternatively, you can use the keyboard shortcut **Ctrl + Q**.

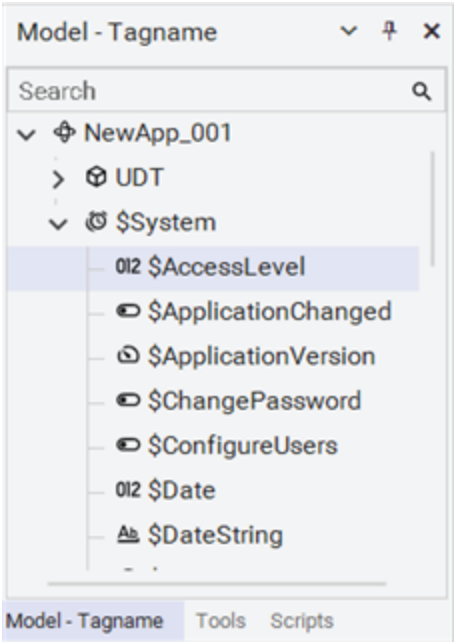


   All the OPC UA server connections configured in the Gateway Communication Driver are displayed.
3. You can expand the OPC UA tree to view and browse the item references.

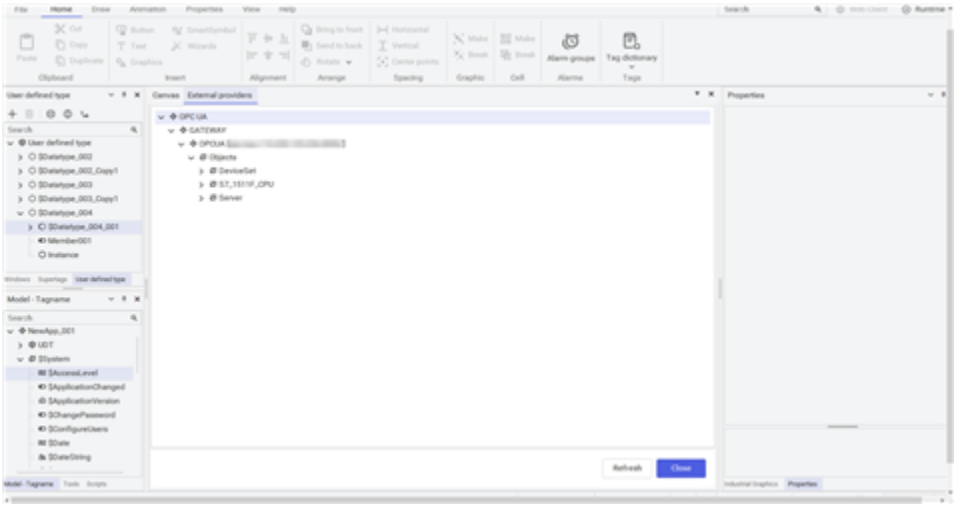


   Note:
   The OPC UA servers from the remote Gateway communication Driver will not be displayed.
   After the OPC UA items are displayed, if you configure another OPC UA server, select the **Refresh** button present at the bottom of the **External providers** window to view the newly added OPC UA server items.
4. Open the **Model - Tagname** pane.

   The **Model - Tagname** pane displays all the alarm groups and tags of the InTouch application. For more information on the Model - Tagname pane refer to the Model - Tagname topic.



5. Drag and drop the items from the **External providers** window to the Model - Tagname. You can select multiple items by pressing and holding **Ctrl** key while selecting the items.



   - You can drag and drop to the required alarm group node to add the OPC UA tags under a particular alarm group.
   - If you randomly drag and drop; the tags are created under **$System** node.

     Once you drag and drop the items, the OPC UA tags gets created and you can see the newly created OPC UA tags in **Model - Tagname** pane.

6. To close the **External providers** window, select **Close**.

Additional information:

- For different OPC UA connection different access name is added. The access name of the tag will be the node name of the OPC UA connection in the Gateway Communication Driver.

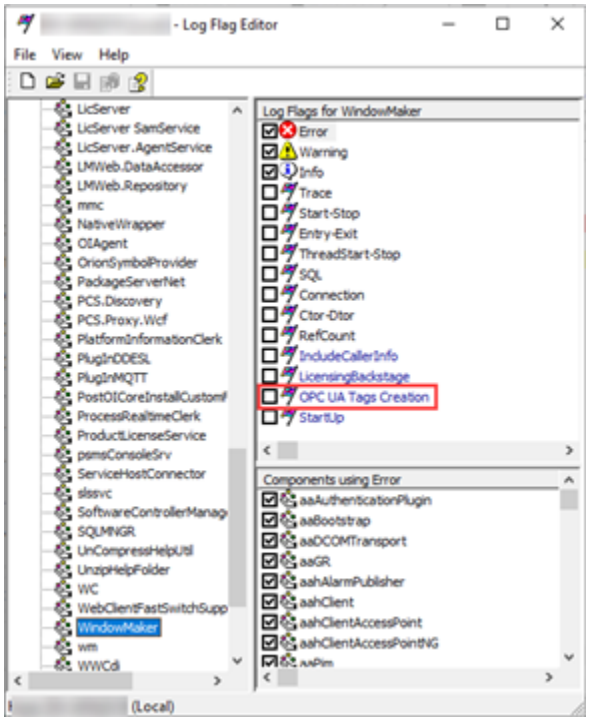
  Note: After adding the tag if you rename the OPC UA node in the Gateway Communication Driver, you have

to manually change the access name and it will not be changed automatically.

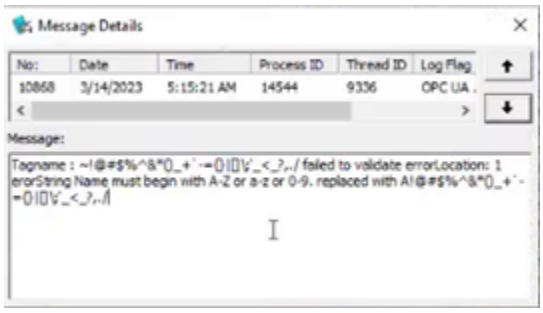
- Duplicate tag names are appended with index number. For example: if there are two tags of name ‘Input’, then the first tag will be named as ‘Input’ and the second tag will be named as ‘Input_1’.

  Note: If the tag name length exceeds maximum tag length 128, the characters from end of the tag name are removed to make length 128. When the tag name length exceeds 128 characters length and there is a duplicate tag name, then the number of characters from end of the tag name are removed so that the tag name +_{ index } length measures 128.

- If data type of the OPC UA tag is unknown, then it will create the corresponding InTouch tag as I/O message type.
- You can also use InTouch as OPC UA server by configuring it as OPC UA server in Application Manager. For more information refer to Configure and using the InTouch OPC UA server topic.
- In **Log Flag Editor**, under WindowMaker if you select the **OPC UA Tags Creation** checkbox then you can view the log messages related to OPC UA tags in the Log Viewer.

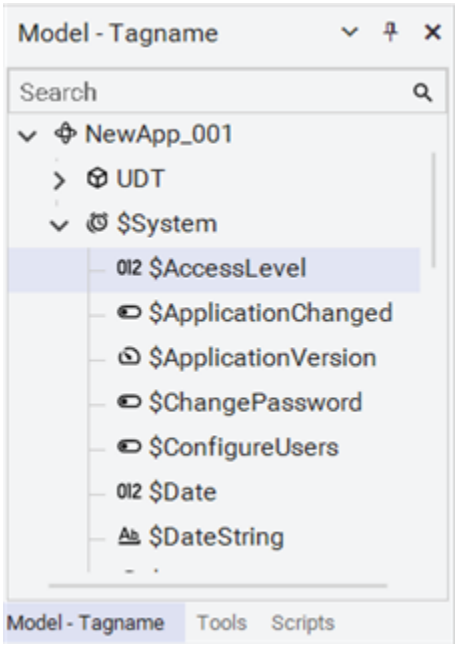


- If the OPC UA item name has unsupported special characters, while creating the corresponding InTouch tags the unsupported special character in the tag name will be replaced by ‘A’ or ‘_’. If the unsupported character is at the beginning of the tag, then it is replaced with 'A', in other places it is replaced with '_’. You can view the details on what character is replaced in the **Message Details** window of the Log Viewer.



Model - Tagname

The **Model - Tagname** tab is in the left-hand side of the InTouch WindowMaker along with Windows and Supertags tabs. Select the **Model - Tagname** tab to view the Model - Tagname pane. It displays all the alarm groups and tags of the InTouch application. You can search for required tags using the **Search** box.



Open a tag

1. In the **Model - Tagname** pane, right-click the tag that you want to open.
2. Select **Open**.
   The tag will be opened in the **Tagname Dictionary** window.

Move a tag from one alarm group to another

1. In the **Model - Tagname** pane, select the required tags that you want to move.
2. Drag and drop to the required alarm group.
   The selected tags will be moved to the required alarm group.

Delete a tag

1. In the **Model - Tagname** pane, right-click the tag that you want to delete.
   You can select multiple tags.
2. Select **Delete**.
3. In the confirmation message to delete the tag, select **Yes**.
   The selected tag gets deleted.

Delete tags

A count is maintained for all tags defined for an InTouch application. The tag count does not decrease automatically when a window containing animation links or scripts is deleted. The tags associated with the deleted window are still considered to be in use and cannot be deleted.

To be able to delete a tag that is no longer used, you must close WindowViewer and update local and remote tag use counts. You can determine where a tag is being used with the InTouch Cross-Reference utility. For more information about using the Cross-Reference Utility and updating tag counts, see Reduce tag usage.

Tags can be deleted after the use count is updated. For more information, see Delete unused tags.

Print a tag list and usage information

You can print the contents of an InTouch application's database, windows, and scripts in WindowMaker using the WindowMaker Printout utility.

For more information, see Saving and Printing a Tag Cross-Reference List.

Use tag dotfields to view or change tag properties

Every type of InTouch tag has a unique set of properties that describe the data or possible conditions associated with the tag. A dotfield identifies a tag property. There is a dotfield for almost every tag property shown in the Tagname Dictionary. Some dotfields are common to every type of InTouch tag. For example, the .Name dotfield is always associated with a tag's name. Other dotfields apply only to the unique properties of a specific type of tag.

Using dotfields in a script, expression, or user input, you can monitor and modify a tag's properties while an application is running. The following example shows the syntax of a dotfield to access tag properties within a script or expression.

tag_name.property_dotfield

For example, to enable operators to change the HiHi alarm limit while an application is running, you can create an Analog - User Input touch link. Then, apply the link to a button that is defined with the Analog_Tag.HiHiLimit dotfield expression. During run time, the operator simply selects the button and types in a new value for the HiHi alarm limit for the tag.

You can use dotfields to allow input and output of data associated with a tag and you can use historical dotfields to modify the historical trend currently shown from a running application. For example, you can use dotfields in scripts that allow operators to modify historical trend scrolling, lock or reposition the scooters on the trend, or reassign the pens to new tags.

Available dotfields for tag types

Each type of InTouch tag has a set of dotfields associated with its unique properties. The following table shows an alphabetical list of dotfields for all types of tags.

Dotfields	Tag Types														
	Memory			I/O				Indirect			Miscellaneous				
	Discrete	Integer	Real	Message	Discrete	Integer	Real	Message	Discrete	Analog	Message	Alarm Group	Hist Trend	Distributed Alarm Object / Controls	TagID
.Ack	•	•	•		•	•	•		•	•		•			
.AckDev		•	•			•	•			•		•			
.AckDsc	•				•				•			•			
.AckROC	•	•	•			•	•			•		•			
.AckValue		•	•			•	•			•		•			
.Alarm	•	•	•		•	•	•		•	•		•			
.AlarmAccess															•
.AlarmAckModel	•	•	•		•	•	•		•	•					
.AlarmClass															•
.AlarmComment	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

Dotfields	Tag Types													
.AlarmDate													•	
.AlarmDev		•	•			•	•			•		•		
.AlarmDevCount		•	•			•	•			•		•		
.AlarmDevDeadband		•	•			•				•				
.AlarmDevUnAckCount		•	•			•	•			•		•		
.AlarmDisabled	•	•	•		•	•	•		•	•		•		
.AlarmDsc	•				•				•			•		
.AlarmDscCount	•				•				•			•		
.AlarmDscDisabled	•				•				•			•		
.AlarmDs	•				•				•			•		

Dotfields	Tag Types														
.cEnabled															
.AlarmDscInhibitor	•				•				•			•			
.AlarmDscUnAckCount	•				•				•			•			
.AlarmEnabled	•	•	•		•	•	•		•	•		•			
.AlarmGroup														•	
.AlarmGroupSel														•	
.AlarmHiDisabled		•	•			•	•			•					
.AlarmHiEnabled		•	•			•	•			•					
.AlarmHiHiDisabled		•	•			•	•			•					
.AlarmHiHiEn		•	•			•	•			•					

Dotfields	Tag Types														
abled															
.AlarmHiHiInhibitor		•	•			•	•			•					
.AlarmHiInhibitor		•	•			•	•			•					
.AlarmLimit														•	
.AlarmLoDisabled		•	•			•	•			•					
.AlarmLoEnabled		•	•			•	•			•					
.AlarmLoInhibitor		•	•			•	•								
.AlarmLoLoDisabled		•	•			•	•			•					
.AlarmLoLoEnabled		•	•			•	•			•					
.AlarmLo		•	•			•	•			•					

Dotfields	Tag Types														
LoInhibitor															
.AlarmMajDevDisabled		•	•			•	•			•					
.AlarmMajDevEnabled		•	•			•	•			•					
.AlarmMajDevInhibitor		•	•			•	•			•					
.AlarmMinDevDisabled		•	•			•	•			•					
.AlarmMinDevEnabled		•	•			•	•			•					
.AlarmMinDevInhibitor		•	•			•	•			•					
.AlarmName														•	
.Alar														•	

Dotfields	Tag Types														
mOprName															
.AlarmOprNode														•	
.AlarmPri														•	
.AlarmProv														•	
.AlarmROC		•	•			•	•			•					
.AlarmROCCount		•	•			•	•			•					
.AlarmROCDisabled		•	•			•	•			•					
.AlarmROCEnabled		•	•			•	•			•					
.AlarmROCInhibitor		•	•			•	•			•					
.AlarmROCUnAckCount		•	•			•	•			•					

Dotfields	Tag Types													
.AlarmState													•	
.AlarmTime													•	
.AlarmTotalCount	•	•	•		•	•	•		•	•		•		
.AlarmType													•	
.AlarmUnAckCount	•	•	•		•	•	•		•	•		•		
.AlarmUserDefNum1	•	•	•		•	•	•		•	•		•		
.AlarmUserDefNum1Set	•	•	•		•	•	•		•	•		•		
.AlarmUserDefNum2	•	•	•		•		•		•	•		•		
.AlarmUserDefNum2Set	•	•	•		•	•	•		•	•		•		

Dotfields	Tag Types														
.AlarmUserDefStr	•	•	•		•	•	•		•	•		•			
.AlarmUserDefStrSet	•	•	•		•	•	•		•			•			
.AlarmValDeadband		•	•			•	•			•					
.AlarmValue			•				•			•		•		•	
.AlarmValueCount		•	•			•	•			•		•			
.AlarmValueUnAckCount		•	•			•	•			•		•			
.Caption														•	
.ChartLength													•		
.ChartStart													•		
.Comment	•	•	•	•	•	•	•	•	•	•	•	•	•		•

Dotfields	Tag Types														
.DevTarget		•	•			•	•			•					
.DisplayMode													•		
.Enabled														•	
.EngUnits		•	•			•	•			•					
.Freeze														•	
.HiHiLimit		•	•			•	•			•					
.HiHiSet		•	•			•	•			•					
.HiHiStatus		•	•			•	•			•					
.HiLimit		•	•			•	•			•					
.HiSet		•	•			•	•			•					
.HiStatus		•	•			•	•			•					
.ListChanged														•	
.ListCount														•	
.ListIndex														•	

Dotfields	Tag Types														
.LoLimit		•	•			•	•			•					
.LoLoLimit		•	•			•	•			•					
.LoLoSet		•	•			•	•			•					
.LoLoStatus		•	•			•	•			•					
.LoSet		•	•			•	•			•					
.LoStatus		•	•			•	•			•					
.MajorDevPct		•	•			•	•			•					
.MajorDevSet		•	•			•	•			•					
.MajorDevStatus		•	•			•	•			•					
.MaxEU		•	•			•	•			•					
.MaxRange													•		
.MaxRaw		•	•			•	•								
.MinEU		•	•			•	•			•					
.MinorDe		•	•			•	•			•					

Dotfields	Tag Types														
vPct															
.MinorDevSet		•	•			•	•			•					
.MinorDevStatus		•	•			•	•			•					
.MinRange													•		
.MinRaw		•	•			•	•			•					
.Name	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
.NewIndex														•	
.NextPage														•	
.Normal	•	•	•		•	•	•		•	•		•			
.NumAlarms														•	
.OffMsg	•				•				•						
.OnMsg	•				•				•						
.PageNum														•	
.Pen1													•		

Dotfields	Tag Types														
through.Pen8															
.PendingUpdates														•	
.PrevPage														•	
.PriFrom														•	
.PriTo														•	
.Quality	•	•	•	•	•	•	•	•	•	•	•				
.QualityLimit	•	•	•	•	•	•	•	•	•	•	•				
.QualityLimitString	•	•	•	•	•	•	•	•	•	•	•				
.QualityStatus	•	•	•	•	•	•	•	•	•	•	•				
.QualityStatusString	•	•	•	•	•	•	•	•	•	•	•				
.QualitySubstatus	•	•	•	•	•	•	•	•	•	•	•				
.QualityS	•	•	•	•	•	•	•	•	•	•	•				

Dotfields	Tag Types														
ubstatusString															
.QueryState														•	
.QueryType														•	
.RawValue	•	•	•		•	•	•		•	•		•			
.ReadOnly														•	
.Reference	•	•	•	•	•	•	•	•	•	•	•				
.ReferenceComplete	•	•	•	•	•	•	•	•	•	•	•				
.ROCPct		•	•			•	•			•					
.ROCSet		•	•			•	•								
.ROCStatus		•	•			•	•			•					
.ScooterLockLeft													•		
.Scooter													•		

Dotfields	Tag Types														
LockRight															
.ScooterPosLeft													•		
.ScooterPosRight													•		
.Successful														•	
.SuppressRetain														•	
.TagID	•	•	•		•	•	•		•	•					
.TimeDate	•	•	•	•	•	•	•	•	•	•	•				
.TimeDateString	•	•	•	•	•	•	•	•	•	•	•				
.TimeDateTime	•	•	•	•	•	•	•	•	•	•	•				
.TimeDay	•	•	•	•	•	•	•	•	•	•	•				
.TimeHour	•	•	•	•	•	•	•	•	•	•	•				
.Tim	•	•	•	•	•	•	•	•	•	•	•				

Dotfields	Tag Types														
.eMinute															
.TimeMonth	•	•	•	•	•	•	•	•	•	•	•				
.TimeMsec	•	•	•	•	•	•	•	•	•	•	•				
.TimeSecond	•	•	•	•	•	•	•	•	•	•	•				
.TimeTime	•	•	•	•	•	•	•	•	•	•	•				
.TimeTimeString	•	•	•	•	•	•	•	•	•	•	•				
.TimeYear	•	•	•	•	•	•	•	•	•	•	•				
.TopIndex														•	
.TotalPages														•	
.UnAck	•	•	•		•	•	•		•	•		•			
.UpdateCount													•		
.UpdateinProgress													•		
.Upd													•		

Dotfields	Tag Types														
ateTrend															
.Value(Tagname)	•	•	•	•	•	•	•	•	•	•	•	•			•
.Value(Windows Control)														•	
.Visible														•	

Dotfields can be categorized by their intended function. For more information about dotfield functional categories, see the following sections:

Category	See
Values and limits	Change the value limits of a tag.
Alarm parameters	Control alarm properties of tags and groups at runtime in AVEVA™ InTouch HMI Application Run Time.
I/O	Data access with I/O.
Distributed Alarm Object	*Working with the Distributed Alarm Display Object*.
Trend display	Trending tag data.
Window controls	Wizards

Change the value limits of a tag

Raw input data from an I/O Server is transformed to a range of values appropriate for an InTouch application. The values of a tag are clamped to a range bounded by minimum and maximum values specified by the **Min Raw** and **Max Raw** properties of the Tagname Dictionary.

Then, these raw values are converted to a range of engineering units set by the **Min EU** and **Max EU** options. You can use a set of dotfields to monitor and modify a tag's raw value and engineering units ranges.

The following table lists the dotfields that monitor or change tag values while an application is running.

Dotfield	Read/Write	Shows
.MinRaw	Read-only	The low clamp setting of a raw value received from an I/O Server.
.MaxRaw	Read-only	The high clamp setting of a raw value received from an I/O Server.
.MinEU	Read-only	The minimum value in engineering units assigned to the tag.
.MaxEU	Read-only	The maximum value in engineering units assigned to the tag.
.EngUnits	Read/Write	The text value assigned to an analog tag from the **Eng Units** option of the Tagname Dictionary.
.RawValue	Read-only	The actual discrete or analog value received by the tag from an I/O Server before scaling is applied.
.Value	Read/Write	The current value of a tag.
.OnMsg	Read/Write	The message assigned to a discrete tag when its value evaluates to True, On, or 1.
.OffMsg	Read/Write	The message assigned to a discrete tag when its value evaluates to False, Off, or 0.
.Comment	Read/Write	The tag comment specified from the Tagname Dictionary.

View the raw value limit

Raw input tag values from an I/O Server may require clamping before they can be used in an InTouch application. Clamping restricts raw values to a range with defined lower and upper limits. The .**MinRaw** and .**MaxRaw** dotfields show the lower and upper boundaries of the raw input range.

.MinRaw Dotfield

The .**MinRaw** dotfield shows the Min Raw low clamp setting assigned to a tag. The value for .**MinRaw** dotfield comes from the **Min Raw** value assigned to the I/O tag in the Tagname Dictionary. Any raw value less than this setting is clamped to this minimum value.

Category

Tags

Usage

```
tag_name.MinRaw;
```

Parameter

Tag_name

The name of any I/O integer, I/O real, and indirect analog tag.

Remarks

This read-only dotfield shows the value assigned to the Min Raw low clamp setting.

Data Type

Real or integer (read-only).

Valid Values

Any analog value.

Example

The following script shows an error window if the raw value associated with pump inlet pressure is outside of the lower and upper value boundaries set by the tag's **Min Raw** and **Max Raw** properties.

```
IF ((PumpInP.RawValue > PumpInP.MaxRaw) OR
   (PumpInP.RawValue < PumpInP.MinRaw)) THEN
   Show "Instrument Failure Window";
ENDIF;
```

See Also

.EngUnits, **.MinEU**, **.MaxEU**, **.MaxRaw**, **.RawValue**

.MaxRaw Dotfield

The .**MaxRaw** dotfield shows the Max Raw high clamp setting assigned to an I/O tag from the **Max Raw** property in the Tagname Dictionary. Any raw data value that exceeds this setting is clamped to this maximum raw value.

Category

Tags

Usage

```
Tag_name.MaxRaw
```

Parameter

Tag_name

The name of any I/O integer, I/O real, and indirect analog tag.

Remarks

This read-only dotfield shows the value assigned to the Max Raw high clamp setting.

Data Type

Real or integer (read-only).

Valid Values

Any analog value.

Example

This script determines if a tag value is out of normal operating range and a window is shown if this is the case.

```
IF ((Temp01.RawValue > Temp01.MaxRaw) OR (Temp01.RawValue <
   Temp01.MinRaw))THEN
   Show "Instrument Failure Window";
ENDIF;
```

See Also

.EngUnits, **.MinEU**, **.MaxEU**, **.MinRaw**, **.RawValue**

View the raw value of a tag

The .**RawValue** dotfield shows the actual discrete or analog value of a monitored property received from an I/O Server. The raw value is the actual input value before clamping and scaling are applied to normalize the value to the tag's engineering units.

.RawValue Dotfield

The .**RawValue** dotfield shows the actual value received from an I/O Server by WindowViewer. The .RawValue dotfield allows you to access the value of an I/O tag before InTouch applies scaling.

Category

Tags

Usage

```
Tag_name.RawValue
```

Parameter

Tag_name

The name of any I/O integer, I/O real, I/O discrete, indirect discrete, and indirect analog tag.

Remarks

This read-only dotfield is used to show the actual discrete or analog I/O value before InTouch applies scaling.

Data Type

Any data appropriate for the type of tag associated with the .RawValue dotfield. For example, real numbers for real tags or discrete values for discrete tags (read-only).

Example

The following script issues a warning message when the raw pump inlet pressure is below or above the tag's minimum and maximum clamping limits.

```
IF ((PumpInP.RawValue > PumpInP.MaxRaw) OR (PumpInP.RawValue < PumpInP.MinRaw)) THEN
   AlarmMessage = "Pump sensor is out of calibration or requires replacement.";
ENDIF;
```

See Also

.EngUnits, **.MinEU**, **.MaxEU**, **.MinRaw**, **.MaxRaw**

View the engineering units value limit

A value from an I/O Server is considered raw when it first arrives in WindowViewer. Raw values may require scaling. The InTouch HMI performs an arithmetic transformation on the raw clamped input values to scale them to the engineering units range of the tag. I/O Integer and real tag types include **Min EU** and **Max EU** properties that show the lower and upper boundaries of the engineering unit range.

.MaxEU Dotfield

The **.MaxEU** dotfield shows the maximum engineering unit value assigned to the specified tag from the Tagname Dictionary.

Category

Tags

Usage

```
Tag_name.MaxEU
```

Parameter

Tag_name

Any integer, real, or indirect analog tag.

Remarks

The .MaxEU dotfield is used to scale raw data values to an engineering unit range defined for the tag. It defines the upper limit of engineering unit range.

Data Type

Real for real tags and integer for integer tags (read-only).

Valid Values

Depends on the type of tag specified.

Example

A level gauge is read by a Programmable Logic Controller in the field. The level transmitter sends a signal that ranges between 4 and 20mA. The PLC converts this signal to an integer value between 0 and 4095. This value is assigned to the TankTwoLevel tag.

Displaying the raw value (between 0 and 4095) provides no useful data to the operator. It is necessary to scale this value to an appropriate engineering range.

To accomplish this, the Minimum engineering unit and Maximum engineering unit fields must be set up correctly. In our example, if the raw value of 0 (4mA from the field) translated to "0 Gallons" and the value of 4095 (20mA from the field) translated to "100 Gallons", the following set up would be required to show the correct value on the screen:

```
TankTwoLevel.MinRaw = 0;
TankTwoLevel.MaxRaw = 4095;
TankTwoLevel.MinEU = 0;
```

```
TankTwoLevel.MaxEU = 100;
```

With these settings, when the raw value in the field is 4095, the value shown to the operator is 100.

See Also

.EngUnits, **.MinEU**, **.MinRaw**, **.MaxRaw**, **.RawValue**

.MinEU Dotfield

The **.MinEU** dotfield shows the minimum engineering unit value assigned to the specified tag from the Tagname Dictionary.

Category

Tags

Usage

```
Tag_name.MinEU
```

Parameter

Tag_name

Any integer, real, or indirect analog tag.

Remarks

The .MinEU dotfield is used to scale raw data values to an engineering unit range defined for the tag. It defines the lower limit of engineering unit range.

Data Type

Real for real tags and integer for integer tags (read-only).

Valid Values

Depends on the type of tag specified.

Example

This example assigns the engineering unit range defined for the Tag1 tag to the **AbsoluteTagRange** tag.

```
AbsoluteTagRange = (Tag1.MaxEU - Tag1.MinEU);
```

See Also

.EngUnits, **.MaxEU**, **.MinRaw**, **.MaxRaw**, **.RawValue**

Change the engineering units of a tag

You can associate a dotfield to a tag to determine the text value of the engineering unit assigned to the tag.

.EngUnits Dotfield

The .**EngUnits** dotfield shows the text value assigned to an analog tag with the **Eng Units** property. The .**EngUnits** dotfield shows the engineering unit as a text value.

Note: Values written to this field are not retentive.

Category

Tags

Usage

```
Tag_name.EngUnits
```

Parameter

Tag_name

Any integer, real, or indirect analog tag.

Data Type

Message (read/write).

Remarks

The .**EngUnits** dotfield does not affect the scale, conversion, or format of the actual data associated with the tag.

Valid Values

Any string containing 0 to 31 characters.

Example

The following script calls a Fahrenheit temperature conversion function if the tag's engineering unit is Celsius.

```
IF Temperature.EngUnits == "Celsius" THEN
   CALL TempFConvert(Temperature);
ENDIF;
```

See Also

.MinEU, **.MaxEU**, **.MinRaw**, **.MaxRaw**, **.RawValue**

View the value of tag in engineering units

InTouch tags are assigned a default dotfield when no dotfield is explicitly specified within the application.

.Value Dotfield

The .**Value** dotfield shows the current value of the specified tag in engineering units. .**Value** is the default InTouch dotfield implicitly applied to all tags. If a tag does not have an assigned dotfield, the .**Value** dotfield is assumed.

Category

Tags

Usage

```
tag_name.Value
```

Parameter

tag_name

Any type of tag except the Hist Trend tag.

Remarks

You rarely need to use the **.Value** dotfield. However, in some instances, it makes a calculation or parameter usage more clear.

Data Type

The same as the specified tag's type (read/write).

Valid Values

Depends on the type of tag specified.

Example

The following statement sets the value of the memory integer **PumpRPM** tag equal to 100:

```
PumpRPM.Value=100;
```

which is functionally equivalent to:

```
PumpRPM=100;
```

View or change discrete tag messages

The .**OnMsg** and .**OffMsg** dotfields show the messages assigned to a discrete tag's on or off states from the Tagname Dictionary. A discrete tag's on and off messages are short strings with a maximum of 15 characters.

.OnMsg Dotfield

The .**OnMsg** dotfield allows you to access the On message assigned to a discrete tag from the Tagname Dictionary.

Category

Tags

Usage

```
Tag_name.OnMsg
```

Parameter

Tag_name

Any discrete tag.

Data Type

Message (read/write). Values written to this dotfield are not retentive.

Valid Values

Any string containing 0 to 15 characters.

Example

The following statement issues a message if the indirect Ind**PumpState** tag On message is assigned a string value of "Pump1 running".

```
IF IndPumpState.OnMsg == "Pump1 running" THEN
   TypeOfTag = "The IndPumpState tag is assigned to Pump1.";
ENDIF;
```

See Also

.OffMsg

.OffMsg Dotfield

The .**OffMsg** dotfield allows you to access the Off message assigned to a discrete tag from the Tagname Dictionary.

Category

Tags

Usage

```
Tag_name.OffMsg
```

Parameter

Tag_name

Any discrete tag.

Data Type

Message (read/write). Values written to this dotfield are not retentive.

Valid Values

Any string containing 0 to 15 characters.

Example

The following statement assigns the appropriate string to the **StateMessage** tag according to the state of the **MyDiscrete** tag.

```
StateMessage=Dtext (MyDiscrete, MyDiscrete.OnMsg, MyDiscrete.OffMsg);
```

See Also

.OnMsg

View or change the comment of a tag

A tag comment can be permanently changed only from the Tagname Dictionary. You can assign another comment with the .**Comment** dotfield while the application is running. This only changes the comment for the duration of the run-time session. It does not permanently change the tag's comment in the Tagname Dictionary. After WindowViewer is shut down and restarted, the original comment is assigned to the tag.

.Comment Dotfield

The .**Comment** dotfield shows the comment assigned to a tag from the Tagname Dictionary. A tag comment can be a string up to 160 characters.

Category

Tags

Usage

```
Tag_name.Comment
```

Parameter

Tag_name

Any tag name.

Remarks

The comment associated with a tag can be modified with the .Comment dotfield while an InTouch application is running. After the application is stopped, the original comment specified from the Tagname Dictionary remains assigned to the tag.

Data Type

Message

Valid Values

Any string from 1 to 160 characters.

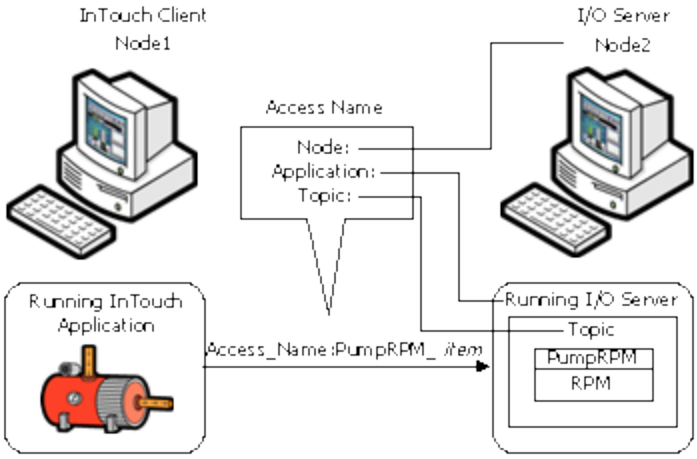
Example

The following statement creates an operator message by combining a tag's assigned comment to the name of the tag:

```
OperatorMessage=PumpRPM.Name + " has a comment of: " + PumpRPM.Comment;
```

Data access with I/O

You can develop distributed applications in which the functional components of an InTouch system are located on different nodes. The figure below shows how you configure an I/O request for data stored on another node.



You can set up an InTouch application to identify an element of data stored on another node by using a three-part addressing convention. This convention includes the node, application, and topic names. To obtain data from a remote node, you need to configure an Access Name for your InTouch application that specifies these three items.

For example, if you want to access data from a remote I/O Server running on another node, your Access Name consists of the following:

Access Name Option	Description
Node Name	Node name of the computer running the I/O Server program.
Application Name	Name of the I/O Server program running on the node. For example, "UA_SampleServer" might be the name of an OPC UA server. For more information about the application names associated with Operations Integration (OI) servers, refer to the OI server documentation.
Topic Name	Label assigned to the I/O Server Device Group.

If you use an Excel spreadsheet as your InTouch data source, you can define your Access Name as follows:

Access Name Option	Description
Node Name	Node name of the computer running the Excel program.
Application Name	Excel is the Application Name.
Topic Name	Name of the Excel book and spreadsheet containing the requested data. For example, [Book1]Sheet1.

In addition to the node, application, topic, and item, you need to specify the type of data located on the remote node. This information determines the I/O type for the tag when it is defined in the Tagname Dictionary.

Work with the Gateway Communication Driver

Gateway Communication Driver is an application that acts as a portal and protocol converter to allow two computer systems or programs to communicate with one another. You can use OI Gateway to link clients and data sources that communicate using different protocols, such as OPC and OPC UA. You can access Gateway Communication Driver from InTouch to connect to field devices.

InTouch leverages Gateway Communication Driver to use OPC, OPC UA, and other protocols to communicate between automation and control applications, field systems and devices, and business and office applications.

Gateway Communication Driver is bundled with WindowViewer and is automatically installed when you install InTouch. It can also be installed as a standalone application.

About the Gateway Communication Driver

The Gateway Communication Driver acts as a communications protocol converter. It can be used to link clients and data sources that communicate using different protocols.

This user assistance publication covers only the information you need to configure and run the Gateway Communication Driver component. The documentation that accompanies the related components provide details on their operation.

You can troubleshoot problems with Gateway Communication Driver using the Log Viewer, a snap-in to the Operations Control Management Console (OCMC). See the Log Viewer help file to find information on:

- Viewing error messages.
- Determining which messages are shown.
- Bookmarking error messages.

You may also be able to troubleshoot problems using your client application, such as the InTouch HMI software. The client application can use system device items to determine the status of nodes and the values of some parameters.

The basic rules for Gateway Communication Driver include:

- One instance of Gateway Communication Driver can run per node.
- The Gateway can be activated and deactivated using the OI Server Manager snap-in.
- The Gateway can be activated as a COM Server (OPC Server) using standard COM activation mechanisms.
- The Gateway can be run only out-of-proc within OPC clients.
- The Gateway can communicate only with ArchestrA data source components delivered with Application Server v2.0 and later. Earlier versions of Application Server are not supported.

Get started with the Gateway Communication Driver

The following steps explain the workflow of getting started with Gateway Communication Driver.

1. Install InTouch.
   - Gateway Communication Driver is included as part of the InTouch installation.
   - You must separately install OPC and OPC UA servers.

2. Install OPC or OPC UA servers and configure them to communicate with your field devices.
3. Configure other protocols as necessary.
4. Create your application in InTouch.
5. Configure Gateway Communication Driver to point to your installed OPC server or OPC UA server or other data sources.
6. Activate Gateway Communication Driver.

   For more information about setting up Gateway Communication Driver, see *Setting up the Gateway Communication Driver for the First Time* in the Gateway Communications Driver Help.

Supported InTouch communication protocols

You can configure the InTouch HMI to use DDE or SuiteLink. The InTouch HMI also supports the Message Exchange communication protocol.

For more information about using Message Exchange within an InTouch application, see Access Application Server data from InTouch.

Dynamic Data Exchange

The Dynamic Data Exchange (DDE) communication protocol enables Windows applications to communicate with each other. DDE implements a client-server relationship between two concurrently running applications. The server application provides data and accepts requests from any other application interested in its data. Requesting applications are called clients. An InTouch application can be simultaneously both a client and a server.

SuiteLink

SuiteLink is a TCP/IP-based protocol designed specifically for industrial applications. SuiteLink provides data integrity, high throughput, and simple diagnostic procedures. The SuiteLink protocol is supported by Microsoft Windows NT 4.0 and later.

SuiteLink is not a replacement for DDE or NetDDE. Each connection between a client and a server depends on your network requirements.

SuiteLink provides the following:

- Value Time Quality (VTQ) places a time stamp and quality indicator on all data values delivered to VTQ-aware clients.
- Extensive diagnostics of data throughput, server loading, computer resource consumption, and network transport are made accessible through the Microsoft Windows operating system performance monitor.
- Consistent high data volumes can be maintained between applications even if the applications are on a single node or distributed over a large set of nodes.
- The network transport protocol is TCP/IP using Microsoft's standard Winsock interface.
- You can securely configure the Suitelink node using the TLS 1.2 protocol.

Configure a secure Suitelink communication as a standard user

A secure Suitelink connection, enabled with TLS 1.2 protocol, encrypts the communication channel between client and server. If you are using the encrypted Suitelink, to access remote date, you must configure the System Management Server (SMS) from the Configurator. The encryption is achieved through the certificates provided by SMS. For the SuiteLink server and client to use encrypted communication, the ASB Runtime Components feature must be selected in the SuiteLink 3.0 installation.

Since the introduction of secure SuiteLink (V3), SuiteLink has been providing the fallback so that a SuiteLink server can accept incoming connections in a secure (V3, encrypted) or an unsecure (V2, unencrypted) mode together. From the System Platform 2023 release, by default, the fallback mode to accept V2 connection is disabled. If you want to enable the fallback mode to accept the unsecure connection, you have to make the configuration changes in the System Management Server. You can verify the mode of SuiteLink connection, whether secure or unsecure, by checking the WWSLS component in the Log Viewer.

Note: As of System Platform 2023, the SuiteLink connection is by default set to secure with TLS encryption. If unsecure fallback mode is not enabled and secure connection requirements are not met, then SuiteLink connection will fail. For more information, see Configure the System Management Server in the Configurator.

Enable or disable the fallback (unsecure) Suitelink connection using the Registry Editor

1. Open the Registry Editor.
2. Navigate to the path: [HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ArchestrA\WebApplications\Default\SuiteLink]
3. Locate the entries:

   "V2Server"=dword:00000001

   "V3Server"=dword:00000001
4. To enable the fallback mode, set the value of V2Server to 1.
5. To disable the fallback mode, set the value of V2Server to 0.
6. Restart the computer.

Configure the System Management Server in the Configurator

The **System Management Server (SMS)** is part of the System Platform common platform services, and is used to implement important security measures for System Platform 2023. These include:

- Setting port numbers for inter-node communications.
- Setting the SuiteLink security mode: Communication over a SuiteLink connection can be configured to use only encrypted (secure) communications, or to allow unencrypted communications, if a secure (TLS) connection cannot be established. SuiteLink is used for a number of different applications in System Platform. For information about configuring SuiteLink security, select the Advanced Configuration button and go to the Communications Tab.
- Certificate management.
- User authentication via the OpenID connect standard, which allows single sign on (SSO) via an external identity provider.

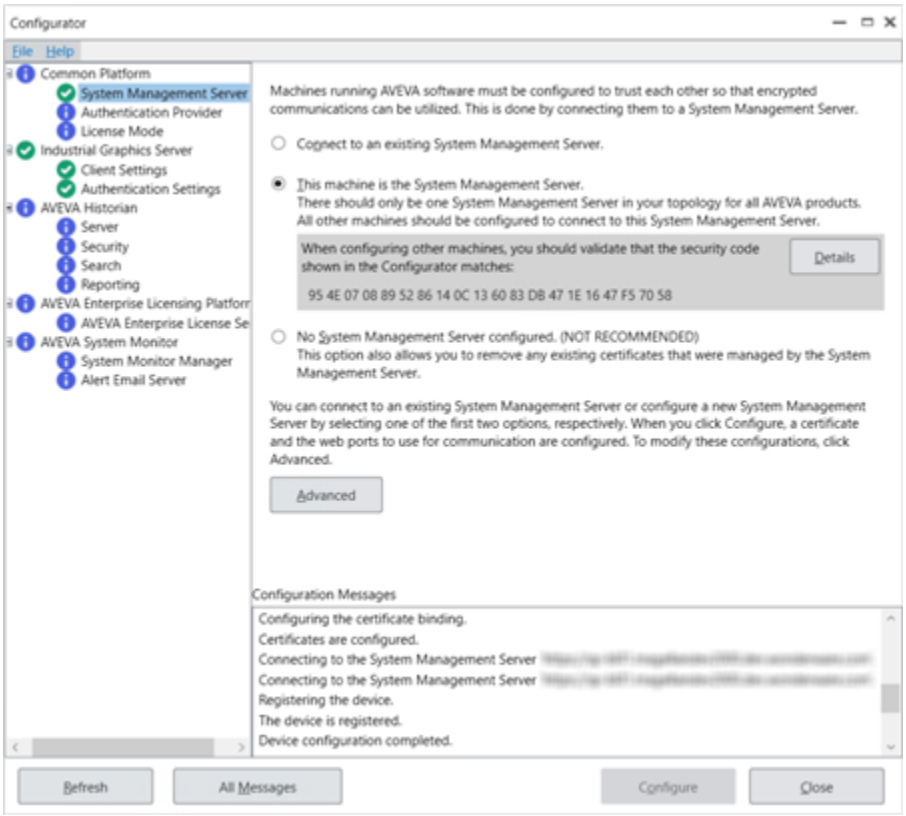
To enable security, every System Platform node must communicate with the System Management Server. There should only be one System Management Server in your System Platform topology, otherwise, communication disruptions may occur. The System Management Server stores shared security certificates and establishes a trust relationship between machines. You can configure one additional node as a redundant SSO server, which functions as a backup for single sign-on if the System Management Server cannot be reached.

If some nodes have not been upgraded to System Platform 2017 Update 3 or later, communication with those older nodes may need to utilize unsecure communication. However, communication between nodes running System Platform 2017 Update 3 or later will be encrypted, as long as the nodes are configured to communication with the System Management Server.

Configure the System Management Server

1. In the Configurator, expand **Common Platform**, and select **System Management Server**.

   Note: If you are prompted for user credentials for the System Management Server, use the following format to enter the user name: **DomainName\UserName**. The prompt for user credentials may be displayed if you have domain admin privileges but are not an admin on the local machine. You must be a member of the **Administrators** or **aaAdministrators** OS group to configure the System Management Server. For more information, see User Credentials for Configuring the System Management Server In the AVEVA System Platform help.



   Note: The Configurator is automatically invoked when installation completes. You can also start the Configurator at any time after from the Windows Start menu on any System Platform node.

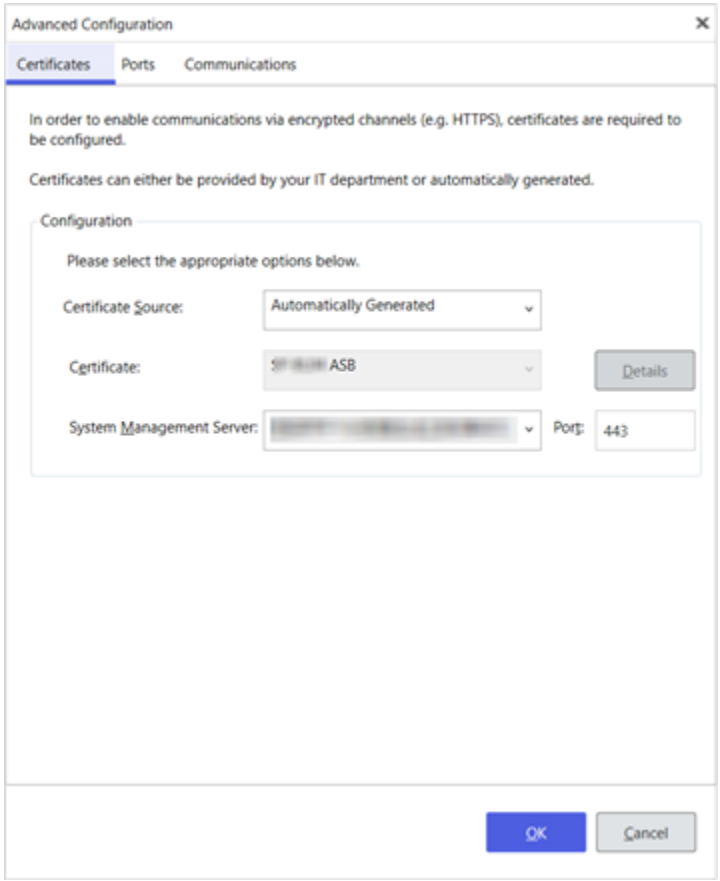
2. You are presented with three choices:
   - **Connect to an existing System Management Server:** This is the default option. The System Platform discovery service looks for any existing System Management Servers on its network. If any are found,

they will be displayed in a drop down list. Select the server you want to use, or enter the machine name of the server. All computers in your System Platform topology should connect to the same server.

- **This machine is the System Management Server:** Select this option if this computer will be the System Management Server. All other computers in your System Platform topology should the be configured to connect to this server by using the **Connect to an existing System Management Server** option.
- **No System Management Server configured. (NOT RECOMMENDED):** Select this option to set up your computer without encryption and secure communications. You can still configure other computers in the topology to use a System Management Server.

3. Select the **Advanced** button.

   The **Advanced Configuration** window opens.



   a. Configure the certificates parameters in the **Certificates** tab.

      a. **Certificate Source:** Select either **Automatically Generated** (default), or **Provided by IT.** If your IT department is providing the certificate, press the **Import** button and navigate to the certificate file. For more information, see "Import a certificate" in System Platform Installation.

      b. **Certificate:** The certificate name is displayed. Select **Details** to view an imported certificate. The certificate is periodically renewed through an automatic update process, both on the server node and on remote nodes.

      c. **System Management Server:** If you are connecting to an existing System Management Server, the name and port number of the server you selected is shown.

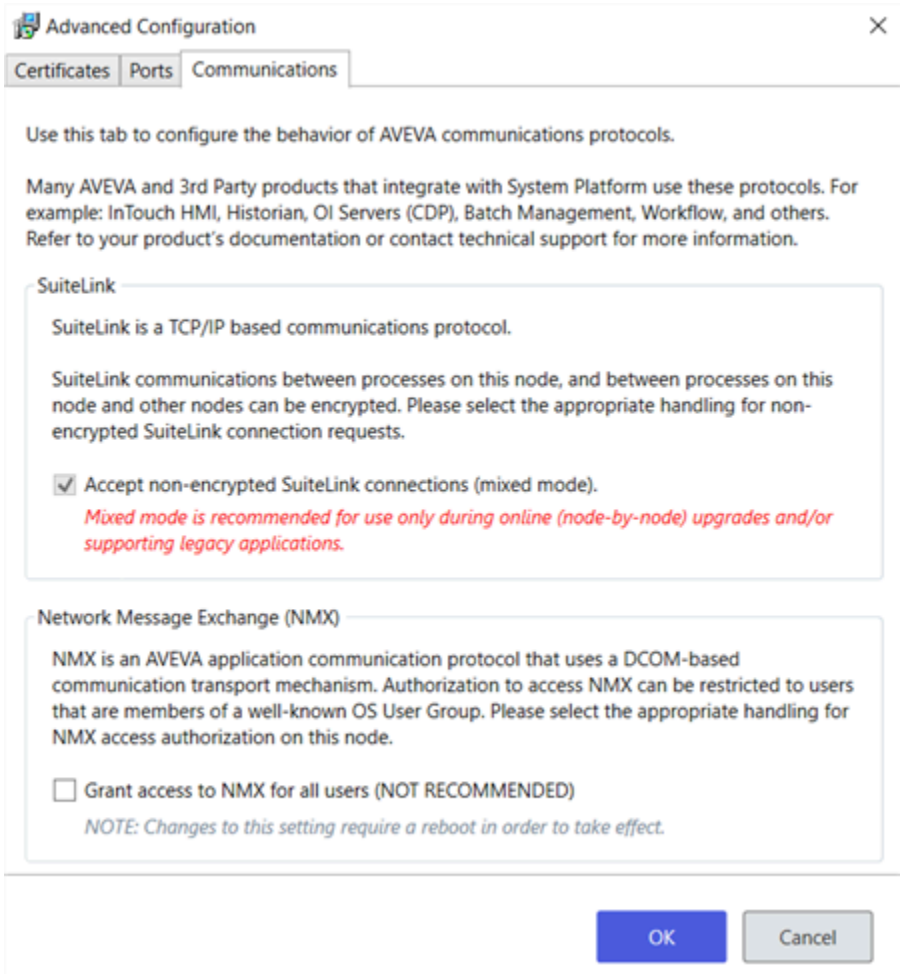
      d. **Common Platform Ports:** The ports for the common platform are used for communications with certain AVEVA software, such as the Sentinel System Monitor. Generally, you can use the default settings. Remote nodes must be configured with the same port numbers as configured here.

Default HTTP port: 80
Default HTTPS port: 443

For more information refer to the Advanced Configuration Options section of AVEVA System Platform Installation.

b. Configure the communications parameters in the **Communications** tab.

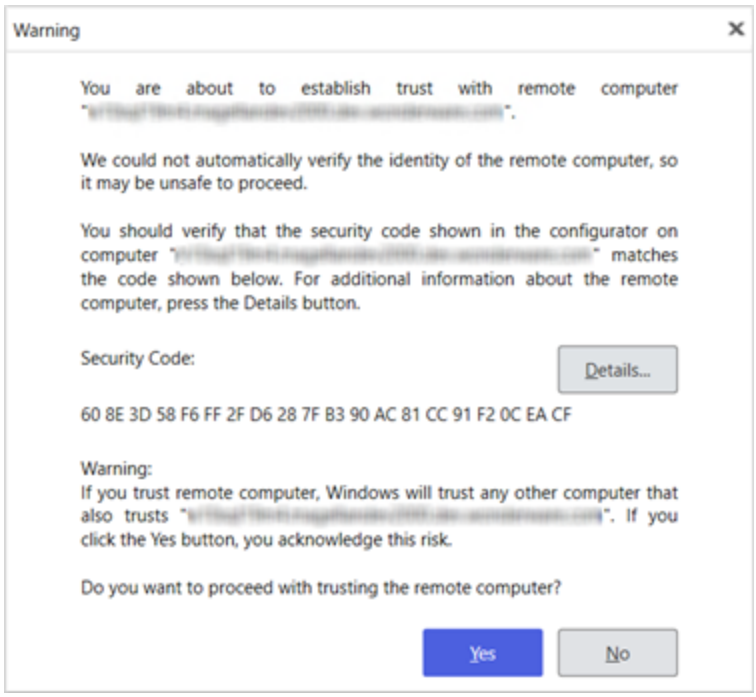
a. Select the **Accept non-encrypted SuiteLink connections (mixed mode)** checkbox and select **OK**. This enables the mixed mode. That is , both encrypted and non-encrypted connections are accepted.



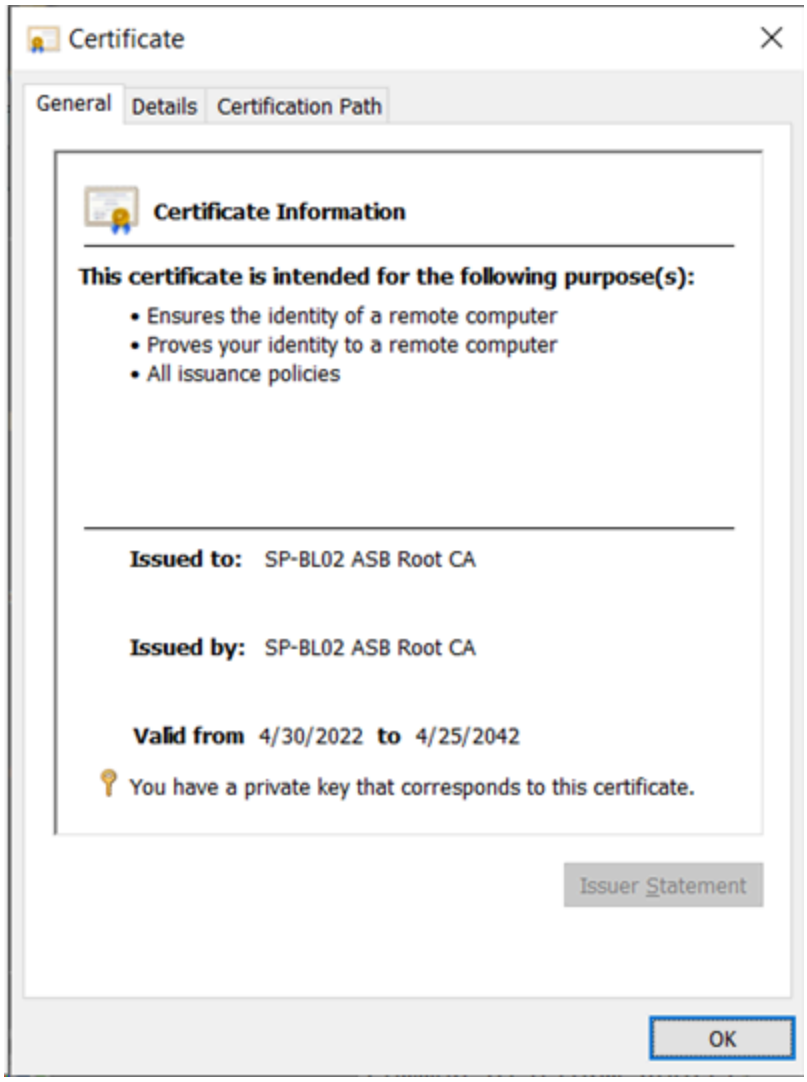
Note: If you want only the encrypted connection (V3), then unselect the **Accept non-encrypted SuiteLink connections (mixed mode)** checkbox, and select **OK**.

4. Select **OK** to save your changes, and return to the SMS configuration window.
5. Select **Configure**.

   A Security Warning window is displayed:



By establishing trust between machines, communications can pass freely. This will be a security concern if you are not sure of the identity of the remote computer. If you have any doubt about the computer you are connecting to, verify the security code and certificate details by selecting the **Details...** button in the Advanced Configuration dialog to open the certificate.



6. Select the next item in the left pane that requires configuration. When all required items have been configured, press the **Close** button to complete installation. See "System restart after configuration" in Application Server User documentation.

For more information on System Management Server configuration, refer to the "Configure the System Management Server" section of System Platform Installation.

Access the server as a standard user

When accessing the server as a standard user, you cannot establish a secure SuiteLink channel. For a secure, encrypted communication workflow, ensure one of the following:

- **Add standard user to the user group:** The standard user should be added to the 'ArchestrAWebHosting' user group on the server side. For more information about adding users to user-groups, refer to the Windows-specific documentation.
- **Run WindowViewer as a service:** For InTouch tag-based applications in a Tag Server architecture, you can achieve a secure encrypted Suitelink communications by running InTouch WindowViewer as a service at the

tag server node.

- **Run WindowViewer in interactive mode with administrator privilege:** This allows WindowViewer to run under a non-interactive user acting as a Suitelink Server with the necessary user privileges to access the private keys with which SuiteLink Server – Client communications are encrypted, something not possible when using an interactive user. This is necessary because the user under which WindowViewer is running provides the security context used to retrieve the private keys required by the SuiteLink server for secure communications. If WindowViewer is run as an interactive application, then the user is an interactive user which, as described previously, cannot access the private keys for secure encrypted Suitelink communications.

Running InTouch as a tag server as a service can be coupled with InTouch Tag Server Client licenses at the client nodes, which limit InTouch to communicate with a tag server only. Users working in this scenario commonly utilize PC virtualization technologies like HyperV or VMWare as examples, and run WindowViewer as a service within that VM, to reduce the Total Cost of Ownership of their solution.

Run WindowViewer as a standard user

WindowViewer is said to be running as a standard user without administrative privilege in the following scenarios:

- Launch System Platform IDE as a standard user without administrative privilege, edit $InTouchViewApp in WindowMaker, and then fast switch to WindowViewer.
- Launch WindowViewer either from AVEVA Application Manager or launch it directly as a standard user without administrative privilege .

Troubleshoot SuiteLink communication problems

If you encounter SuiteLink communication:

- Confirm that Microsoft TCP/IP is operational on the computer on which the InTouch HMI is installed.
- Verify the computer node name is 15 characters or fewer.
- Confirm that SuiteLink is running as a service on the computer on which the InTouch HMI is installed.
- SuiteLink is automatically installed during the InTouch installation. The SuiteLink service starts automatically. If the SuiteLink service stops, you must start it again.

Suitelink communication problems when accessing the server as a standard user

The table below shows the Suitelink connection behavior for a non-admin user (standard user) in different versions of System Platform and the corresponding troubleshooting steps.

Scenario	Until System Platform 2020 R2 SP1	System Platform 2023	Troubleshooting
SMS not configured, run WindowViewer	No Logger message	No Logger message	NA

SMS not configured, connect client to WindowViewer	Unsecure connection established to WindowViewer. Warning message in Logger.*	Connection fails. Warning message in Logger.*	• In the configurator, select the option to allow unsecure connection. Unsecure connection gets established, with the warning message in Logger. • Configure Secure Suitelink connection.
SMS configured, run WindowViewer	Connection fails. Error message in Logger.**	Connection fails. Error message in Logger.**	• Run WindowViewer as service • Run WindowViewer in interactive mode with administrator privilege, or • Add the WindowViewer user to the "ArchestrAWebHosting" group.
SMS configured, connect client to WindowViewer	Unsecure connection established to WindowViewer. Warning message in Logger.*	Connection fails. Warning message in Logger.*	• In the configurator, select the option to allow unsecure connection. Unsecure connection gets established, with the warning message in Logger. • Configure Secure Suitelink connection
SMS is not configured, Secure Suitelink Connection (V3) not configured, and Unsecure Suitelink Connection (V2) as fallback disabled	• Suitelink client connects to WindowViewer. • Web Client binds to InTouch tag. • Warning message in Logger.*	• Suitelink client does not connect to WindowViewer. • Web Client does not bind to InTouch tag. Error message in Logger when	• In the Configurator, configure the SMS. • Configure Secure Suitelink connection. • Select the option to allow unsecure connection as

		WindowViewer is launched.**	fallback.

Warning message* - Warning WWSLS Non-Encrypted communication channel established from 127.0.0.1(VIEW). Secure this channel by installing System Management Server on both nodes and upgrading to the latest version of both client and server products

Error message** - Error WWSLS [Multi-Line Message] - Encrypted inbound SuiteLink connections are not possible in this configuration.

OPC

OPC is not itself a communication protocol, but functions as a driver—a non-proprietary set of standard interfaces, based on Microsoft's OLE/COM technology. This standard makes possible interoperability between automation/control applications, field systems/devices, and business/office applications.

Avoiding the traditional requirement of software application developers to write custom drivers to exchange data with field devices, OPC defines a common, high-performance interface that permits this work to be done once, and then easily reused by HMI, SCADA, control and custom applications.

Over a network, OPC uses DCOM (Distributed COM) for remote communications.

OPC UA

OPC Unified Architecture (OPC UA) is an industrial machine-to-machine communication protocol for interoperability. It provides process control with enhanced security, advanced communication, security, and information models, and cross-platform connectivity.

OPC UA is implemented as a client in OI Gateway.

OPC UA differs significantly from OPC. The following provides the key differences between classic OPC and OPC UA.

Classic OPC	OPC UA
Uses the COM/DCOM technology of Microsoft to communicate. It does not have configurable time-outs. It depends on the DCOM time-out, which is configured in the system.	Uses a services architecture to export data, which improves the ease of communication and connectivity.
Is dependent on Windows operating systems.	Is platform independent and can connect to a wide variety of devices and platforms.
Has limited security.	Has built-in security.
No built-in capabilities to handle problems, such as lost messages.	Has built-in capabilities to handle problems, such as lost messages.

MQTT

MQTT, formerly called Message Queuing Telemetry Transport, is a publish/subscribe messaging protocol for use over TCP/IP. MQTT is designed to ensure that devices can communicate with each other while minimizing power and bandwidth requirements. It is a simple messaging protocol that is well-suited for use with devices that rely on slow or unreliable networks.

The MQTT protocol is an application layer specification, and has been published as standard ISO/IEC PRF 20922. MQTT uses a Publish-Subscribe mechanism which requires a mediating broker. The publishers send data to the broker, and subscribing clients receive data published to the broker. Only clients that have subscribed to a particular topic receive messages about that topic. The protocol supports bidirectional communication such that a device that is a publisher can also receive updates.

Set up access names

You must associate InTouch I/O tags or remote tag references with an Access Name. An Access Name defines a communication link with another I/O data source. Each Access Name specifies an I/O address consisting of a node name, an application name, and a topic.

In a distributed application, I/O references can be set as global addresses to a network I/O Server, or local addresses to a local I/O Server.

InTouchView shows the visual interface of HMI applications designed specifically for use in an Application Server environment. InTouchView applications run as a client with Application Server acting as a server that provides most HMI functionality.

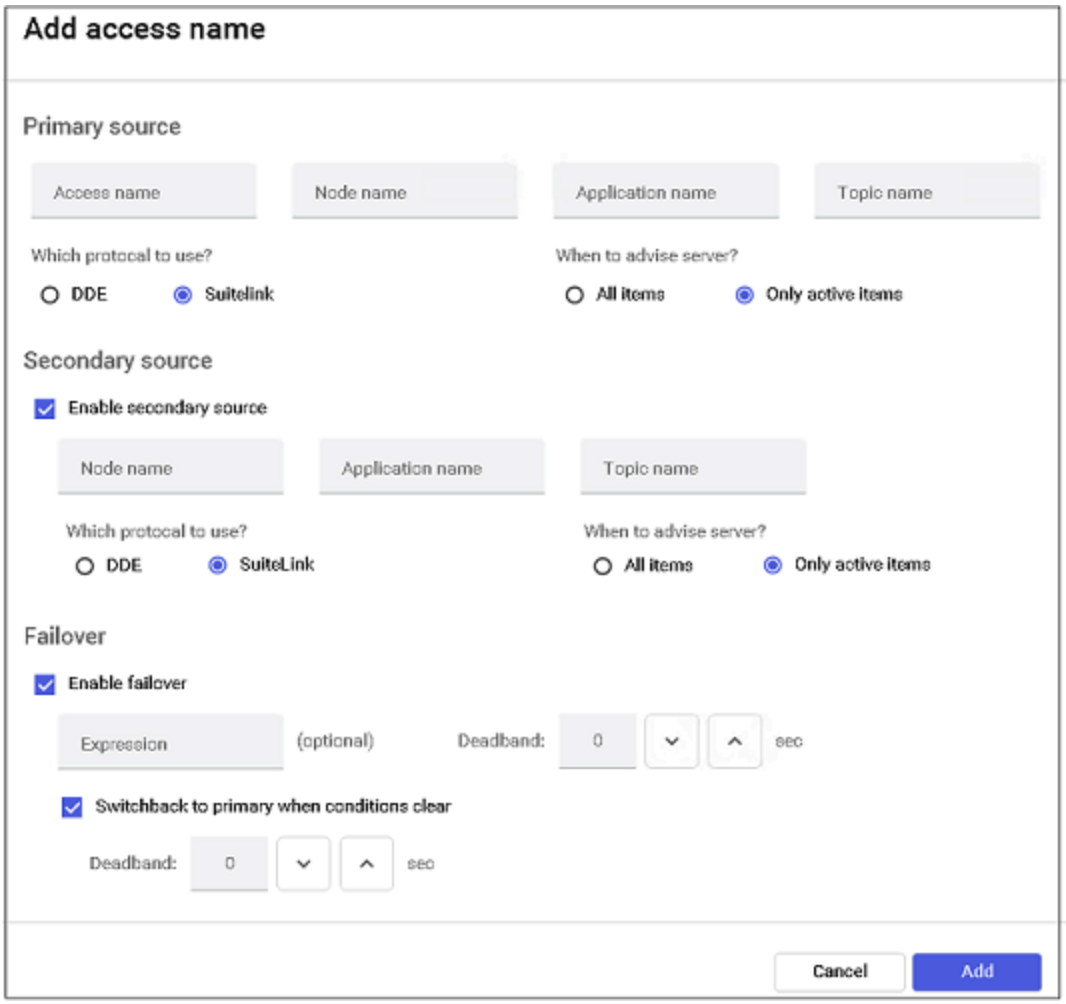
InTouchView applications offer only some of the standard functions available from full-featured InTouch applications. InTouchView applications cannot connect to I/O sources other than the Application Server Galaxy. You can only use the default Galaxy Access Name with your InTouchView application and cannot create Access Names.

Create an access name

1. On the **Home** menu, in the **Tags** group, select **Access Names**.

   The **Access names** dialog box appears.
2. Select **Add**.

   The **Add access name** dialog box appears.



3. Set the properties of the **Add access name** dialog box. Do the following:
   - In the **Access name** box, type a name that identifies this Access Name.
   - If the data resides on a network I/O Server, type the remote server's node name in the **Node name** box.
   - In the **Application name** box, type the actual program name of the I/O Server program from which data will be acquired.

     If the I/O data source is a DAServer, type the name of the DAServer program, do not include the .exe file name extension of the program.
   - In the **Topic name** box, type the topic name you want to access.

     The topic name is an application-specific sub-group of data elements. In the case of data coming from a DAServer program, the topic name is the same name configured for the topic in the DAServer program. When communicating with Microsoft Excel, the topic name must be the name assigned to the book and spreadsheet when it was saved. For example, [Book1]Sheet1.
   - Select the communication protocol to communicate with the I/O Server.
   - Select the option to poll information stored on the server.

Option	Definition
Advise all items	Polls for all data whether or not it is in visible windows, alarmed, logged, trended, or used in a script. Selecting this option affects performance, so its use is not recommended.
Advise only active items	Polls for data shown in visible windows and data that is alarmed, logged, trended, or used in any script. **Note**: A button action script is not polled unless it appears in a visible window.

4. Select **Enable Secondary Source** if you want to select a secondary back up server, and enter the required details for the secondary source back up server.
5. When you are done specifying the Access Name, select **Add**.

   The new Access Name is added to the list.
6. Select **Close**.

For more information about setting up your secondary server for failover switching, see Use failover functionality with Access Names.

Delete access names

You can delete an Access Name that you no longer need. Before you delete an Access Name, make sure:

- No tags are associated with the Access Name.
- WindowViewer is stopped.

Delete an Access Name

1. On the **Home** menu, in the **Tags** group, select **Access Names**.

   The **Access Names** dialog box appears. The current Access Names are listed.
2. To delete an Access Name, highlight it on the list and select **Delete**.

   A message appears requesting confirmation that the Access Name should be deleted.
3. Select **Yes**.
4. Select **Close** or repeat this procedure if you need to delete other defined Access Names.

Access I/O data with I/O tags

The InTouch HMI can send and receive data from local or remote Windows applications with I/O tags. Each I/O type tag references a valid item in the I/O Server program. You can define different types of I/O tags in the Tagname Dictionary.

Configure I/O tag properties

You define the different types of I/O tags in the Tagname Dictionary.

Specify a discrete I/O Tag

I/O discrete tags indicate the binary state of all inputs and outputs from programmable controllers, process computers, and data from network nodes.

An I/O discrete tag must be assigned an initial value of On or Off. You can also configure the discrete I/O tag to toggle to the opposite value of its binary source. You can specify messages that appear in the alarm event window when the process associated with the tag goes into an alarm state.

For more information about the overall procedure to create a tag from the Tagname Dictionary, see Create new tags.

Define an I/O discrete tag

1. Open the Tagname Dictionary and assign a name for a new tag.
2. Assign the tag as an I/O discrete type from the **Tag Types** dialog box. The detail portion of the **Tagname Dictionary** dialog box appears.

3. Do the following:
   - Select **On** or **Off** as the initial value associated with the tag.

     The InTouch HMI assigns this value to the tag when the application starts, but does not write this initial value to the I/O device.
   - Select **Direct** or **Reverse** as the input conversion applied to the value received from a remote I/O tag.

Input Conversion	Description
Direct	The input value is read unchanged directly from the I/O Server program.
Reverse	The I/O discrete value is toggled when read from the server program. For example, if the I/O input is 0, the value is automatically set to 1.

4. Enter messages in the **On Msg** and **Off Msg** boxes that appear when the tag transitions in and out of an alarm state.

   These messages are available for use in any animation link or script, regardless of whether the tag has alarms configured or not.
   - If you define a discrete alarm that is active when the tag value is equal to 1 (On, True), the message entered in the **On Msg** box appears in the **Value** and **Limit** columns of your ActiveX alarm displays.

     When the tag's alarm state returns to normal, the message entered in the **Off Msg** box appears in the

Value column and the On message remains in the **Limit** column.

- If you define a discrete alarm that is active when the tag value is equal to 0 (Off, False), the message entered in the **Off Msg** box appears in the **Value** and **Limit** columns of your ActiveX alarm displays.

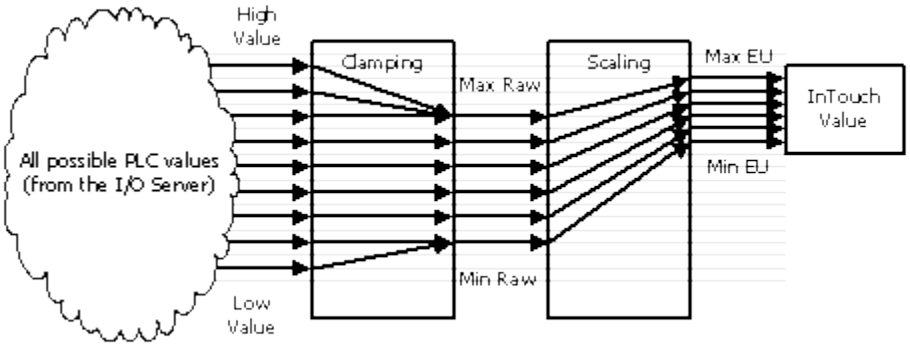
  When the tag's alarm state returns to normal, the message entered in the **On Msg** box appears in the **Value** column and the Off message remains in the **Limit** column.

5. Save your changes to the tag.

Specify integer and real I/O tags

You must assign I/O integer and real tags a set of attributes that characterize numerical data sent between the InTouch application and external processes.

The InTouch HMI normalizes raw input data from a PLC. The figure below shows the process of clamping raw I/O data values and then scaling them to engineering units that can be shown from an InTouch application.



I/O integer and real tags include attributes that set minimum and maximum limits for raw input data sent by the PLC. The InTouch HMI clamps I/O values that are below or above the raw value range. Clamping reassigns values outside of the range to the minimum or maximum raw values.

I/O integer and real tags include attributes that scale clamped raw values within a range of engineering units. Minimum and maximum engineering unit attributes set the upper and lower boundaries of the scaled values.

When you define integer and real I/O tags, you specify the type of conversion to scale raw values when calculating engineering units. You can select Linear or Square Root.

For linear scaling, the result is calculated using linear interpolation between the minimum and maximum end points. The algorithm for linear scaling of input is:

```
EUValue = (RawValue - MinRaw) * ((MaxEU - MinEU) / (MaxRaw - MinRaw)) + MinEU
```

The algorithm for linear scaling of output is:

```
RawValue = (EUValue - MinEU) * ((MaxRaw - MinRaw) / (MaxEU - MinEU)) + MinRaw
```

For square root scaling, the minimum and maximum raw values are used for interpolation. This is useful for scaling inputs from nonlinear devices like pressure transducers. The algorithm for square root scaling of input is:

```
EUValue = sqrt(RawValue - MinRaw) * ((MaxEU - MinEU) / sqrt(MaxRaw - MinRaw)) + MinEU
```

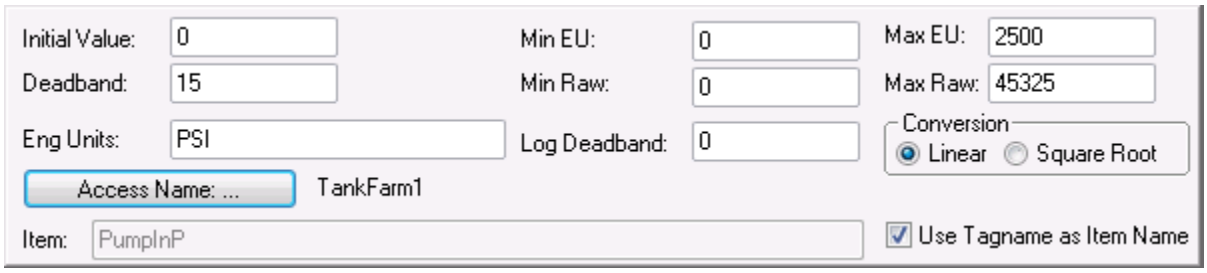
The algorithm for square root scaling of output is:

```
RawValue = square((EUValue - MinEU) * (sqrt(MaxRaw – MinRaw) / (MaxEU -MinEU))) + MinRaw
```

Define integer and real I/O tags

1. Open the Tagname Dictionary and assign a name for a new tag.
2. Assign I/O integer or I/O real as the type of tag from the **Tag Types** dialog box. The detail portion of the

Tagname Dictionary dialog box appears.



3. Do the following:
   - In the **Initial Value** box, type the integer or real number associated with the tag when the application starts.

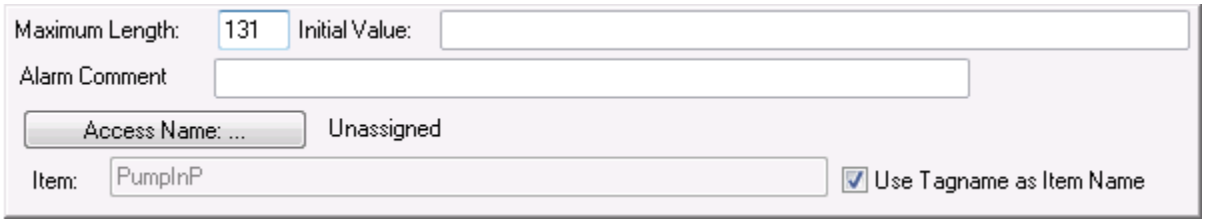
     The application does not write this initial value to the external process.
   - In the **Min EU** box, type the minimum engineering unit for the tag.
   - In the **Max EU** box, type the maximum engineering unit for the tag.
   - In the **Min Raw** box, type the minimum value of the low clamp for raw I/O integer or real numbers.
   - In the **Max Raw** box, type the maximum value of the high clamp for raw I/O integer or real numbers.
   - In the **Eng Units** box, type the label to use for the tag's engineering units.
   - Select **Linear** or **Square Root** as the type of conversion to scale raw values when calculating engineering units.
4. Save your changes to the tag.

Specify a message I/O tag

You can specify an I/O message tag options that specify the network address of remote processes. Its message properties are the same as a memory message tag.

Define memory and I/O message tag values

1. Open the Tagname Dictionary and assign a name for a new tag.
2. Select **I/O Message** as the type of tag from the **Tag Types** dialog box. The detail portion of the **Tagname Dictionary** dialog box appears.



3. In the **Maximum Length** box, type the maximum number of characters allowed in the tag's message.

   You can enter messages to a maximum of 131 characters.
4. In the **Initial Value** box, type the text string that you want shown when WindowViewer starts the application.
5. Save your changes to the tag.

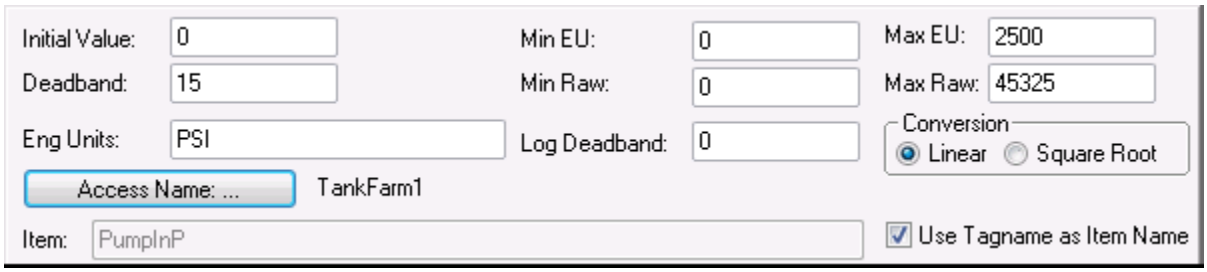
Set I/O access parameters

You can set the I/O attributes of tags in the Tagname Dictionary. These attributes identify the external data source associated with the tag.

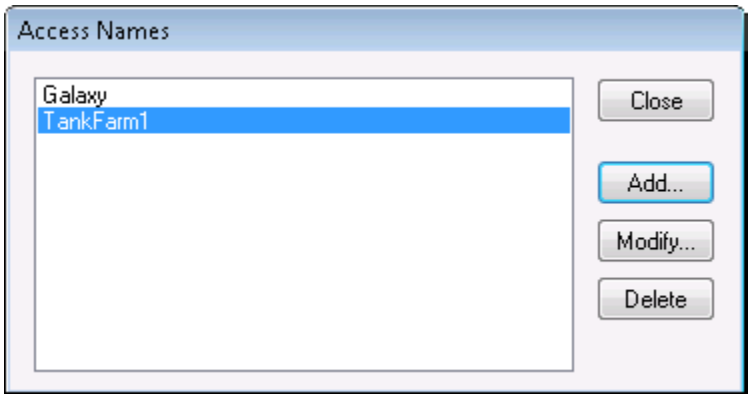
These steps only explain how to specify I/O attributes from the Tagname Dictionary. For more information about configuring Galaxies and Access Names, see Data access with I/O.

Set tag I/O attributes

1. Assign an I/O tag type to the tag from the **Tag Types** dialog box. The detail portion of the **Tagname Dictionary** dialog box appears.



2. Select **Access Name** to define or select the Access Name assigned to the tag. The **Access Names** dialog box appears showing a list of current Access Names recognized by the InTouch HMI. (Galaxy is the default Access Name for an ArchestrA connection.)



3. Add an Access Name or accept the default.
4. Select the data point in the server program that the I/O tag will read and write data.
   - To read and write data to and from a process data point in the server program, type the Item Name in the **Item** box. For example, to read a value from a register in a PLC, type the identity of the register as the Item Name. For example:

     To use the register 1 of an Allen-Bradley PLC as the Item Name, type R1 in the **Item** box.

     To use the least significant bit of register 1 of an Allen-Bradley PLC as the Item Name, type R1:0 in the **Item** box.
   - To use the tag as the item, select **Use Tagname as Item Name**. Item Name supports a maximum of 254 characters.

Retrieve information about I/O tags at run time

You can write scripts with functions that return the names of the node, application, and topic specified in an

Access Name definition.

IOGetNode() Function

The **IOGetNode()** function returns the node address defined for a specific Access Name to a tag associated with the function in the script.

Category

Miscellaneous

Syntax

```
IOGetNode("AccessName");
```

Argument

AccessName

The existing Access Name to return node information for.

Remarks

You can specify the Access Name as a literal string, or as a string value provided by other InTouch tags or functions.

Example

The following example returns the node information for the ModbusPLC1 Access Name to the **NodeName** tag.

```
NodeName = IOGetNode("ModbusPLC1");
```

IOGetApplication() Function

The **IOGetApplication()** script function returns the application name defined for a specific Access Name to a tag assigned as an argument of the function.

Category

Miscellaneous

Syntax

```
IOGetApplication("AccessName");
```

Argument

AccessName

The existing Access Name in which the application is defined.

Remarks

You can specify the Access Name as a literal string, or as a string value provided by other InTouch tags or functions.

Example

The example returns the name of the application specified for the ModbusPLC1 Access Name to the **AppName** tag.

```
AppName = IOGetApplication ("ModbusPLC1");
```

IOGetTopic() Function

The **IOGetTopic()** script function returns the topic name defined for a specific Access Name to a tag associated with the function in the script.

Category

Miscellaneous

Syntax

```
IOGetTopic("AccessName");
```

Argument

AccessName

The Access Name whose topic name is returned.

Remarks

The Access Name can be specified as a literal string, or it can be a string value provided by other InTouch message tags or functions.

Example

This example returns topic information for the ModbusPLC1 Access Name to the **TopicName** tag.

```
TopicName = IOGetTopic("ModbusPLC1");
```

Dynamically change I/O tag references at run time

The InTouch HMI uses dynamic references to view data points whose values are only needed temporarily, such as in diagnostic applications. Using Dynamic Reference Addressing allows you to address multiple data sources with a single tag.

You can use several methods to dynamically reference multiple data sources with a single tag:

- Assign different Access Name characteristics with the .**Reference** dotfield of an I/O tag
- Use the **IOSetItem()** script function to set an I/O tag's .**Reference** dotfield
- Use the **IOSetAccessName()** script function to change the characteristics of an Access Name during run time

.Reference Dotfield

You can implement Dynamic Reference Addressing by assigning a valid reference to the .**Reference** dotfield of an I/O tag. You can use the **.Reference** dotfield to dynamically change the data source by modifying the characteristics of the Access Name assigned to the I/O tag.

The syntax of the .**Reference** dotfield is:

tag.Reference="accessname.item"	Changes the Access Name and item assigned to a tag.
tag.Reference="[.]item"	Changes the item assigned to an I/O tag.
tag.Reference="accessname."	Changes the Access Name of an I/O tag.
tag.Reference=""	Deactivates the I/O tag.

Each I/O type tag has a .**ReferenceComplete** dotfield. The value of this discrete dotfield indicates if the item requested in the .**Reference** dotfield is reflected in the .**Value** dotfield.

The .**ReferenceComplete** field is set to false (0) when the application starts in WindowViewer. When it is confirmed that the .**Value** dotfield is being updated by the source specified in the .**Reference** dotfield, the .**ReferenceComplete** value is set to true (1). If the .**Reference** dotfield is changed, the **.ReferenceComplete** dotfield is automatically set to false (0), and then updated to true (1) when the new value is updated.

IOSetItem() Function

You can implement Dynamic Reference Addressing by using the **IOSetItem()** function within a script. **IOSetItem()** includes arguments to change the values assigned to the .**Reference** dotfield of an I/O tag during run time.

Category

Miscellaneous

Syntax

```
IoSetItem ("Tag", "AccessName", "Item");
```

Arguments

Tag

Any InTouch I/O tag enclosed in quotation marks.

AccessName

The Access Name assigned to the I/O tag.

Item

The Item assigned to the I/O tag.

The Tag, AccessName, and Item arguments can be specified as literal strings or they can be string values provided by other InTouch tags or functions.

Examples

In the following example, the .**Reference** dotfield of the **PumpInP1** tag is changed to point to the excel Access Name and the R1C1 item.

```
IOSetItem("PumpInP1", "excel", "R1C1");
```

or

```
Number = 1;
TagNameString = "PumpInP" + Text(Number,"#");
IOSetItem(TagNameString, "excel", "R1C1");
```

If an empty string ("") is specified for both the Access Name and item values, then the tag is deactivated. For example, the **PumpInP2** tag is deactivated by:

```
IOSetItem("PumpInp2", "", "");
```

If a null is specified only for an item, then the tag's current item value is retained and its Access Name value is updated. For example, the following changes the Access Name for the **PumpInP3** tag to excel2 without affecting its current Item value:

```
IOSetItem("PumpInP3", "excel2", "");
```

Likewise, if a null string is specified only for an Access Name, then the tag's current Access Name value is retained and its item value is updated. The following example changes the Item for the **PumpInP4** tag to R1C2 without affecting its current Access Name value:

```
IOSetItem("PumpInP4", "", "R1C2");
```

IOSetAccessName() Function

You can implement Dynamic Reference Addressing by using the **IOSetAccessName()** function within a script. **IOSetAccessName()** modifies the application or topic name characteristics of an I/O tag's Access Name during run time.

Note: When the IOSetAccessName() function is processed, there is a time delay while the existing conversation is terminated and the new conversation is started. During this period, any attempted POKEs or writes to the new topic are lost.

Category

Miscellaneous

Syntax

```
IOSetAccessName("AccessName", "NodeName", "AppName", "TopicName");
```

Arguments

AccessName

The existing Access Name to assign the new AppName and Topic Name values to. Actual string or message tag.

NodeName

The new Node Name to assign. Actual string or message tag.

AppName

The new Application Name to assign. Actual string or message tag.

TopicName

The new Topic Name to assign. Actual string or message tag.

Remarks

The values assigned to the AccessName, AppName, and TopicName arguments can be specified as literal strings or string values provided by other InTouch tags or functions.

If you configured a secondary source for access name failover using the **Add Access Name** dialog box, you cannot change the secondary source configuration at run time using the IOSetAccessName() function.

Note: When creating Access Names in WindowMaker, if the Access Name is SuiteLink type, the InTouch HMI prevents Access Names from accessing the same, node, application, and topic. Do not allow the **IOSetAccessName()** function to redirect Access Names to duplicates during run time. Using the **IOSetAccessName()** function in run time allows SuiteLink type Access Names to be redirected to duplicate topics. The redirected Access Name will not work.

Examples

The **MyAccess1** Access Name can be changed to point to the **Excel** application and the [**Book1]Sheet1** topic, without affecting the current NodeName, by using the following script function:

```
IOSetAccessName("MyAccess1", "", "EXCEL", "[Book1]Sheet1");
```

If an empty string is specified for a Topic, the Access Name's current Application value is updated and its Topic value is retained.

For example, the following script changes the Application Name for the **MyAccess2** Access Name to EXCEL without affecting its current Topic value:

```
IOSetAccessName("MyAccess2", "", "EXCEL", "");
```

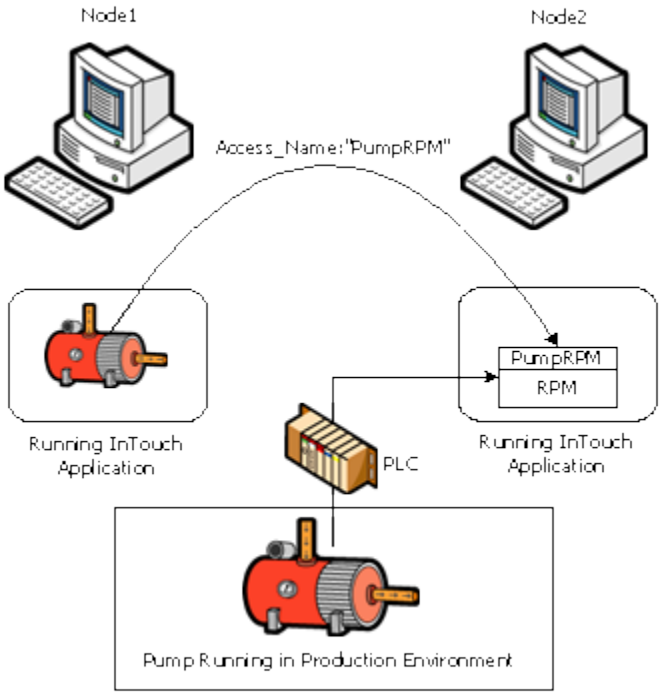
If an empty string is specified only for an Application Name, the tag's current Topic value is updated and its Application value is retained. For example, the following script changes the Topic for the **MyAccess3** tag to [**Book2]Sheet1** without affecting its current Application Name value:

```
IOSetAccessName("MyAccess3", "", "", "[Book2]Sheet1");
```

This example can be used when PLC redundancy is a requirement.

Convert tags to remote references

You can create distributed InTouch applications based upon a client-server architecture. Client applications can run on one network node that use tags defined on other remote nodes. The following figure shows an InTouch application running on Node 1 making a remote reference to the PumpRPM tag on Node 2.



In this example, you can retrieve the value of the PumpRPM tag from Node 2 in two ways:

- Create an I/O type tag in Node1's Tagname Dictionary that uses Node2 as the node name in the Access Name associated with the I/O tag.
- Use a direct remote reference to the **PumpRPM** tag. For example, **PLC1:"PumpRPM"**.

  In a window or QuickScript, you can reference a remote tag by appending the Access Name as the prefix to the remote tag name in the form:

  access_name:"tag_name"

When importing a window or QuickScript, you can convert the placeholder tags to remote tag references. For example, you can convert the placeholder tags to point to the application from which you imported the window. The tags do not need to be defined in your local Tagname Dictionary.

You can use several methods to convert local tags to remote tag references:

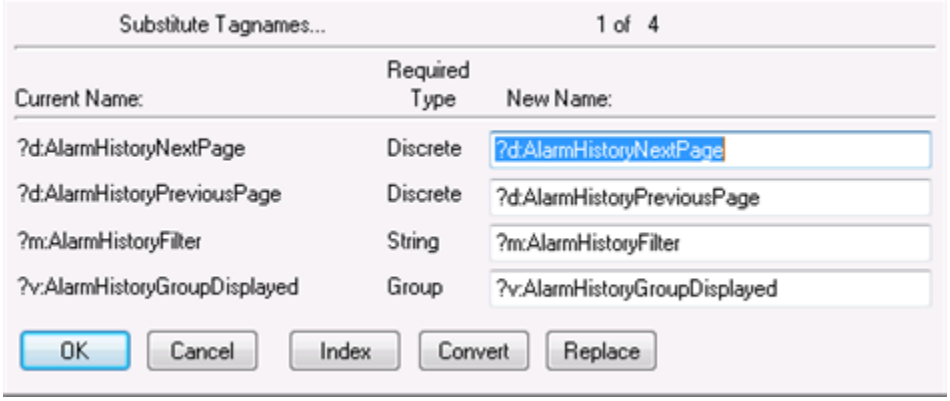
- Append the remote tag reference
- Convert the placeholder tags associated with an imported window
- Launch the Tag Browser and open the tag source's Tagname Dictionary to select the remote tag reference.

Manually convert tags to remote tag references

1. Open an application window in WindowMaker.
2. Select the object associated with the local tag that you want to change to a remote tag reference.

3. On the **Animation** tab, in the **Substitute** group, select **Tags**.

   The **Substitute Tagnames** dialog box appears with a list of tags associated with the object.

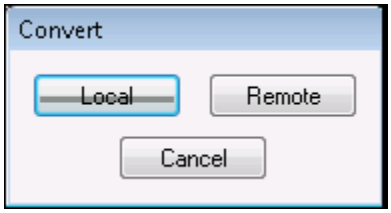


4. Select **Index** to add an index to each tag name.
5. Select **Convert**. The **Convert** dialog box appears with options to convert the tags to local or remote reference tags.
6. Select **Remote**. The **Access Names** dialog box appears. All Access Names defined in your local InTouch application are shown.
7. Select an Access Name from the list.
8. Select **Close**. All tags listed in the **Substitute Tags** dialog box are automatically converted to remote tag references with the Access Name appended to the tag name.
9. Select **OK**.

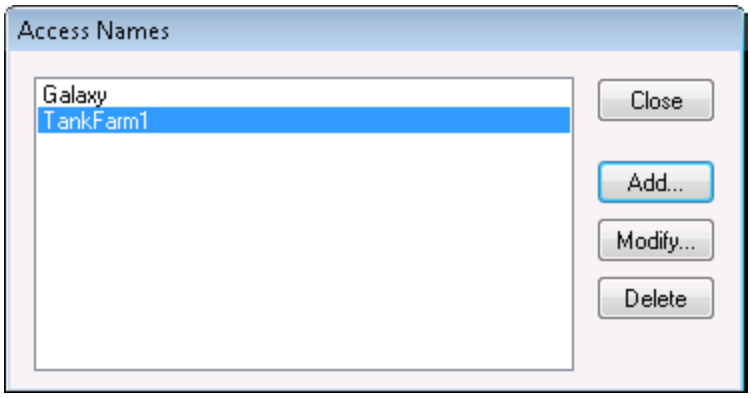
Convert an imported window's tags to remote references

1. Open the imported window and select all objects.
2. On the **Animation** tab, in the **Substitute** group, select **Tags**.

   The **Substitute Tagnames** dialog box appears.
3. Select **Convert**. The **Convert** dialog box appears.



4. Select **Remote**. The **Access Names** dialog box appears. All Access Names defined in your local InTouch application are shown.

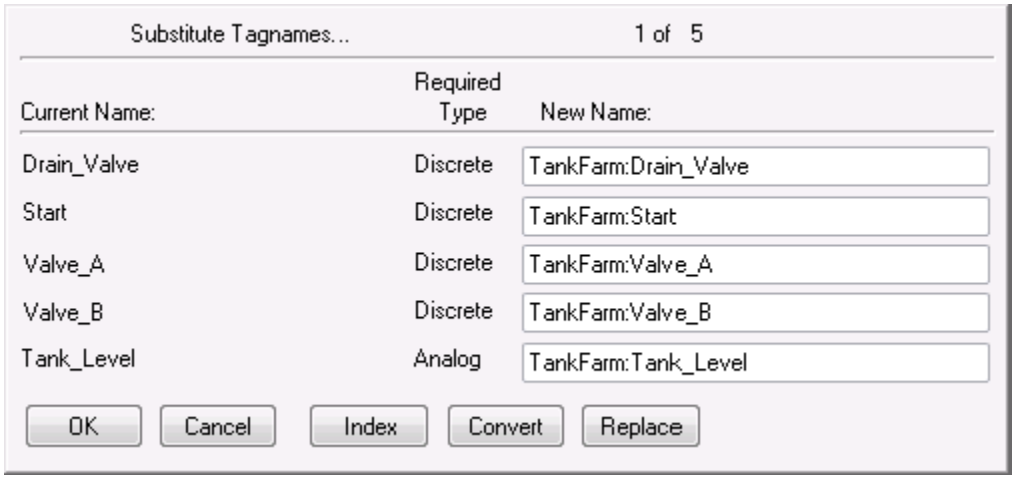


5. Select an Access Name from the list.

   To verify the Access Name is correctly configured, select **Modify**.

   If you do not have an Access Name currently defined that points to the tag source, select **Add** and define it. The Access Name must include the name of the remote node where the application resides.

6. Select **Close**. All tags listed in the **Substitute Tags** dialog box are automatically converted to remote tag references with the Access Name appended to the tag name.

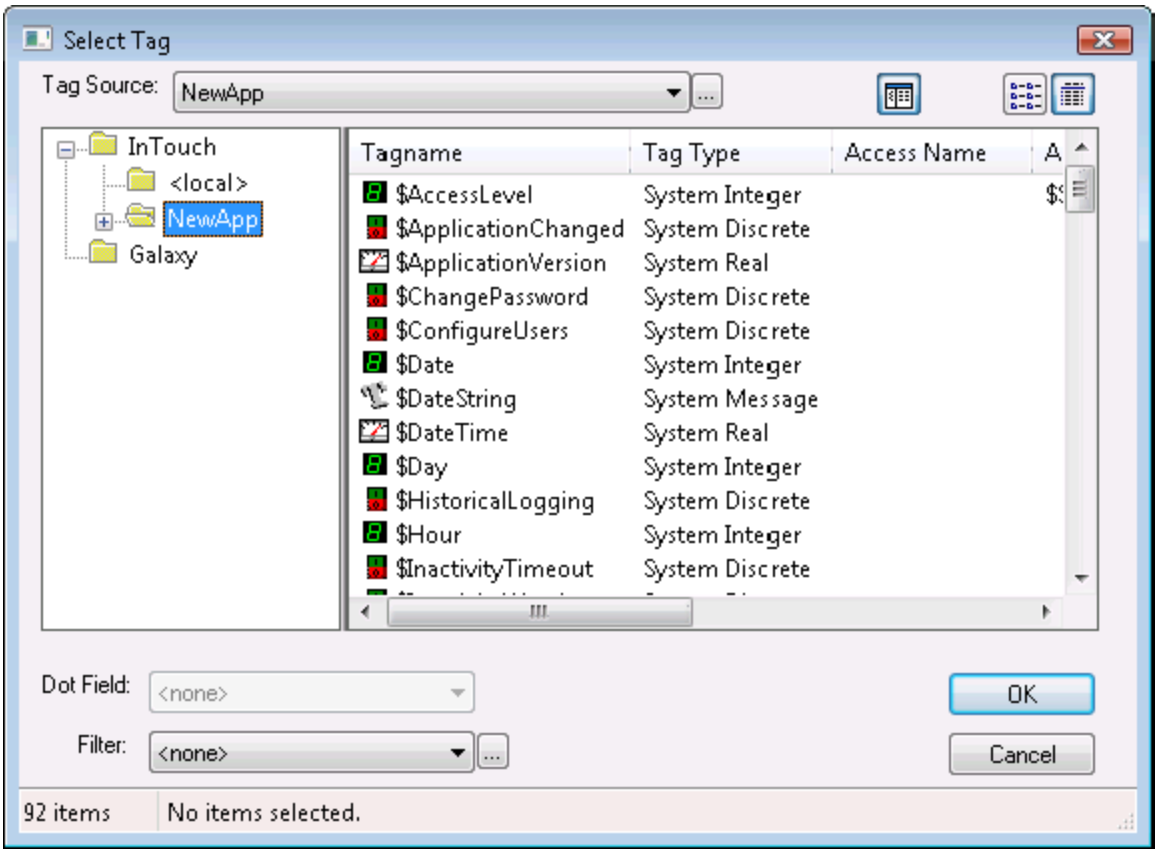


7. Select **OK**.

Select a remote tag reference in the Tag Browser

1. Select the objects associated with the local tag that you want to convert to a remote tag reference.
2. On the **Animation** tab, in the **Substitute** group, select **Tags**.

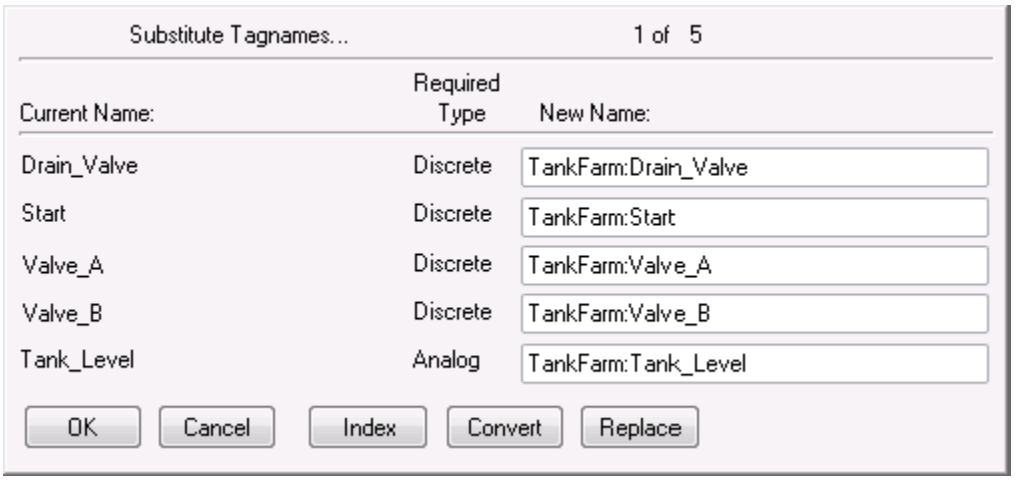
   The **Substitute Tagnames** dialog box appears showing the selected tags.
3. Delete the tag name in the **New Name** box that you want to replace with a remote tag reference.
4. Double-click the **New Name** box. The **Select Tag** dialog box appears with a list of tags associated with the application.
5. Select a remote tag using the Tree view.
   a. Select the **Tree** icon to show a hierarchical list of all local and remote Access Names in the left pane.



b. Select a remote Access Name folder to show its assigned tags in the right pane.

c. Select a remote tag that you want to use as a remote reference.

d. Select **OK**. The **Substitute Tagnames** dialog box appears with the selected remote tag name in the **New Name** box.

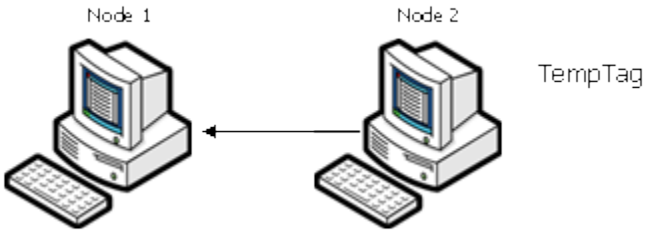


6. Select **OK** to close the dialog box and associate the remote tag with the selected objects.
7. Repeat these steps for each tag that you want to associate with a remote reference.

Access I/O data by remote references

The InTouch HMI supports true client-server architecture for factory automation applications. You can design client applications without using any tags from the local Tagname Dictionary located on the same node as the running InTouch application. You can run an application on one node that uses tags from a remote node by using remote tag referencing.

The following figure shows a simple example where the **TempTag** is defined locally on Node2:



In this example, the InTouch application running on Node1 can retrieve the value of TempTag on Node2 by two methods:

- Create an I/O type tag in Node1's Tagname Dictionary that uses Node2 as the Node Name in the Access Name associated with the I/O tag.
- Use a remote reference directly to **TempTag**. For example, Node2:"TempTag".

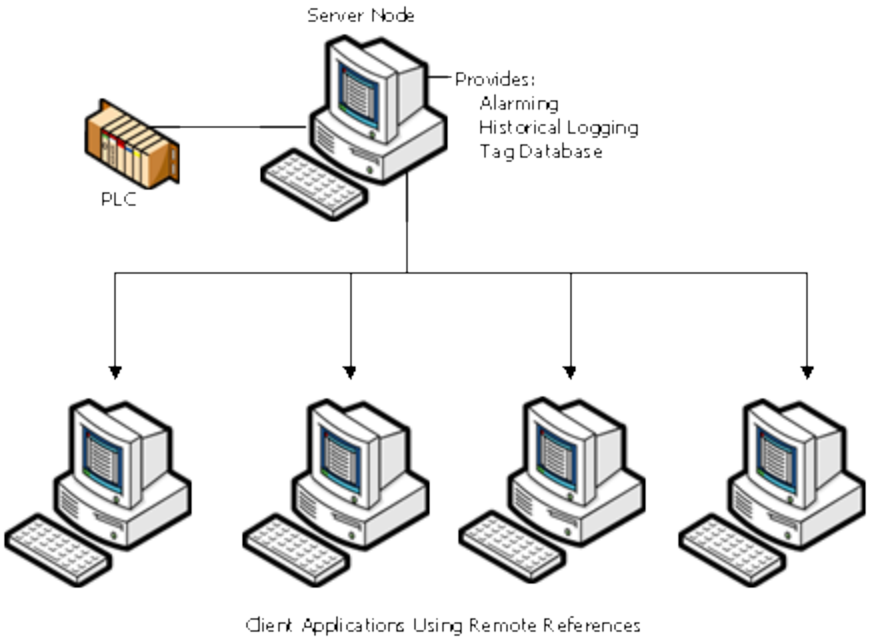
When you want to directly reference a remote tag in any other application, only AccessName:item is required. You do not have to define the remote tag in your local Tagname Dictionary. Remote references can also access data from any I/O data source such as a DAServer or Microsoft Excel.

You can also make a remote reference to SuperTags. The valid syntax for a remote tag reference to a SuperTag is:

Access_name:Parent_Instance\ChildMember\SubMember.

For more information about remote references to SuperTags, see Reference Supertag members.

When importing a window or QuickScript, you can convert the placeholder tags to remote tag references. You do not have to define tags in your local Tagname Dictionary. The remote references are accessible from any application on the network.



Redirect remote references during run time

You can redirect Application Server object references or remote references to InTouch tags at run time with a script. You can switch object instances for a graphic symbol based on certain conditions being met or directly by an operator action.

IOSetRemoteReferences() Function

You can use the **IOSetRemoteReferences()** script function to redirect Application Server object references or remote references to tags while an InTouch application is running. **IOSetRemoteReferences()** finds all remote references that match specified strings and changes those references according to specified argument values. You can create a script that triggers the function to redirect references based on conditions being met or by a user action.

Category

Misc

Syntax

```
IOSetRemoteReferences(BaseAccess, NewAccess, MatchString, SubstituteString, Mode)
```

Arguments

BaseAccess

This string argument specifies the original configured Access Name to match in the references.

NewAccess

The new Access Name. The new Access Name is applied to all references in which the original Access Name matches the string provided with BaseAccess and for which the original Item Name matches the MatchString value if one is specified.

MatchString

The string to match in the original configured Item Name in the references. If the MatchString value is an empty string, it is regarded as a match for any Item Name.

SubstituteString

The string to substitute for the original Item Name. The string replaces the MatchString value to create the new active Item Name for the references. If SubstituteString is an empty string, no substitution is made.

Mode

Determines the method used to compare the original configured Item Name to the MatchString value. Matching is always from the beginning of the Item Name. A Mode value of 0 specifies the match must be for the entire Item Name or up to a period (.) within the name. A Mode value of 1 specifies that a partial match is allowed, even if the next character is not a period.

Remarks

IOSetRemoteReferences() does not check the validity of the new tag or Access Name before changing the object references.

- **IOSetRemoteReferences()** only changes remote references. The function redirects those references in which the original, configured Access Name matches the specified value of the BaseAccess argument, and the original Item Name matches the MatchString value.
- A single call to **IOSetRemoteReferences()** affects all remote references in active windows that are in memory in which the original, configured name strings match the values assigned to the BaseAccess and MatchString arguments.
- If you do not assign a value to the BaseAccess argument, **IOSetRemoteReferences()** does not redirect any remote references.
- If the MatchString argument is empty, **IOSetRemoteReferences()** redirects all remote references in which the original Access Name matches the value assigned to the BaseAccess argument.
- When the Mode argument is set to 0, replacement in the Item Name is only done for full object names (or tags), or full property names (or dotfields). The value of the MatchString argument must match the entire original Item Name or up to a character followed by a period.
- When the Mode argument is set to 1, partial replacement of the item string is allowed when the item string starts with the match item string. That is, MatchString must match some portion of the original item string, but that sub-portion must start at the beginning of the item string. The last character in the matching string does not need to be followed by a period.
- The original, configured names for a remote reference remain unchanged. Subsequent calls to **IOSetRemoteReferences()** do not need to recognize the current active name. Calls to **IOSetRemoteReferences()** can be made in any order.
- If you want two or more windows to refer to one remote reference, that remote reference acts like an I/O tag. When you redirect it, all windows see the same thing. Do not use a single name to refer to two separate targets at the same time.

  Note: Changing many references simultaneously, for example in a Window OnShow script, can take some time before all references are resolved.

Examples

The following example redirects all remote references to the pump001 item name of the Galaxy Access Name if the original item name exactly matches pumpX.

```
IOSetRemoteReferences("Galaxy","","pumpX", "pump001",0);
```

The following example matches changes the Galaxy Access Name to TagServer1 if the item name exactly matches pumpX. Also, the item name is changed to p2.

```
IOSetRemoteReferences("Galaxy","TagServer1", "pumpX","p2",0);
```

The following example changes the TagServer1 Access Name to TagServer2 when the item name is pumpX. Also, the item name is changed to backpump3.

```
IOSetRemoteReferences("TagServer1","TagServer2", "pumpX","backpump3",0)
```

The following example changes the Tank item name of the TagServer1 Access Name to Plant.

```
IOSetRemoteReferences("TagServer1","","Tank", "Plant",1)
```

The following example does not redirect any remote references because no value is assigned to the *BaseAccess* argument.

```
IOSetRemoteReferences("","Galaxy","pumpX", "pump001",0);
```

Restore references

If the NewAccess argument is empty without an assigned value, IOSetRemoteReferences() restores the active Access Name to the original base Access Name.

If the MatchString argument is empty without an assigned value, IOSetRemoteReferenc**es()** restores the active Item Name to the original Item Name.

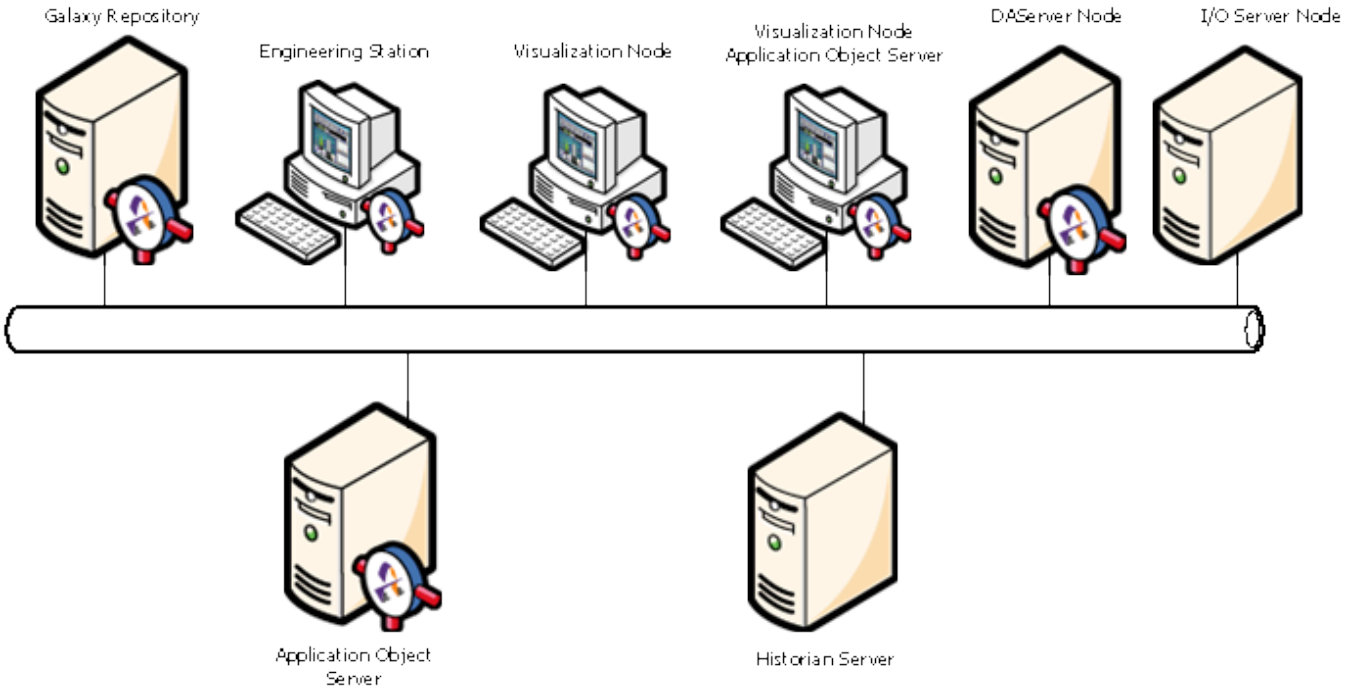
Note: Even if SubstituteString is not empty, if MatchString is empty, the Item Name is restored to the original Item name. Inserting text at the beginning of the name is not allowed. For example, running the script IOSetRemoteReferences("Access1", "", "","Valve",0); does not append the string Valve at the beginning or end of the all original Item Names.

If SubstituteString is empty without an assigned value, IOSetRemoteReferences() restores the active Item Name to the original Item Name. Using a non-empty MatchString with an empty SubstituteString enables you to select a subset of remote references on the indicated original base access and restore them to their original Item names.

Access Application Server data from InTouch

ArchestrA provides a set of common services and underlying architecture for a suite of products. You can select from this array of products to build plant automation and information systems using modular ArchestrA components.

Application Server provides a set of services to build plant automation applications. Application Server services are distributed across a set of nodes within the system.



Typically, the InTouch HMI works with Application Server to provide the visual interface for the application that operators interact with to manage a plant process.

Use Application Server object attributes with InTouch tags

You can use InTouch tags to interact with Application Server object attributes to transfer data between an InTouch application and an Application Server data repository.

The InTouch HMI communications protocol support includes Message Exchange. When WindowViewer runs an InTouch application, Message Exchange regards WindowViewer as an anonymous engine.

This anonymity means the InTouch application has no attributes that can be accessed by other Message Exchange clients. WindowViewer is not configured, managed, or viewed as an AutomationObject within an Application Server Galaxy. The InTouch HMI only uses Message Exchange to subscribe to those active items available from Application Server.

You can use the InTouch Tag Browser to select a Galaxy as a tag source for remote tags and browse through the namespace of the Galaxy. An Application Server object attribute or property of an attribute can be used in a remote reference or used as an item for an InTouch I/O tag.

For more information about using Application Server objects as a remote tag source, see Configure the InTouch HMI to use a Galaxy as a remote tag source.

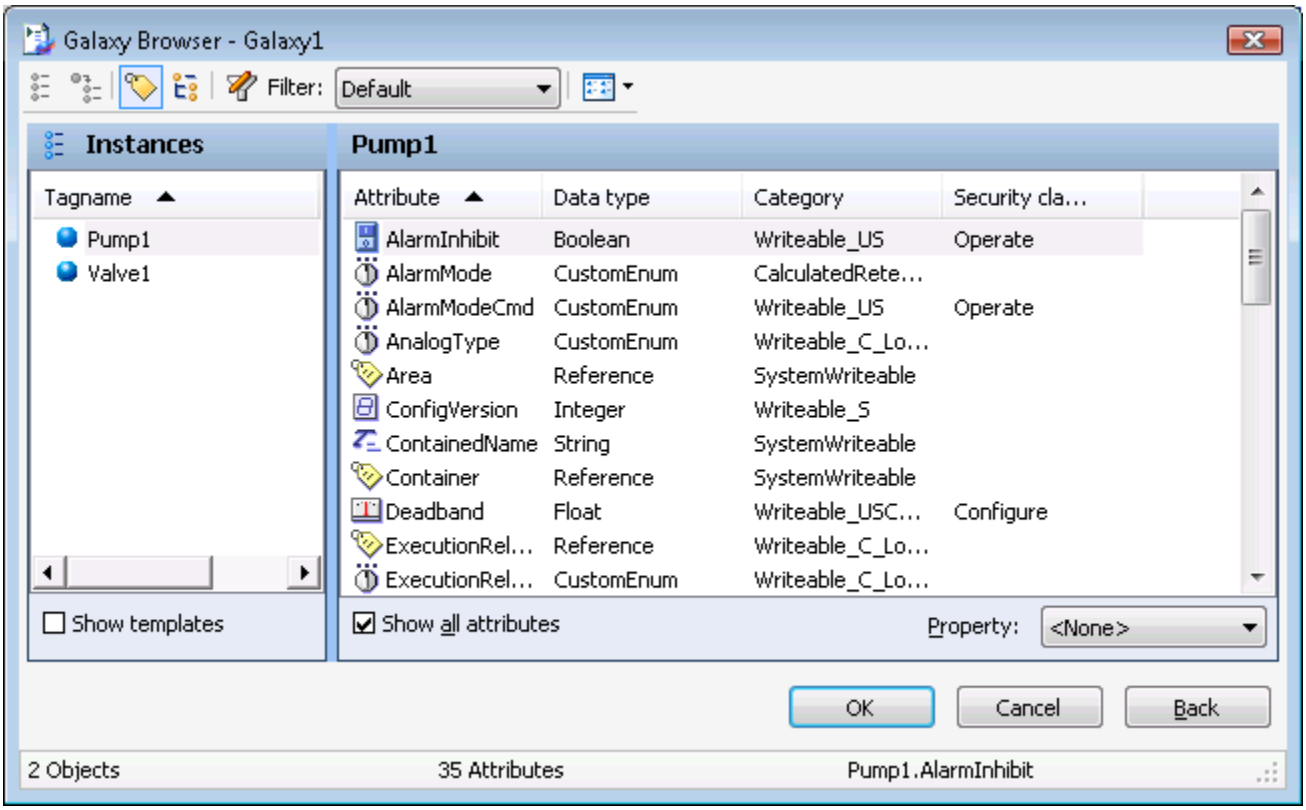
A pre-configured Access Name called Galaxy is available in WindowMaker for Message Exchange access. The Galaxy Access Name is only relevant for the InTouch HMI in the ArchestrA environment. There are no user configurable properties for the Galaxy Access Name.

Browse Application Server object attributes from InTouch

To browse and select Application Server attributes from the InTouch HMI, you must first set up a tag source for the Galaxy in the InTouch HMI. For more information, see Configure the InTouch HMI to use a Galaxy as a remote tag source.

Select the tag source from within the InTouch **Select Tag** dialog box.

The InTouch Galaxy Browser dialog box lists all objects within the target Galaxy. You can expand an object to see its contained objects or run-time accessible attributes. The Galaxy Browser dialog box does not show those attributes that start with an "_" (marked as hidden) or any attribute of type QualifiedStruct.



To return from the Galaxy Browser dialog box to the regular InTouch Select Tag dialog box, select **Back**.

Application Server browser restrictions

The following restrictions apply when viewing Application Server objects with the InTouch Attribute Browser dialog box:

- Only run-time visible attributes in a single Galaxy's namespace are viewable. This includes the capability to switch between an object's TagName and its HierarchicalName.

  An object attribute can be selected from the **Attribute Browser** dialog box if it meets the following requirements:

  - Visible during run time
  - From a currently checked in AutomationObject
  - Attribute name does not have an "_" following a "."
- The Attribute Browser dialog box only shows Application Server object attribute data types supported by the InTouch HMI. For more information about supported data types, see Map Application Server data types to InTouch data types.
- The InTouch Attribute Browser dialog box does not show any attributes that would result in an InTouch Item Name greater than the 95 character maximum.

- Automation object array elements can be displayed or retrieved in the InTouch HMI by using "TagName.AttributeName[index]" as a reference. Use an index of -1 to show or retrieve all array element values.
- You can select a property of an object attribute with the Tag Browser. By default, the Value property is selected when the attribute is selected.

Special extensions in Application Server objects

The WindowMaker Tag Browser and the Message Exchange client in WindowViewer add and recognize special extensions to each Application Server object attribute. These extensions provide access to information that otherwise would not be available to the InTouch HMI.

These special extensions are optional and do not need to be used by the InTouch application. However, applications that handle status and quality information frequently need to use these extensions.

These items extend the namespace of attributes to include additional properties that WindowViewer can expose to application scripts and Windows. For example, the reference to "TIC101.PV.#ReadSts" provides access to the MxStatus information for the subscription to TIC101.PV. This information is very useful for displaying extended information that is exposed by Message Exchange.

These properties do not exist as object attributes in Application Server as named elements. The properties are client-side extensions, supplied in the client abstraction layer, that make object attributes visible to the InTouch HMI. The following table describes the attribute extensions for the InTouch HMI:

Attribute Extension	Data Type	Purpose
None	Coerced	The default extension. Means that no extension is provided. The item is read/written with the value data type coerced as appropriate to InTouch. Failed read/write information can be obtained if the client subscribes to the #ReadSts or #WriteSts items described below. Example: "Pump1.PV".
.#VString for floats / doubles attributes only: .#VString1 .#VString2 .#VString3 .#VString4	String (read/write)	Sets up a subscription to the reference that has ".#VString" as a suffix. This is the underlying reference. Returns the current value of the underlying reference as a string when reads and writes are both working correctly. If the UserGetAttribute returns bad status, this item returns an abbreviated status description string based on MxStatus instead of the value. The abbreviated status description strings are: "?Pending" - pending "?Warning" - warning "?Comms" - communication error "?Config" - configuration error "?Oper" - operational error "?Security" - security error "?Software" - software error "?Other" - other error

For .#VString, if the status is good but the quality is bad, this item returns the quality description string, available from Message Exchange, instead of the value.

The value is returned as a string only when quality and status for UserGetAttribute are both good or when the quality is good and the status is uncertain. This may require coercion if the data type returned by Message Exchange is not a string. When quality or status is uncertain, the value shows a "?" as a suffix. For example, "3.27?" or "True?".

Map Application Server data types to InTouch data types

Application Server includes some attributes and data types that do not map directly to the four primary data types supported by InTouch tags.

The following table shows how the client abstraction layer maps data types for read and write operations. It also shows the data types that the Galaxy dictionary exposes to InTouch.

Attribute **Property Data Type**	**InTouch Data Type**	**Notes**
Float	Real - 32 bit	Pass through.
Double	Real - 32 bit	If double is IEEE NAN, then convert to float IEEE NAN. If this overflows, set **Quality** to Bad and pass float IEEE NAN. If the double fractional value is a smaller fraction that the smallest float fraction of 1.17549E-38, treat it as 0.0 float and set **Quality** to Good.
Boolean	Discrete	False = 0, True = 1.
Integer	Integer - 32 bit	Pass through.
String (always Unicode)	Message - MBCS (multi-byte character set encoded)	Truncate the string if it is too long for InTouch and set the quality to uncertain. Retain both bytes of each Unicode character.
Time	Message - MBCS	Format as an appropriate string for the locale. Use **MxValue** to convert the string.
ElapsedTime	Real	Pass as float seconds. **MxValue** supports coercion to this type.
MxDataType	Message - MBCS	Pass the string.
MxSecurityClassification	Message - MBCS	Pass the string.
MxQuality	Message - MBCS	Pass the string.
MxReference	Message - MBCS	Pass the reference string only as Unicode.
MxCategorizedStatus	Message - MBCS	Pass the string.
MxQualifiedStruct	Not supported	Not supported.
MxQualifiedEnum	Message - MBCS	Pass the Enum string. The integer ordinal value can be accessed by applications by referencing #EnumOrdinal. For example, "Pump1.PV.#EnumOrdinal".

Attribute Property Data Type	InTouch Data Type	Notes
Array of Strings	Message - MBCS (Read-only)	Put each element of the array into a comma-separated string such as: "String1,String2,String3" up to the maximum limit of an InTouch string value. If this is truncated, the associated quality sent to the InTouch HMI is uncertain. You cannot write to an entire array of strings, but you can write to individual elements of an array.
All arrays	Integer, Real, Message, or Discrete	Only supports a subscription to a single element of an array. In this case, the conversions described above are applicable. Otherwise, the return is an empty string with Bad quality.
MxInternationalizedText	Message	This is accessed as a string type at run time.

Read/Write behavior of Application Server attributes

When the system writes to an Automation object attribute, its write status is initially set to "?Pending".

When the write is completed, the #WriteSts string is updated with the result of the write. If the write completes successfully, #WriteSts value is set to an empty string. If the write returns an error and is pending, the #WriteSts item continues to show the most recent write status even if subscription updates continue to occur on reads.

You can also use the #VString1 to #VString4 items to convert float values or double values to a string format, The number N indicates the number of decimal places to be returned. For example, "3.1234" is the string for #VString4. You can use the #VString item without a number to round the float or double value to an integer and to return it as a string value.

Attribute Extension	Data Type	Purpose
.#EnumOrdinal	Integer (read/write)	Contains the currently read ordinal value for attributes of the Qualified Enum type. This is a way to return an integer for enumerations rather than returning a string.
.#ReadSts	String (Read-only)	Contains the current read status of the subscribed to item to which the string is concatenated. It appears as "TIC101.PV.#ReadSts". This is provided by Message Exchange as a string. Its' value can be one of the following: "?Config"- configuration error "?Comms" - communication error "?Oper" - operational error "?Pending" - pending "?Warning" - warning "?Security" - security error "?Software" - software error "?Other" - other error **Note:** If the associated item (for example, TIC101.PV) is not subscribed to, the returned string is blank.
.#WriteSts	String (Read-Only)	Contains the last write status of the item to which this string is concatenated, for example Pump1.Cmd.#WriteSts. This is provided by Message Exchange as a string. If the string is blank, the last write to the item is successful. Otherwise, #WriteSts can be one of the following values: "?Config"- configuration error "?Comms" - communication error "?Oper" - operational error "?Pending" - pending "?Warning" - warning "?Security" - security error

Attribute Extension	Data Type	Purpose
		"?Software" - software error "?Other" - other error **Note:** If the associated item (for example, TIC101.PV) is not subscribed to, the returned string is blank.

Configure the InTouch HMI to use a Galaxy as a remote tag source

You can use the InTouch Tag Browser to select an Application Server object as a tag source and browse the Galaxy database. Application Server object attributes or attribute properties can be used in remote references or as items for InTouch I/O tags.

When an InTouch application runs as a client providing the visual interface for an Application Server application, you must install Application Server Bootstrap and a WinPlatform object on the same node as the InTouch application.

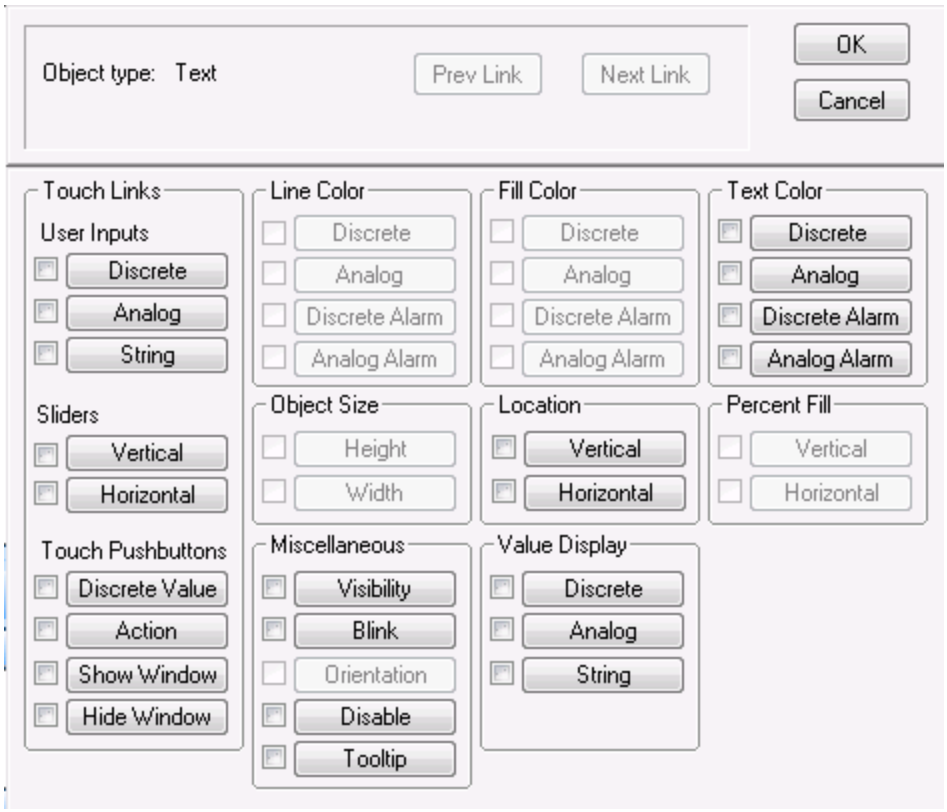
To browse the Galaxy namespace from the InTouch HMI, you also need to install the Integrated Development Environment (IDE).

The InTouch HMI uses the Message Exchange functionality of the Platform to view the Galaxy namespace and provide better data communication.

Configure InTouch to use a Galaxy as a remote tag source

1. Open an application window in WindowMaker.
2. Double-click on a text object.

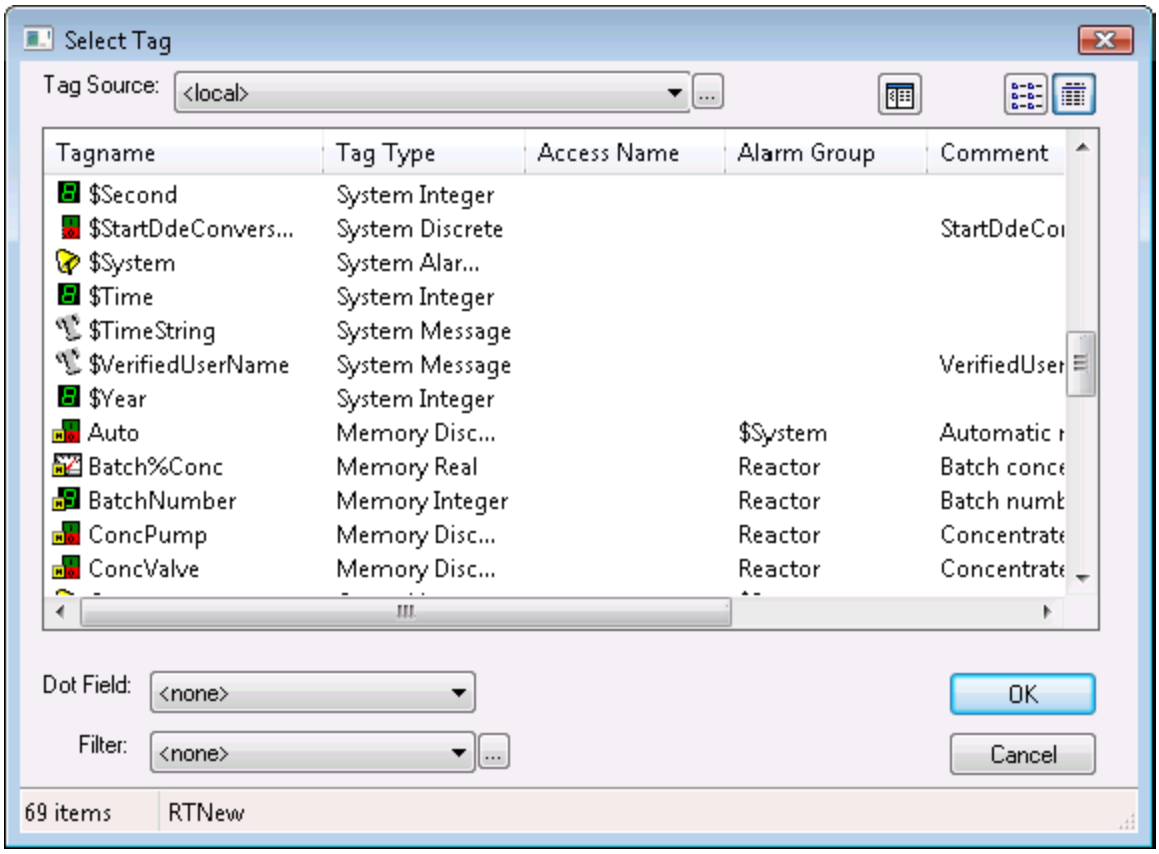
   The **Animation Links** dialog box appears.



3. In the **Value Display** area, select **Analog.**

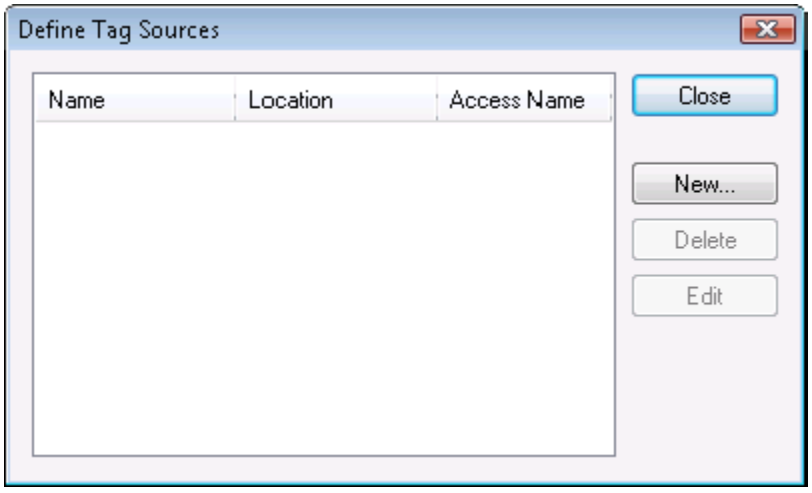
   A dialog box appears to insert an expression.
4. Delete any expression located within the **Expression** box.
5. Double-click in the **Expression** box.

   The **Select Tag** dialog box appears.

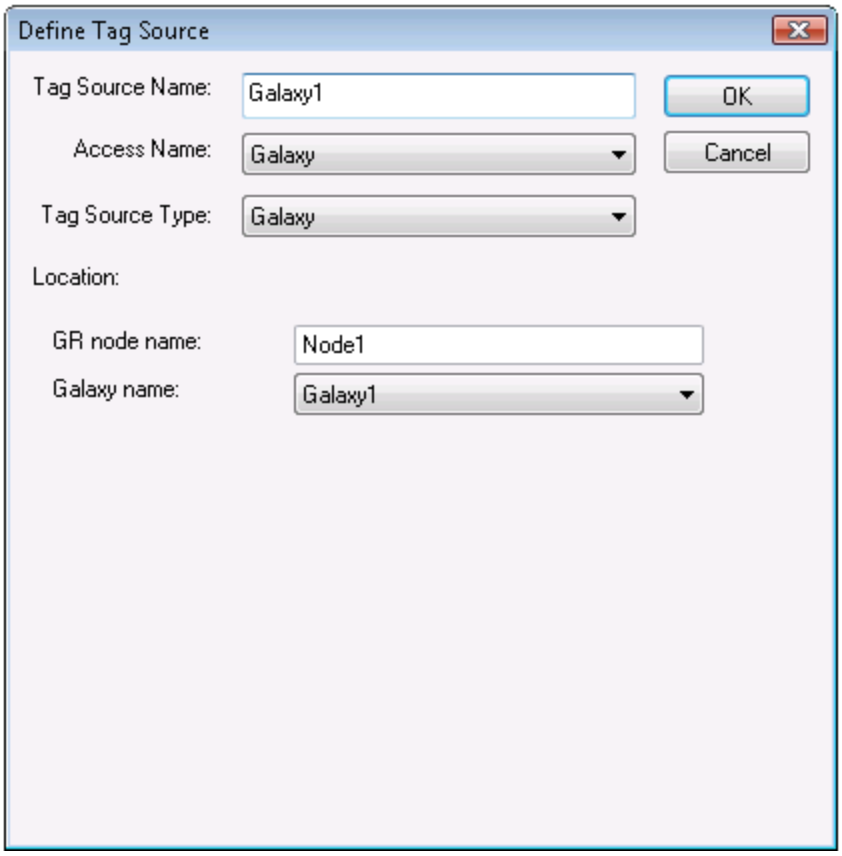


6. Select the button to the right of the **Tag Source** box.

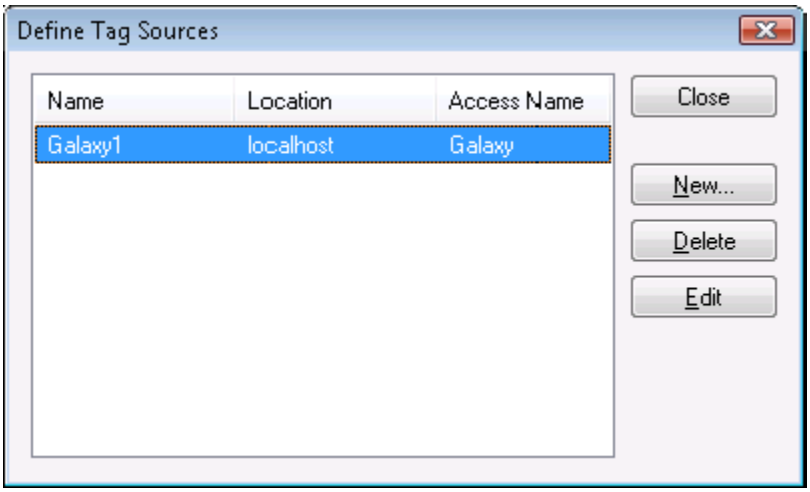
   The **Define Tag Sources** dialog box appears.



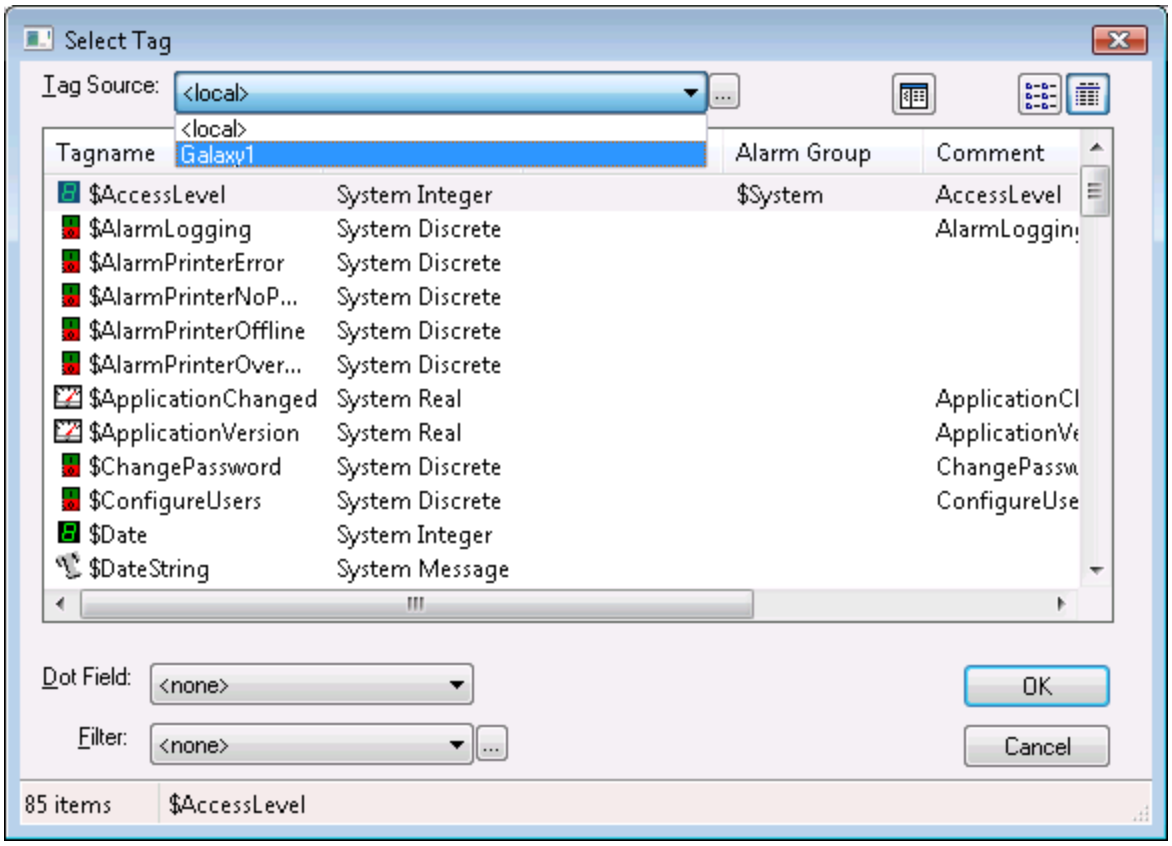
7. Select **New** to show the **Define Tag Source** dialog box.



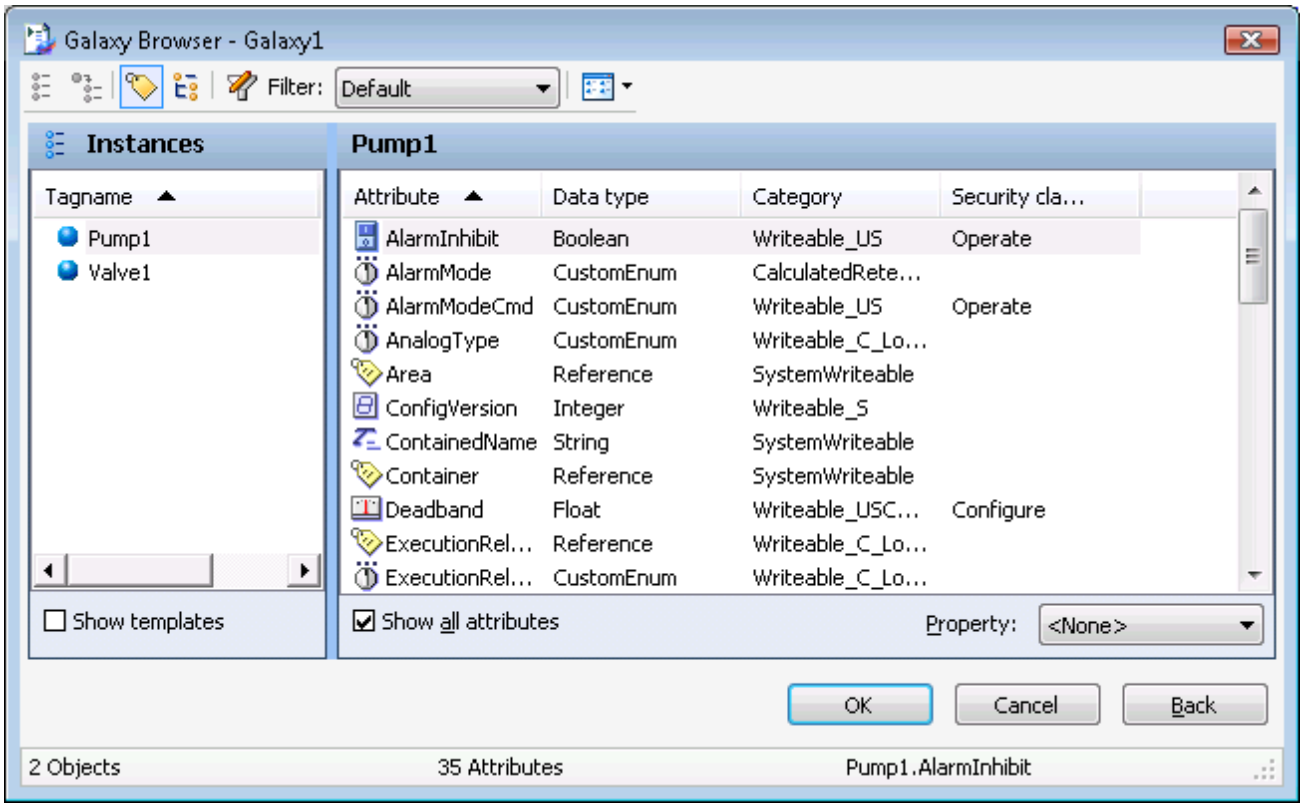
8. Enter values for the boxes of the **Define Tag Source** dialog box. Do the following:
   a. In the **Tag Source Name** box, type the name of your remote Galaxy tag source.
   b. In the **Access Name** box, select Galaxy from the list.
   c. In the **Tag Source Type** box, select Galaxy from the list.
   d. In the **Location** area, type the Galaxy Repository Node name and select the Galaxy from the list.
   e. Select **OK**. The **Define Tag Sources** dialog box shows the remote tag source you defined in its list.



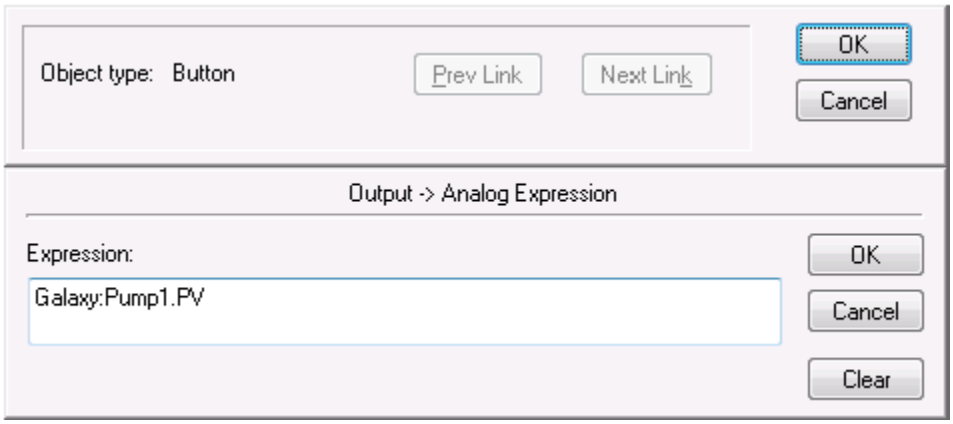
9. Select **Close** to close the Define Tag Sources dialog box. The **Select Tag** dialog box shows the new tag source from the list of the **Tag Source** box.



10. Select the new tag source you created from the **Tag Source** box. The **Galaxy Browser** dialog box opens with a list of tags in the left pane.



11. Select a tag from the left pane of the **Galaxy Browser** dialog box. The right pane of the **Galaxy Browser** dialog box refreshes with the attributes of the selected tag.
12. Highlight the attribute you want to use and select **OK**. The **Output -> Analog Expression** dialog box appears with an expression in the **Expression** box.



13. Confirm the string expression is correct.

    The expression uses the form:

    Galaxy:tag_name.attribute_name

    Example:

    ```
    Galaxy:Pump1.PV
    ```

14. Select **OK** to close the **Output > Analog Expression** dialog box.
15. Configure remaining object links as needed.
16. Select **OK** in the animation link dialog box.
17. Select **Runtime**. The text object shows the value for the configured tag attribute.

View timestamp and quality information for an I/O tag

The InTouch HMI places value, time, and quality (VTQ) indicators on all data values delivered to VTQ-aware clients. InTouch uses a set of dotfields as indicators of data quality that are useful for troubleshooting purposes.

- The .Value dotfield contains the value of the specified tag. This is also the default dotfield of every InTouch tag. If no other dotfield is specified, the .Value dotfield is assumed.
- The set of **Time** dotfields are time stamps indicating the last time a tag was updated.
- The **Quality** dotfields show the reliability of data values assigned to an I/O tag.

View timestamp information for an I/O tag

The set of **Time** dotfields are assigned to tags in the following format:

Tag_name.Time_Dotfield

.TimeDate Dotfield

The .**TimeDate** dotfield shows the whole number of days that have passed between January 1, 1970 and the last

update of a tag value from an I/O Server.

Category

Tag

Usage

```
Tag_name.TimeDate
```

Parameter

Tag_name
Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

See Also

.TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeDateString Dotfield

The **.TimeDateString** dotfield shows the date and time when a tag value is updated from an I/O Server.

Category

Tag

Usage

```
Tag.TimeDateString
```

Parameter

Tag
Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Message (read-only).

See Also

.TimeDate, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeDateTime Dotfield

The .**TimeDateTime** dotfield shows the fractional number of days that have passed between January 1, 1970 and the last update of a tag value from an I/O Server.

Category

Tag

Usage

```
Tag.TimeDateTime
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Real (read-only).

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeDay Dotfield

The .**TimeDay** dotfield shows the number of days within the month that have passed since the last update of a tag value from an I/O Server.

Category

Tag

Usage

```
Tag.TimeDay
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

Valid Values

Values can range from 1-31.

See Also

.TimeDate, .TimeDateString, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeHour Dotfield

The .**TimeHour** dotfield shows the number of hours within a day that have passed since the last update of a tag value from an I/O Server.

Category

Tag

Usage

```
Tag.TimeHour
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

Valid Values

Values can range from 0-23.

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeMinute Dotfield

The .**TimeMinute** dotfield shows the minute portion of the time when the tag value was last updated from an I/O Server.

Category

Tag

Usage

```
Tag.TimeMinute
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

Valid Values

Values can range from 0-59.

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeMonth Dotfield

The .**TimeMonth** dotfield shows the month number (1-12) of the date when a tag value is updated from an I/O Server.

Category

Tag

Usage

```
Tag.TimeMonth
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

Valid Values

Values can range from 1-12.

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeMsec Dotfield

The .**TimeMsec** dotfield shows the millisecond portion of the time when the tag value was last updated from an I/O Server.

Category

Tag

Usage

```
Tag.TimeMsec
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

Valid Values

Values can range from 0 - 999.

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString, .TimeYear

.TimeSecond Dotfield

The .**TimeSecond** dotfield shows the second portion of the time when the tag value was last updated from an I/O Server.

Category

Tag

Usage

```
Tag.TimeSecond
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

Valid Values

Values can range from 0 - 59.

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeTime, .TimeTimeString, .TimeYear

.TimeTime Dotfield

The .**TimeTime** dotfield shows the timestamp when a tag value is updated from an I/O Server expressed as the number of milliseconds that have elapsed since midnight.

Category

Tag

Usage

```
Tag.TimeTime
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

Valid Values

Values can range from 0 - 86399999.

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTimeString, .TimeYear

.TimeTimeString Dotfield

The **.TimeTimeString** dotfield shows the time when a tag value is updated from an I/O Server.

Category

Tag

Usage

```
Tag.TimeTimeString
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Message (read-only).

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeYear

.TimeYear Dotfield

The .**TimeYear** dotfield shows the four-digit year when a tag value is updated from an I/O Server.

Category

Tag

Usage

```
Tag.TimeYear
```

Parameter

Tag

Any discrete, integer, real, message, indirect analog, indirect discrete, or indirect message tag.

Data Type

Integer (read-only).

Valid Values

Any year expressed as a four-digit number.

See Also

.TimeDate, .TimeDateString, .TimeDay, .TimeDateTime, .TimeHour, .TimeMinute, .TimeMsec, .TimeMonth, .TimeSecond, .TimeTime, .TimeTimeString.

View quality information for an I/O tag

You can use a set of quality dotfields to ensure the integrity of data sent between an I/O Server and your InTouch applications. Quality dotfields represent the quality state of an item's data value. This quality attribute makes it fairly easy to monitor the integrity of InTouch data sent between network nodes.

The data quality standard is based on the OLE for Process Control (OPC) proposed standard, which in turn is

based on Fieldbus Data Quality Specifications.

You can configure how you want numeric values formatted at run time based on data type and data quality.

Quality data format

An I/O Server can report six mutually exclusive states of data quality sent to clients by assigning values to a set of InTouch .Quality dotfields. The low eight bits (Least Significant Byte) of the Quality dotfields are currently defined in the form of three bit fields; Quality (Q), Substatus (S), and Limit (L) with the following format: **QQSSSSLL.** When the client application cannot communicate with the server, the .**Quality** dotfield's value is 0.

The data quality states reported by an I/O Server with .Quality dotfields are shown in the following table:

Quality States	Decimal Value	Hex Value	MS Byte XXXXXXXX	LS Byte QQSSSSLL	Quality	Quality Sub Status	Limit
Good	192	0x00C0	00000000	11000000	Q=3	S=0	L=0
Clamped High (Out of Range)	86	0x0056	00000000	01010110	Q=1	S=5	L=2
Clamped Low (Out of Range)	85	0x0055	00000000	01010101	Q=1	S=5	L=1
Cannot Convert	64	0x0040	00000000	01000000	Q=1	S=0	L=0
Communications Failed	24	0x0018	00000000	00011000	Q=0	S=6	L=0
Cannot Access Point	4	0x0004	00000000	00000100	Q=0	S=1	L=0

About data Quality dotfields

The .Quality dotfields indicate the quality of data values the last time data was received. The SuiteLink and DDE protocols only send clients (for example, WindowViewer) updated quality when a data change is provided by the I/O Servers. Therefore, you only observe a quality change when a new data value is received. Some I/O Servers can send current data values with updated quality when the quality associated with the data changes.

It might not be possible to directly reference the quality of a server item's value using the SuiteLink and DDE protocols. To do this, the I/O Server must directly support Item.Quality. Without this support, the item fails to go on advisement and the .Quality dotfield value never changes from 0.

The TestProt I/O Server simulator does not directly support Item.Quality. The simulator does not send out new data values when you modify the quality using the Quality menu command.

If you want to observe data quality for an I/O item and the I/O Server does not directly support addressing of

Item.Quality, then define an InTouch I/O tag to look at the server item and then monitor the .Quality of the InTouch tag. If you exceed your tag limit, then consider using the IOSetRemoteReferences() function in a script to dynamically adjust the I/O points.

The SuiteLink and DDE protocols do not interpret connection state or other changes in I/O Server status as quality items sent to the client. As a result, an item's quality does not necessarily indicate the current data server status nor the current status of the client-to-server connection. An I/O Server process can stop and the value of the .Quality field does not change. If the communications link is lost, the value of the .Quality field does not necessarily change.

Use DDE or SuiteLink internal status items to monitor the I/O Server connection.

.Quality Dotfield

The .**Quality** dotfield shows a numerical assessment of the quality of data provided by an I/O Server.

Category

Tag

Usage

```
Tag.Quality
```

Parameter

Tag

Any discrete, integer, real, indirect analog, or message tag type.

Data Type

Integer (read-only).

Valid Values

Values can range from 0-255.

See Also

.QualityLimit, .QualityStatus, .QualitySubstatus

Example

```
IF I0Tag.Quality <> 192 THEN
   LogMessage("This data is not Good, assuming high-byte of 2-byte quality is zero");
   LogMessage("Better to check .QualityStatus to avoid this assumption");
ENDIF;
```

.QualityLimit Dotfield

The .**QualityLimit** dotfield shows the quality limit of a data value provided by a connected I/O Server.

Category

Tag

Usage

```
Tag.QualityLimit
```

Parameter

Tag
Any discrete, integer, real, indirect analog, or message tag type.

Data Type

Integer (read-only).

Valid Values

0 = Not Limited
1 = Low Limited
2 = High Limited
3 = Constant

See Also

.Quality

.QualityLimitString Dotfield

The .**QualityLimitString** dotfield shows the quality limit string of a data value provided by a connected I/O Server.

Category

Tag

Usage

```
Tag.QualityLimitString
```

Parameter

Tag

Any discrete, integer, real, indirect analog, or message tag type.

Data Type

Message (read-only).

See Also

.Quality, .QualityLimit

.QualityStatus Dotfield

The .**QualityStatus** dotfield shows an integer quality status of a data value provided by an I/O Server.

Category

Tag

Usage

```
Tag.QualityStatus
```

Parameter

Tag

Any discrete, integer, real, indirect analog, or message tag type.

Data Type

Integer (read-only).

Valid Values (SSSS)

0 = Bad

1 = Uncertain

3 = Good

Example

```
IF I0Tag.QualityStatus <> 3 THEN
   LogMessage("This data is not Good!");
```

```
ENDIF;
```

See Also

.Quality, .QualitySubStatus

.QualityStatusString Dotfield

The .**QualityStatusString** dotfield shows the quality status string of a data value provided by an I/O Server.

Category

Tag

Usage

```
Tag.QualityStatusString
```

Parameter

Tag
Any discrete, integer, real, indirect analog, or message tag type.

Data Type

Message (read-only).

See Also

.QualityStatus, .QualitySubStatus, .Quality

.QualitySubstatus Dotfield

The .**QualitySubstatus** dotfield shows the quality sub-status of a data value provided by an I/O Server.

Category

Tag

Usage

```
Tag.QualitySubstatus
```

Parameter

Tag

Any discrete, integer, real, indirect analog, or message tag type.

Data Type

Integer (read-only).

Valid Values (SSSS) and (QQ)

Substatus (SSSS) for BAD quality (QQ=0).

0 = Non-specific

1 = Configuration error

2 = Not connected

3 = Device failure

4 = Sensor failure

5 = Last known value

6 = Communication failure

7 =Out of service

Substatus (SSSS) for UNCERTAIN quality (QQ=1).

0 = Non-specific

1 = Last usable value

4 = Sensor not accurate

5 = Engineering Units exceeded

6 = Sub-Normal

Substatus (SSSS) for GOOD quality (QQ=3).

0 = Non-specific

6 = Local override

See Also

.QualityStatus, .QualitySubStatus, .Quality

.QualitySubstatusString Dotfield

The .**QualitySubstatusString** dotfield shows the quality sub-status string of a data value provided by an I/O Server.

Category

Tag

Usage

```
Tag.QualitySubstatusString
```

Parameter

Tag

Any discrete, integer, real, indirect analog, or message tag type.

Data Type

Message (read-only).

See Also

.QualityStatus, .QualitySubstatus, .Quality

Initialize and reset I/O connections at run time

WindowViewer initiates all I/O conversations when it starts the InTouch application. You can also manually restart I/O conversations while your InTouch application is running. You can initialize and reset I/O connections during run time with WindowViewer commands or scripts.

You can also specify that I/O connections should be reinitialized based on their default values. If you select this option, the default settings are used and the current settings are ignored when Access Names are reinitialized. To reinitialize I/O conversations by Access Name, you must have an InTouch application with Access Names already defined.

Reinitialize I/O connections with commands

The **Special** menu in WindowViewer includes a set of commands to reinitialize all I/O conversations or select a specific I/O conversation.

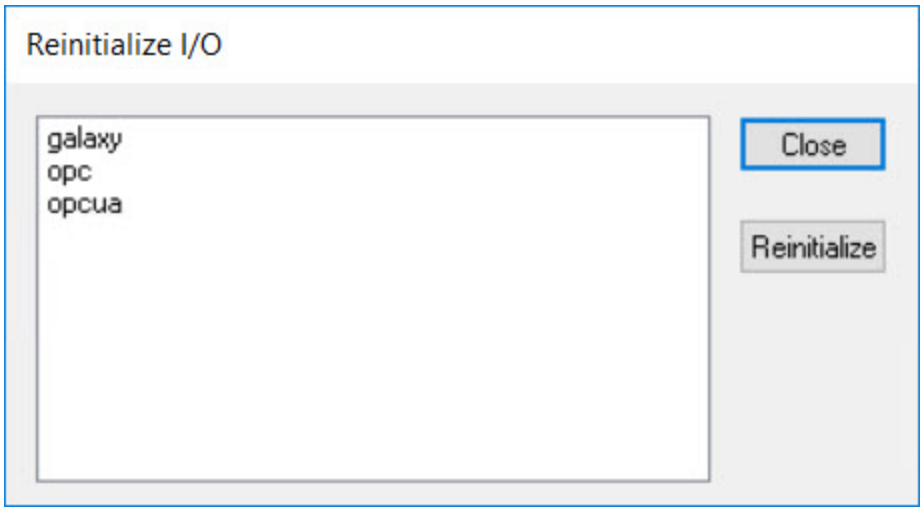
You can reinitialize Access Names using the default settings of InTouch. Using default reinitialization, the Access Name ignores the current values assigned to node name, application name, and topic. The Access Name is reinitialized with the original Access Name settings.

Reinitialize all Access Names at run time

1. On the **Special** menu, select **Reinitialize I/O**.
2. Select **Reinitialize All**. All Access Names are reinitialized.

Reinitialize selected Access Names at run time

1. On the **Special** menu, select **Reinitialize I/O**, then choose **Select**. The **Reinitialize I/O** dialog box shows a list of Access Names.



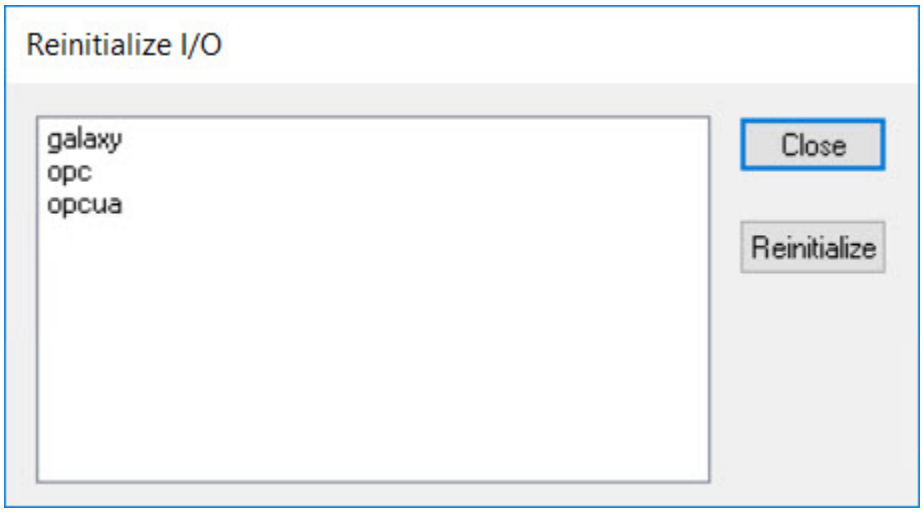
2. Highlight one or more Access Names to reinitialize, then select **Reinitialize**. The selected Access Names are reinitialized.

Reinitialize Access Names using default settings

1. Open an application within WindowMaker.

1. On the **File** menu, point to **Configure**, and then select **WindowViewer**. The **WindowViewer configuration** screen appears.
2. On the **Preferences** tab, select the **Reinitialize Default** checkbox in the **I/O** area.

3. Select **OK**.
4. Open the application in WindowViewer.
5. On the **Special** menu, highlight **Reinitialize I/O**, then choose **Select**. The **Reinitialize I/O** dialog box appears.



6. Highlight one or more Access Names to reinitialize, then select Reinitialize. The current settings are ignored for node name, application name, and topic. The Access Names are reinitialized with the original Access Name settings.

Reinitialize I/O connections with scripts

You can reinitialize an I/O connection to one or more Access Names by creating a script that includes the following functions:

- **IOReinitAccessName()**
- **IOReinitialize()**
- **IOStartUninitConversations()**

IOReinitAccessName() Function

The IOReinitAccessName() function reinitializes the I/O connection to a specified Access Name.

Category

I/O communication

Syntax

```
IOReinitAccessName("AccessName", Default);
```

Arguments

AccessName

Access Name to be reinitialized.

Default

Default = 1. The I/O reinitialization uses the original default Access Name values assigned from WindowMaker.
Default = 0. The I/O reinitialization uses the current node, application, and topic values assigned to the Access Name.

Remarks

The default settings are determined by the settings in the Access Name configuration panel and also in the WindowViewer configuration (Retry Initiates, Start Local Servers, Reinitialize Default).

Examples

This example reinitializes the I/O connection to AccessName1 using the default values assigned to the node, application, and topic.

```
IOReinitAccessName("AccessName1", 1);
```

This example reinitializes the I/O connection to AccessName2 using the current values assigned to the node,

application, and topic.

```
IOReinitAccessName("AccessName2", 0);
```

IOReinitialize() Function

The **IOReinitialize()** function first closes and then restarts all active I/O connections defined for an InTouch application.

Category

Miscellaneous

Syntax

```
IOReinitialize();
```

Arguments

None.

Remarks

The **IOReinitialize()** function performs the same operation as the **Reinitialize I/O** command on the WindowViewer **Special** menu.

If WindowViewer is running as a service, the **IOReinitialize()** function and **Reinitialize ALL** command on the WindowViewer **Special** menu do not reinitialize all Access Names. If you select the Access Names listed in the **Reinitialize I/O** dialog box and select **Reintialize**, the selected Access Names are reinitialized.

For more information about navigating to the **Reinitialize I/O** dialog box, see Reinitialize I/O connections with commands.

Example

This example closes any active I/O connections and restarts all I/O connections defined for the InTouch application.

```
IOReinitialize();
```

IOStartUninitConversations() Function

When WindowViewer begins running an InTouch application, it automatically processes an initiate request to start all I/O conversations. If an I/O Server program does not respond to WindowViewer's initiate request, you can use the **IOStartUninitConversations()** script function to force WindowViewer to attempt to start the I/O conversation again.

Category

Miscellaneous

Syntax

```
IOStartUninitConversations();
```

Arguments

None.

Remarks

The **IOStartUninitConversations()** function performs the same operation as the **Start Uninitiated Conversations** command on the WindowViewer **Special** menu.

Example

This example forces WindowViewer to submit another initiate request to start all I/O connections defined for the InTouch application.

```
IOStartUninitConversations();
```

Use failover functionality with Access Names

You can specify that the InTouch HMI automatically switches to a secondary I/O Server if the primary I/O Server experiences communication problems. This is called I/O failover.

Configure failover

You can specify that your InTouch application switches to a failover secondary I/O Server when it can no longer communicate with the primary I/O Server.

When you set the failover, you specify the failover deadband. The failover deadband is the delay in seconds before switching from the primary Access Name to the secondary Access Name. The InTouch HMI triggers the failover when the expression or an I/O communication failure is true for the length of the deadband period. When the failover deadband is set to 0 or blank, the failover is triggered as soon as an I/O communication failure is detected.

Configure failover for an Access Name

1. If needed, stop WindowViewer.
2. On the **Home** menu, in the **Tags** group, select **Access Names**.

   The **Access names** dialog box appears with a list of all defined Access Names.
3. Select the Access Name from the list to add a failover server.
4. Select **Edit**.

The **Add access name** appears.

5. In the **Secondary source** section, select the **Enable Secondary Source** checkbox.
6. Do the following:
   - In the **Node Name** box, type the node name of the secondary I/O Server.
   - In the **Application Name** box, type the program name of the secondary I/O Server program from which data will be acquired.
   - In the **Topic Name** box, type the topic name you want to access from the secondary I/O source.
   - In the **Which protocol to use** area, select either **DDE** or SuiteLink as the secondary I/O Server communication protocol.
   - In the **When to advise server** area, select **Advise all items** or **Advise only active items** for the secondary I/O source.
7. In the **Failover** section, select the **Enable failover** checkbox.
8. Do the following:
   - Enter an optional failover expression or double-click in the **Failover expression** box to select a tag. For more information about failover expressions, see Force failover to a backup Access Name.
   - In the **Failover Deadband** box, type the length of the failover deadband in seconds.
   - Select **Switch back to primary when Failover conditions clear** if you want to enable switching from the secondary Access Name back to the primary Access Name after the failover condition clears.

     The default is to not switch back to the primary Access Name. If you select **Switch back to primary when failover conditions clear**, then the **Fail-back Deadband** option becomes selectable from the **Failover Configuration** dialog box.
   - From **Failback Deadband**, type the length of the failback deadband in seconds.

     The InTouch HMI triggers a failback to the primary Access Name after the expression and any associated I/O communication failure clear for the deadband period. When the expression is left blank or 0, the fail-back occurs as soon as the I/O communication failure condition clears.
9. Select **Update**.

Edit the Access Name parameters of a failover pair

To edit Access Name parameters that are part of a failover pair, you must have an Access Name configured for failover with a secondary I/O source.

Edit the Access Name parameters of a failover pair

1. If needed, stop WindowViewer.
2. On the **Home** menu, in the **Tags** group, select **Access Names**.

   The **Access Names** dialog box appears.
3. Highlight the Access Name pair and select **Edit**.

   The **Add Access Name** dialog box shows the parameters for the primary and secondary Access Names.
4. Change the Access Name parameters for the primary and secondary Access Names.
5. Select **Update**.

Remove failover for an Access Name

To remove failover for an Access Name, you must have an Access Name configured for failover with a secondary I/O source.

Remove failover for an Access Name pair

1. On the **Home** menu, in the **Tags** group, select **Access Names**.
2. Highlight the Access Name pair and select **Edit**.

   The **Add Access Name** dialog box appears.
3. Clear the **Enable Secondary Source** checkbox.
4. Select **OK**.

   Failover for the Access Name pair is disabled.

Force failover to a backup Access Name

You can manually switch between the primary and secondary sources of an Access Name without experiencing a failover. This is called a forced failover. To force a failover, you must have an Access Name configured for failover with a secondary I/O source.

You can use a failover expression or the **IOForceFailover()** script function to force a failover.

Failover expression

The **Failover Configuration** section shows the **Failover Expression** option to include a tag or expression that triggers a failover. The figure below shows the failover memory discrete tag entered as the value of **Failover Expression**.

Setting the failover expression to true, for example by setting the failover tag to true, switches the Access Name from the primary (false) to secondary (true) I/O data sources.

IOForceFailover() Function

The **IOForceFailover()** script function switches between the primary and secondary data sources of the Access Name. The active I/O node toggles between the primary and secondary nodes with each invocation of the script function.

Typically, the **IOForceFailover()** function is part of a script associated with button or another window object. Operators select the object from an application window to force a failover. After operators select the object again, the **IOForceFailover** function forces the I/O connection back to the formerly active I/O node.

Category

I/O Communication

Syntax

```
IOForceFailover("AccessName");
```

Argument

AccessName

Access name for which failover has been configured.

Example

The Acc1 Access Name has Primary and Secondary data sources and Primary is active. Acc1 fails over to the Secondary data source when the script runs.

```
IOForceFailOver("Acc1");
```

Temporarily disable failover functionality

You can manually disable failover switching between the primary and secondary I/O nodes of an Access Name. A typical case for temporarily disabling failover is during the brief period when the components of an InTouch system are started and not yet ready. After the components have stabilized, you can restore failover switching.

To disable failover for an Access Name, you must have an Access Name configured for failover with a secondary I/O source.

You can manually disable failover switching by two methods:

- Select the **Disable Failover** option from the **Failover** section.
- Run a script that includes the **IODisableFailover()** function.

Disable Failover Configuration Option

In the **Failover Configuration** section, clear the selection of the **Enable Failover** option to prevent the Access Name from switching between the primary and secondary I/O nodes.

You must edit the Access Name definition to set the **Disable Failover** option to active. As long as the option is active for the Access Name, failover is disabled.

IODisableFailover() Script Function

You can use the **IODisableFailover()** function in a script to disable failover for a specified Access Name. **IODisableFailover()** disables switching for all failover methods except by the **IOForceFailover()** script function method.

Category

I/O Communication

Syntax

```
IODisableFailover ("AccessName",Option);
```

Arguments

AccessName

Access name for which failover has been configured.

Option

1 = Disables failover
0 = Enables failover

Remarks

The Access Name can be specified as a literal string or it can be a string value provided by other InTouch tags or functions.

Examples

In this example, failover is disabled for the ModbusPLC1 Access Name.

```
IODisableFailover ("ModbusPLC1",1)
```

In this example, failover is enabled for the ModbusPLC1 Access Name.

```
IODisableFailover ("ModbusPLC1",0)
```

Retrieve information about failover pairs using scripting

You can write scripts that include functions that return the status of the primary, secondary, and active I/O sources of an Access Name. Typically, operators run a script to determine the status of an Access Name's secondary I/O source before forcing a failover.

To create scripts to return Access Name information, you must have an Access Name configured for failover with a secondary I/O source.

IOGetAccessNameStatus() Function

The **IOGetAccessNameStatus()** script function returns an integer indicating the connection status of the primary, secondary, or active I/O source of an Access Name.

Typically, the **IOGetAccessNameStatus()** return value is associated with an integer tag. The value of the tag can drive a discrete value display animation link that shows the status of the Access Names's active, primary, and secondary I/O sources to an operator.

Category

Miscellaneous

Syntax

```
Result=IOGetAccessNameStatus("AccessName", Mode);
```

Arguments

AccessName

The existing Access Name for which to return the status.

Mode

The value assigned to this argument determines what Access Name of the failover pair is queried about its current status.
0 - Status of the active Access Name I/O source
1 - Status of the Access Name primary I/O source
2 - Status of the Access Name secondary I/O source

Results

Returned Value	Description
-1	There is a configuration error in the Access Name. Either the Access Name does not exist or the Access Name does not have a defined secondary I/O source.
0	The connection to the requested I/O source is not successful.
1	The connection to the requested I/O source is successful.

Remarks

The **IOGetAccessNameStatus()** function is typically used in a script that determines the status of the secondary IO Source that is currently inactive. The operator runs the script to verify the status of the secondary connection before forcing a fail-over.

Example

This example returns the status of the secondary I/O source of the ModbusPLC1 Access Name. The returned value is associated with the **ANStatus** tag.

```
ANStatus = IOGetAccessNameStatus ("ModbusPLC1",2)
```

IOGetActiveSourceName() Function

The **IOGetActiveSourceName()** script function returns whether an access name currently uses its primary or secondary data source.

Typically, the **IOGetActiveSourceName()** function is included in a script associated with button or another window object. Operators then select the object from an application window to request the status of the application's I/O Servers.

Category

Miscellaneous

Syntax

```
Result=IOGetActiveSourceName("AccessName");
```

Argument

AccessName

The existing Access Name for which to return the source name.

Remarks

IOGetActiveSourceName() returns a string that indicates whether the primary or secondary nodes of an Access Name are being actively polled. Possible return values of the **IOGetActiveSourceName()** function are:

Primary	The primary node of the Access Name is actively polled.
Secondary	The secondary or failover node of the Access Name is actively polled.
Null	Both the primary and secondary nodes of an Access Name are inactive.

Example

In this example, the **ActiveServer** message tag is assigned the returned value (Primary, Secondary, or Null) that identifies the current active node of the ModbusPLC1 Access Name.

```
ActiveServer = IOGetActiveSourceName ("ModbusPLC1");
```

Monitor the status of an I/O connection

WindowViewer includes a built-in topic called IOStatus to monitor the status of a specific I/O conversation between an InTouch application and an I/O Server communicating with a PLC.

Note: In versions of InTouch before 7.0, the topic name was DDEStatus.

You can set up the **IOStatus** topic to monitor I/O conversations.

Use the IOStatus topic name

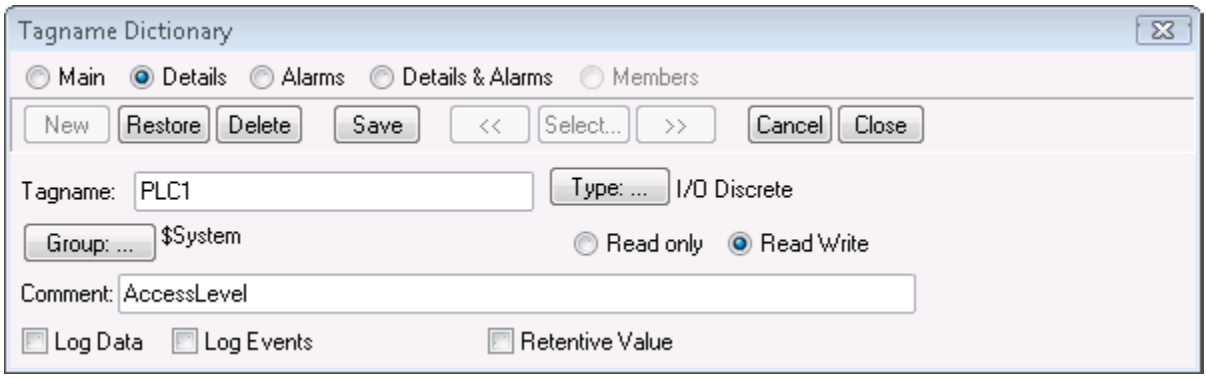
You can prepare the **IOStatus** topic to monitor I/O communication between WindowViewer and an I/O Server. In this example, WindowViewer communicates with the Simulation I/O Server to a PLC defined in the I/O Server with PLC1 as its topic name.

Note: The Simulation Server is a generic DAServer used as a training tool. The Simulation Server is located in the c:\program files\common files\ArchestrA folder.

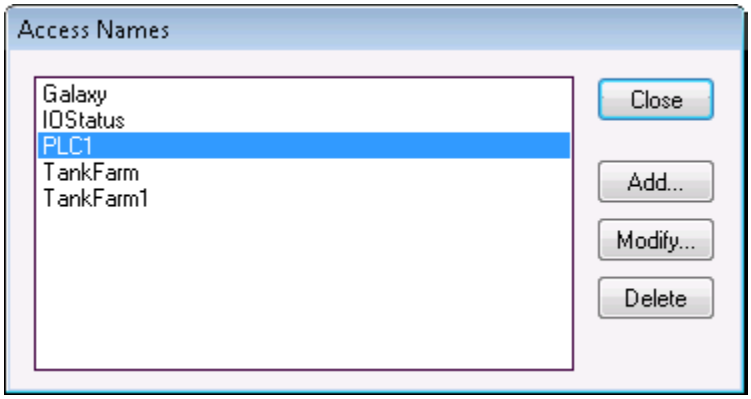
Monitor the status of I/O communications

1. Open an application in WindowMaker.
2. Open the Tagname Dictionary.
3. Create an I/O discrete tag.

When you are monitoring a I/O conversation using IOStatus, you must define at least one I/O type tag to the Access Name being monitored.

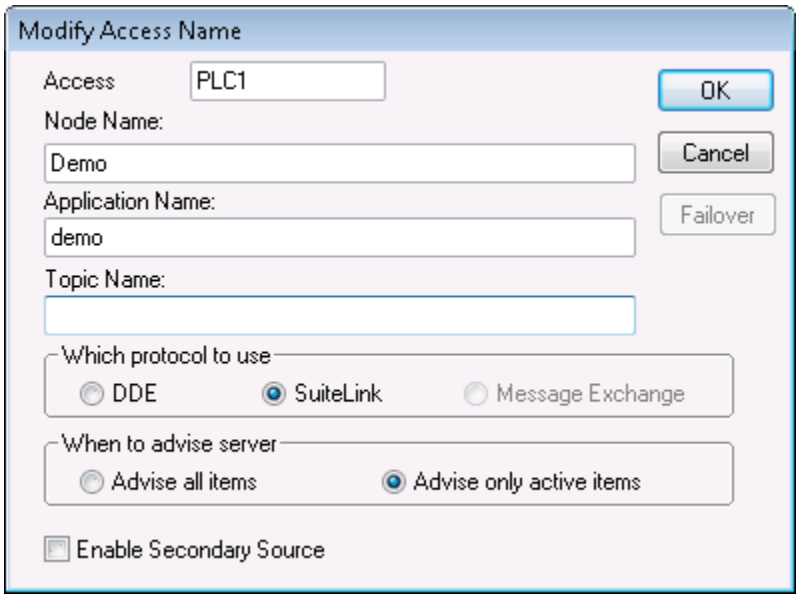


4. Select **Access Name** to assign the tag to an Access Name definition that defines IOStatus as its topic name.



Notice the **PLC1** Access Name definition currently exists.

5. Highlight PLC1 and select **Modify**.

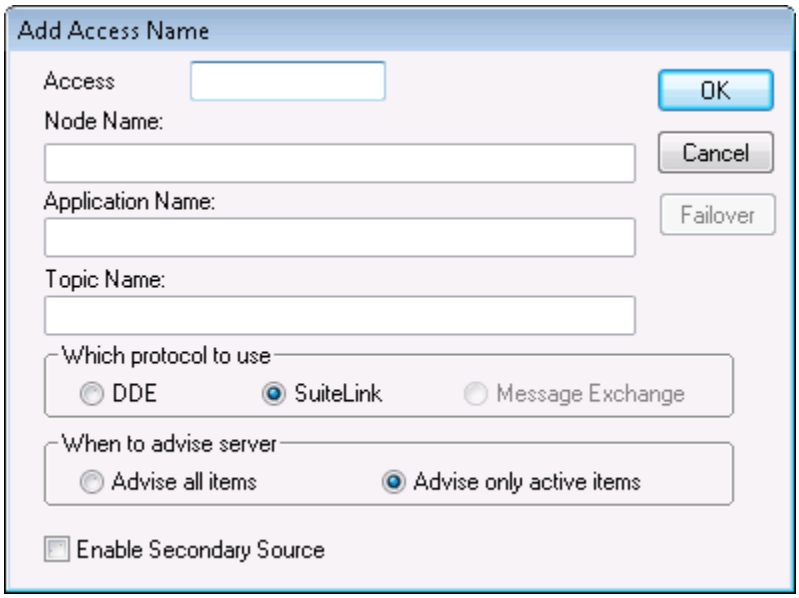


Finding the Access Name containing the right topic name in this example is easy because the tag and Topic Name are the same.

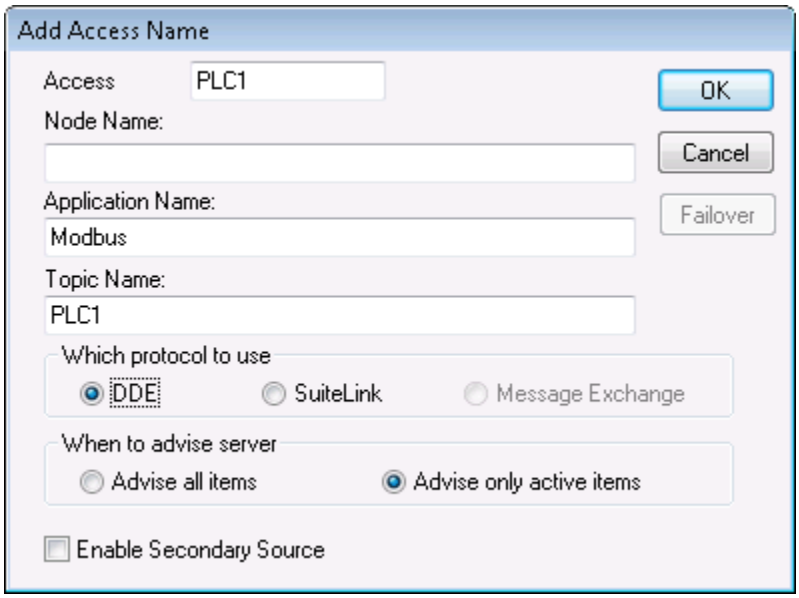
6. Select **Cancel** to close the dialog box and return to the initial **Access Name** dialog box.

7. Select **Add**.

   The **Add Access Name** dialog box appears.



8. Do the following:
   a. In the **Access box**, type IOStatus.
   b. In the **Application Name** box, type **View** because you are going to monitor the status from WindowViewer.
   c. In the **Topic Name** box, type **IOStatus** as the InTouch internal topic.
   d. Select **Advise only active items**.
9. Select **OK** to close the dialog box. The initial **Access Name** dialog box reappears showing your new Access Name, IOStatus, in the list:



10. Select **Close** to close the dialog box and associate the new Access Name with your I/O Discrete tag.

    In the Item box, type the Access Name for the actual topic name that you want to monitor.

11. Because your tag is the same as the Topic Name, you can select Use Tagname as Item Name and automatically enter it as the Item.

Note: When using the built-in topic IOStatus (DDEStatus before InTouch Version 7.0) to monitor an I/O conversation, the name you type in the Access Name box is always also used for the Item.

The Item Name can be a string up to 254 characters. The item name length is calculated as the sum of the Topic Name, Item Name and 1 delimiter character.

Use the IOStatus topic name in Excel

You can use Excel to monitor I/O status by entering the same information as a spreadsheet cell formula. For example, to monitor the same topic described in the previous procedure, enter the following formula in a cell:

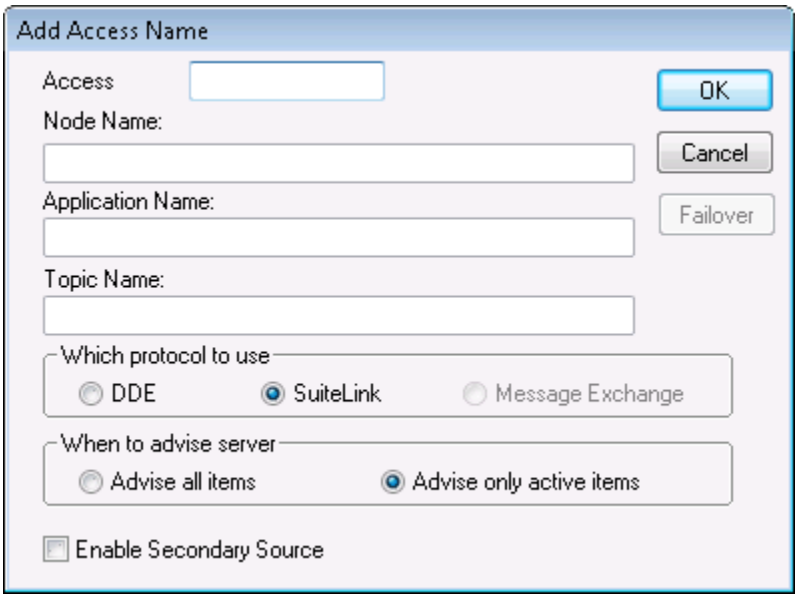
```
=view|IOStatus!'PLC1'
```

Monitor I/O server communications status

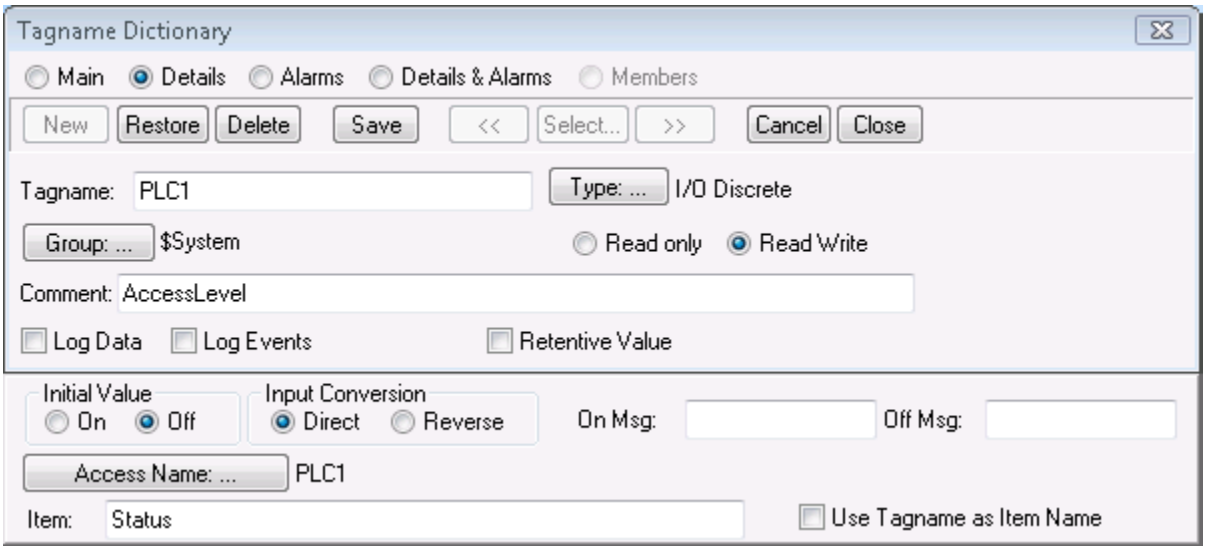
For each topic name being used, you can use a built-in discrete item, **Status**, to monitor communication status with the I/O Server program. The **Status** item is set to 0 when a communication failure occurs. The **Status** item is set to 1 when communicating normally with the I/O Server program.

Note: When you monitor the status of a topic using the IOStatus item, at least one I/O point must be active for the monitored topic.

From the InTouch HMI, you can read the state of the server communications by defining a tag and associating it with the topic configured for the device by using the word **Status** as the Item Name. For example, if WindowViewer is communicating with a PLC using the Modbus DAServer, the Access Name definition is similar to the following example:



To monitor the status of all communication to the topic PLC1, create the following tag definition:



In Excel, you can read the status of the PLC communications by entering the following formula in a cell:

```
=SIMULATE|PLC1!'STATUS'
```

Access InTouch tag data from other applications

When another application requests a data value from the InTouch HMI, it also must know three I/O address items. Follow these InTouch I/O address conventions.

VIEW (application name) identifies the InTouch run-time program that contains the data element.

TAGNAME (topic name) is the word always used when reading/writing to a tag.

ActualTagname (Item Name) is the actual tag defined for the item in the InTouch Tagname Dictionary.

For example, to access a data value in the InTouch HMI from Excel, a DDE Remote Reference formula is specified for the cell in which the data is written:

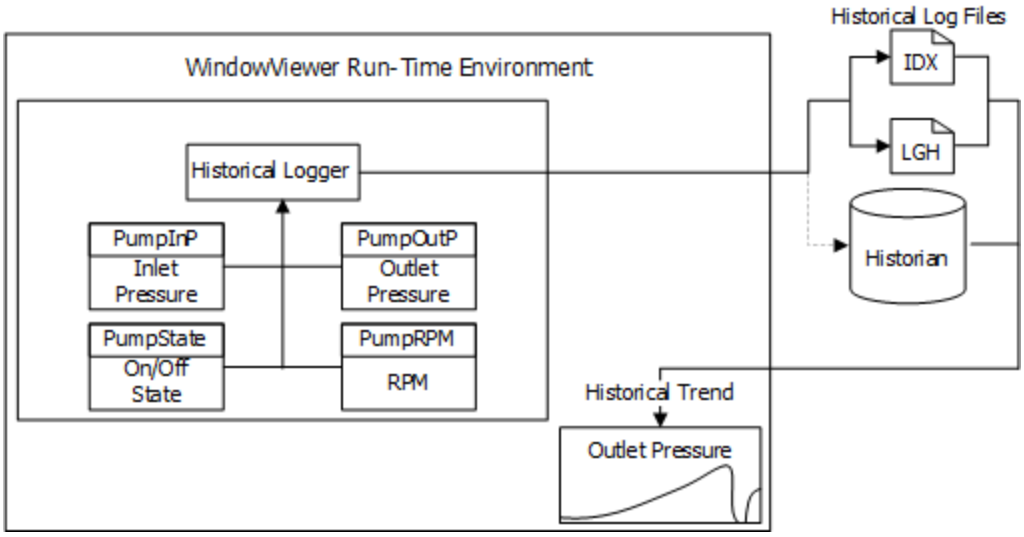
=VIEW|TAGNAME!'ActualTag_name'

Record tag values

While an InTouch application is running, its tag values can be logged and permanently saved. You can use this log data to create historical trend graphs that show some aspect of your plant processes over time.

Note: You should use Historian Server to save historical InTouch data for large applications, or when you want to create detailed reports. Refer to Historian Server documentation for details about configuring historical logging.

The following figure shows tag data from a pump saved to historical log files and/or Historian Server database. The InTouch Historical Logger writes a log entry each time a tag's value changes more than its specified log deadband range.



When storing data to historical log files, the InTouch HMI creates two log files. One file contains logged data stored in a proprietary format. The other file is an index to the data.

Names are assigned to the two log files with the following format:

YYMMDD00.LGH **and** *YYMMDD00.IDX*

where:

YY	Last two digits of the year the log files are created
MM	Two-digit number of the month the files are created (01-12)
DD	Two-digit day of the month the files are created (01-31)
00	Constant value of 00 in the log file names

A daily logging cycle begins and ends at midnight. The Historical Logger writes the last entries to the active log files at midnight and archives them. Two new files are created for the next day and data is logged to them.

Log files are saved for a specified number of days. Log files that are older than the retention period are deleted. For more information about setting the number of days to retain log files, see Configure general logging properties of the historical log file.

Configure historical logging

You need to perform three main tasks to configure historical logging for an InTouch application:

- Configure tags for historical logging
- Configure general logging properties for the InTouch application
  - Configure properties for storing tag values in InTouch historical log files and/or
  - Configure properties for storing tag values in a Historian server
- Optionally, set the historical logging frequency

Note: You can configure logging to LGH files and/or the Historian server.

Configure tags for historical logging

You select tags for historical logging from the Tagname Dictionary. When a selected tag's value changes, the Historical Logger determines if an entry should be written to the log based upon each tag's logging deadband and its current value.

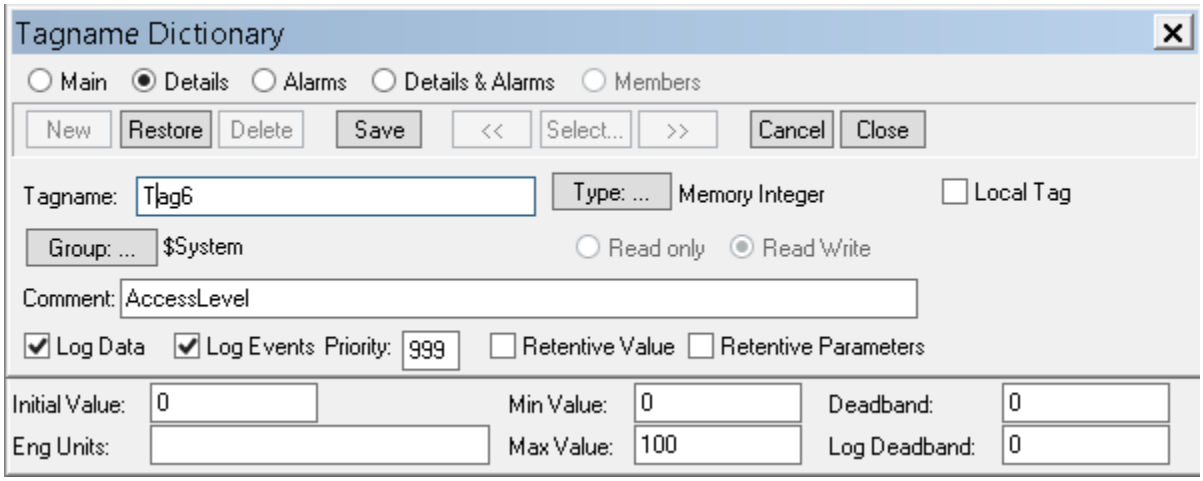
If you change a tag from logged to not logged, the data associated with the tag is no longer saved to the log file. Logging resumes when logging is re-enabled. However, the historical trend shows a gap during the period when logging was disabled.

Changes to tag logging are ignored while WindowViewer is running the application. Logging changes to tags do not become effective until the running application is stopped and restarted.

You configure logging for each individual tag from the Tagname Dictionary.

Enable historical logging for tags

1. If needed, stop WindowViewer from running the application.
2. Open the application with WindowMaker.
3. Open the Tagname Dictionary.
4. Select a tag from the Tagname Dictionary list whose data you want to log.
5. Select the **Log Data** checkbox.



The **Tagname Dictionary** dialog box includes other tag attributes closely associated with logging:

- **Log Deadband** sets a engineering units threshold that must be exceeded before a tag's value is written to the log file. Only new values outside of the deadband are written to the log file. Small value changes within the deadband range are ignored.
- The **Min EU** and **Max EU** properties scale clamped raw values within a range of engineering units. Minimum and maximum EU properties set the upper and lower boundaries of the scaled values.

  The Min/Max engineering units determine the range boundary for log values shown in a trend. By default, an InTouch historical trend shows log data from 0-100 percent of the EU range.

6. Select **Save**.
7. Repeat these steps to enable logging for each tag whose data you want to log.
8. Select **Close** to close the Tagname Dictionary when you are done.

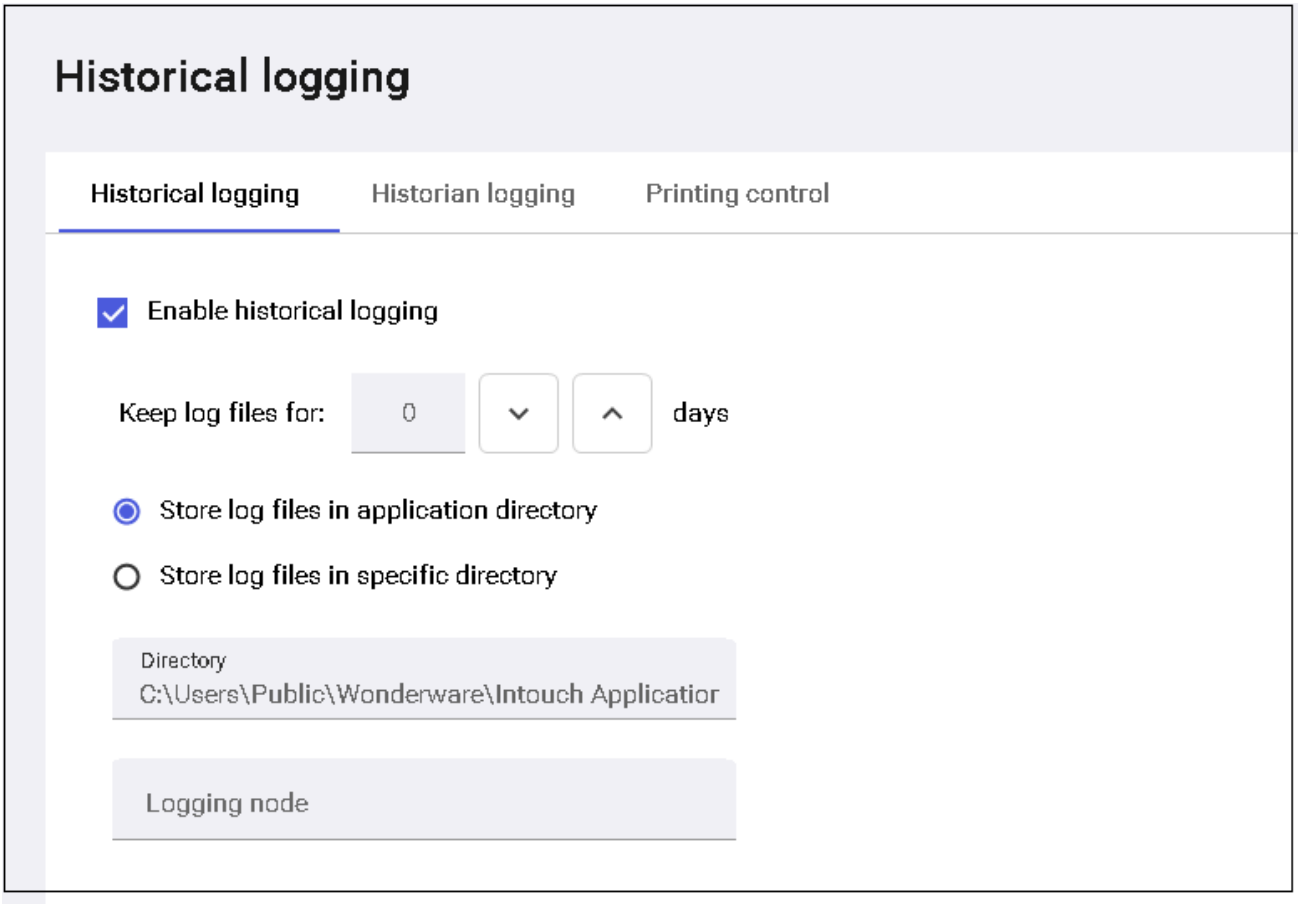
Configure general logging properties of the historical log file

You can set general logging properties that apply to the selected application.

Configure historical logging

1. If necessary, close the InTouch application running in WindowViewer.
2. Open WindowMaker.
3. On the **File** menu, select **Configure**, and then select **Historical Logging**.

   The **Historical Logging** configuration screen appears.



4. Select the **Enable historical logging** checkbox.
5. In the **Keep log files for** box, type the number of days prior to the current day to retain log files.

   Log files are kept for the current day and the number of days within the specified retention period. Log files that are older than the retention period are deleted. Setting the value to 0 retains all log files indefinitely.

   Example:

   Set the retention period to five days and began logging on the first day of the month. On the seventh day of the month, the log files are retained from the five previous days and the current day (02-07). The log files created on the first day of the month are deleted.

   Consider disk space usage when you set the number of days to save logging data. Historical logging stops if your hard disk runs out of free space. You must free disk space to resume logging.
6. Select the location of the folder to save log files.

Note: The folder path and the name of the file to store log data can be a maximum of 55 characters.

Select **Store log files in application directory** to save the log files in the same folder as the InTouch application creating the logged data.

Select **Store log files in specific directory** to specify another folder to store log files. You can specify the folder to store log files as:

- Windows folder path such as C:\History Log Files
- Universal Naming Convention (UNC) path such as \\node\share\directory.

  If you are saving log files to a distributed node, you must specify the directory as a UNC path.

  If configured to write historical data to the master application node's Application Directory, all NAD nodes try to write historical data to the master application. To avoid this, configure historical data on each NAD node to write to a local directory, not the master application node.

7. In the **Logging node** box, type the node name of the computer running the InTouch application creating log data.
8. Select **Save**.

   The logging configuration changes become effective immediately. Logging begins the next time you run the application.

For more information about setting up trend printing, see Print a trend at run time.

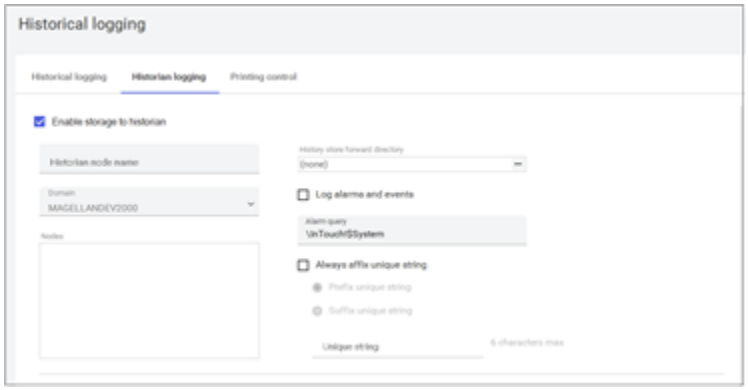
Configure general logging properties storage to Historian

You can set general logging properties that apply to the selected application.

Configure storage to Historian

1. If necessary, close the InTouch application running in WindowViewer.
2. Open WindowMaker.
3. On the **File** menu, select **Configure**, and then select **Historical logging.**

   The **Historical logging** configuration screen appears.
4. Select the **Historian logging** tab.



5. Select the **Enable storage to historian** checkbox.
6. In the **Historian node name** box, type the node name of the computer running the Historian server. You can also specify the IP address and if the server is installed on the same machine you can use ‘localhost’. Only the following special characters are allowed; full stop (.), underscore (_) and hyphen (-).

7. In the **History store forward directory** box, type the path to the local folder location where the files related to store forward are stored. The files will allow the historical data to stored temporarily if the connection to the Historian server is lost. After the connection is established, the Historian server will sync with the files from this store forward directory and preserve all information.
8. Select **Log alarms and events** to enable logging alarms and events. Specify the alarm query in the **Alarm query** text box.

   Note: If different InTouch applications store tag data to the same Historian server and use the same tag name, the InTouch applications will not detect such a scenario and may cause the Historian data to overlap. Use a unique prefix or suffix to differentiate between applications. For more information, see Configure the affix string.
9. Select **Save**.

   The logging configuration changes become effective immediately. Logging begins the next time you run the application.

For more information about setting up trend printing, see Print a trend at run time.

Configure the affix string

In scenarios when the same application is used on multiple nodes and connected to the same Historian Server, you can use a unique suffix or prefix to differentiate between tags from different nodes. You can configure the string via WindowMaker or by manually editing the 'dhistcfg.ini' file on each node.

1. Open WindowMaker.
2. On the **File** menu, select **Configure,** and then select **Historical Logging**. In the Historical Logging configuration screen, select the **Historian logging** tab.
3. Select the **Enable Storage to Historian** checkbox.
4. Select **Always affix unique string** checkbox.
5. Choose between **Prefix unique string** and **Suffix unique string**.
6. Enter the string in the **Unique string** field. The string can be a maximum of 6 characters.
7. Select **OK**.

Manually editing the affix string in the .ini file

You can manually update the affix string by editing the dhistcfg.ini file available within the application folder on each node.

1. Navigate to the application folder.
2. Edit the dhistcfg.ini file.

   Example1: Assign a prefix unique string of "aa"

```
szHistorianNode=<MachineName>
bStorageLoggingEnabled=1
bAffixEnabled=1
bPrefixEnabled=1
bSuffixEnabled=0
szHistUniqueString=aa
```

   Example2: Assign a suffix unique string of "xx"

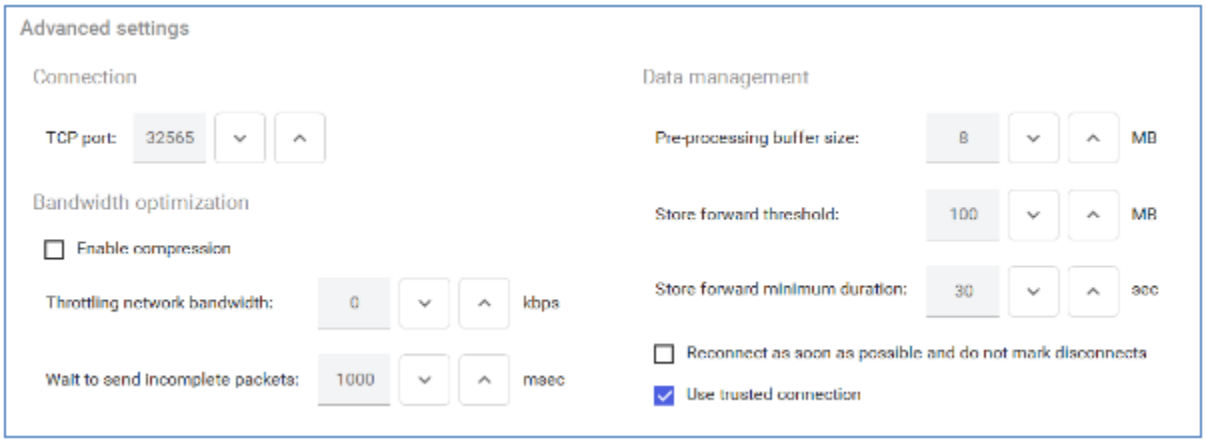
```
szHistorianNode=<MachineName>
bStorageLoggingEnabled=1
bAffixEnabled=1
bPrefixEnabled=0
bSuffixEnabled=1
szHistUniqueString=xx
```

3. Save and close the .ini file.

Any changes will be affective after WindowViewer is restarted.

Configure advanced settings

Update the **Advanced Settings** section to specify settings related to the Historian connection.



1. Under **Connection**, you can specify the **TCP Port**. The TCP port on the Historian Server node to which history data will be sent. The TCP port is configured when the Historian Server is installed. The default port is 32565. The Default Historian TCP port is configurable and is used for data replication and communication with remote IDAS version 2023 R2 and later (which implies gRPC communication).

   Note: The classic Historian TCP port 32568 which supports WCF communication, was used for data replication and remote IDAS communication with Historian versions 2023 and earlier, and remains for compatibility with previous versions. The default port number for applications created in InTouch HMI 2023 and earlier is 32568. During upgrade, the TCP Port of the migrated application will be automatically updated to 32565. However, if you have manually modified the port number, the modified port number will be retained in the upgraded application.

2. Under the **Bandwidth Optimization** section, you can select the **Enable compression** checkbox. Once selected you can specify:
   - **Throttling network bandwidth**: Specifies limit of bandwidth usage, in kbps, for network communication used by HCAL when communicating with the Historian. A value of 0 disables this feature (default). For more information on estimating your bandwidth needs, see the performance and sizing recommendations for the Historian Server in System Platform Installation. The allowed values are 0 kbps to 65535 kbps.
   - **Wait to send incomplete packets**: Specifies the maximum time, in milliseconds, for keeping a partially-filled Historian Client Access Layer (HCAL) buffer before sending it to the Historian Server. If the buffer is full, then it is sent immediately, regardless of any value configured in this field. If you have fast-changing data, this setting is irrelevant. If you have slow-changing data and limited network bandwidth, you might

want to increase this value so that you have less "chatter" on the network. The allowed values are 1000 milliseconds to 30000 milliseconds.

3. Under the **Data Management** section, you can specify:
   - **Pre-processing buffer size**: The total size, in MB, of all buffers used by Historian Client Access Layer (HCAL). The default and minimum value is 8. If you have very high data bursts, then you should increase this value. If this is too low, data loss occurs and error messages related to buffer overflows appear in the Logger. Increase this value in increments of 10 MB. The allowed values are 8 MB to 65535 MB
   - **Store forward threshold**: The size, in MB, of free space to reserve on the HCAL store-and-forward disk. The space designated will not be used during store-and-forward. This value cannot be a negative number. The allowed values are 0 MB to 65535 MB
   - **Store forward minimum duration**: The minimum duration, in seconds, for HCAL to function in store-and-forward mode. HCAL will function in store-and-forward mode for this length of time even if the condition that caused HCAL to function in store-and-forward mode no longer exists. The allowed values are 30 seconds to 3600 seconds.
   - Select the **Reconnect as soon as possible and do not mark disconnects** checkbox: Specifies how trends appear during communication disconnects between Application Server and Historian. It does not affect how trends of history data appear after communications have been restored. If TRUE, no gap will appear in client-side trends for the disconnect interval. While disconnected, the interval is filled in with the last-received value before the disconnect. If FALSE, a gap will appear in client-side trends for the disconnect interval. NULL values are injected on disconnect to create the gap. In both cases, after reconnect, the interval will be filled in with store-and-forward data.
   - Select the **Use trusted connection** checkbox: Ensures only secure communication with the Historian Server.
     The **Use trusted connection** checkbox is enabled by default for all new and migrated applications
     - When the **Use trusted connection** checkbox is selected, only trusted connections will be able to communicate with the Historian server. If there are no common certificates between Client and Historian Node, the connection will fail.
     - When the **Use trusted connection** checkbox is not selected, a connection to Historian will be established, even when there are SSL errors in the logger.

       Note: We recommend you to configure SMS in the Configurator along with selecting this checkbox to ensure a secure connection.
4. Select **Save**.

Services to support WCF and gRPC communication

Ensure the following services are running to support backward compatibility when the Historian version is 2023 R2 or higher.

- aahClientAccessPoint (WCF communication) - Supports clients of versions 2023 and earlier.
- aahClientAccessPointNG (gRPC communication) - Supports clients of versions 2023R2 and higher.

Control historical logging frequency

You can optionally specify that entries are logged based upon two conditions:

- An immediate log entry is written whenever a tag value changes by an engineering unit value greater than its log deadband value.
- The current values of all logged tags are written at a fixed interval. Entries for all logged tags are written regardless of their current values. The default fixed interval is 60 minutes.

You can accept the default fixed logging interval or add two parameters to the intouch.ini file to change the interval.

- *ForceLogging*

  ForceLogging specifies the length of the fixed logging interval in minutes. *ForceLogging* can be set to a value between 5 and 120 minutes. The default is *ForceLogging=60*.
- *ForceLogCurrentValue*

  ForceLogCurrentValue specifies that, at the fixed logging interval, the current tag value is logged, even if it is still within the log deadband of the last logged value. When set to 0, the last logged value is logged again. The default is *ForceLogCurrentValue=0*.

The following example shows an example of the intouch.ini file that includes the two logging parameters.

```
WinFullScreen=1
WinWidth=808
AlarmBufferSize=5000
ForceLogging=5
ForceLogCurrentValue=1
```

In this example, tag values are written to the Historical Log file at five minute intervals.

Modify the historical logging frequency

1. Close WindowMaker and WindowViewer.
2. Locate the intouch.ini file in the same folder as the InTouch application.
3. Edit the intouch.ini file.
4. Insert a *ForceLogging* statement with a value between 5 and 120.
5. Insert a *ForceLogCurrentValue=1* statement.
6. Save your changes and close the intouch.ini file.
7. Restart WindowViewer.

Start and stop historical logging at run time

When an application is running, you can manually stop and restart historical logging with commands from the WindowViewer **Special** menu.

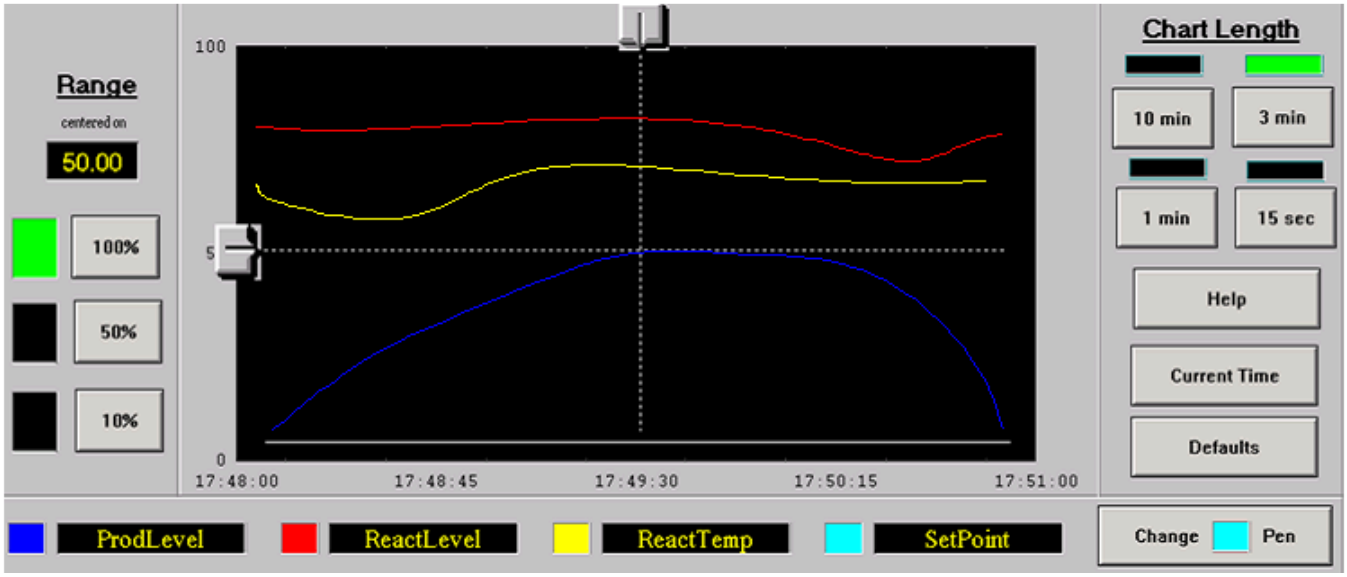
- The **Stop Historical Logging** command stops logging for the duration of the current application session. Logging remains stopped for the current session until it is manually started again.
- The **Restart Historical Logging** command restarts logging after it is manually stopped with the **Stop Historical Logging** option.

You can also add a button to your application and write a QuickScript that includes the **$HistoricalLogging** system tag to start and stop historical logging. Logging starts when **$HistoricalLogging** is assigned a value of 1. Logging

stops when **$HistoricalLogging** is assigned a value of 0. For more information about the **$HistoricalLogging** system tag, see System tags.

Trending tag data

You can create trends that graphically show data collected from an InTouch application. WindowMaker includes a set of utilities and wizards that enable you to create historical and real-time trends. The following figure shows an example of an InTouch Average/Scatter trend.



You can also use a set of trend controls. Using these controls, you can select the data shown in a trend and how data appears in the trend.

You can configure real-time and historical trends. Both trend types include configuration options to set a trend's data collection interval and visual appearance.

Types of InTouch trends

A historical trend shows log data collected from the past and stored in InTouch data repositories.

Using a distributed history system, you can retrieve historical data from any InTouch historical log file located on an accessible network node. The distributed history system extends the retrieval capabilities of historical trends to include remote log databases.

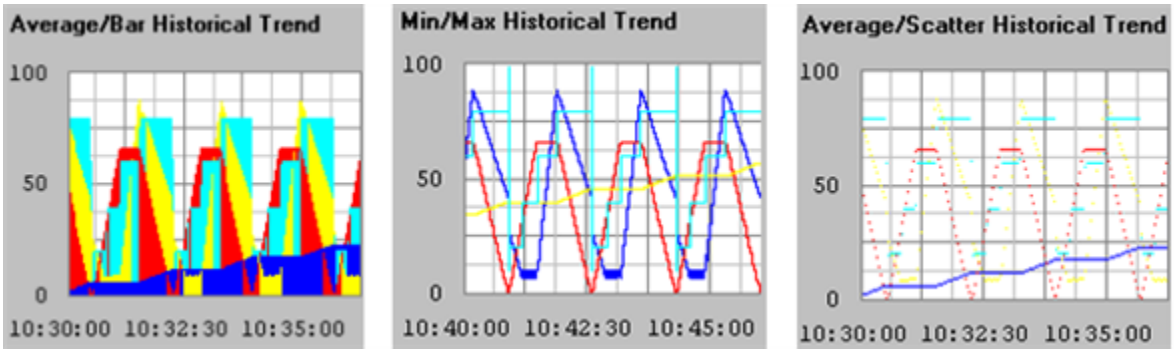
Real-time trends update continuously to show data as it occurs over relatively short periods. You can use WindowMaker's Real-time Trend tool to create a trend object in a window. If the optional 16-Pen Trend tool is installed, you can also create real-time trends that show data from up to 16 tags.

Understand historical trends

Historical trends show a contiguous segment of data from the past. Unlike real-time trends, historical trends are updated only by a script or an operator action.

A historical trend shows a graphical representation of data from a maximum of eight tags. You assign the data that appears in a historical trend by assigning a tag to a trend pen.

The figure below shows the three types of InTouch historical trends.



- The Average/Bar historical trend shows the average value of a data point during the time intervals in bar form.
- The Min/Max historical trend shows the changes in the percentage of engineering units scale as a vertical line over the time span. The emphasis is on time flow and rate-of-change, rather than amount of change.
- The Average/Scatter historical trend shows the average value of the data points over each trend time interval.

You can create graphical sliders called scooters to access the details of trend data based on the scooter's current position within a trend. For example, when the operator positions the scooter over an area on the trend that has visible data, the time and values at that location for all database values being trended are shown.

You can also create buttons or sliders to zoom in and out between the scooters or to data, such as the maximum to minimum value. Average and standard deviation can be shown for the complete chart or for the area between the scooters.

Historical trends can also be scrolled by any amount of time. You can create custom scales and link them to the .**MinEU** and .**MaxEU** dotfields to create a trend that shows the full range of data set by its engineering unit.

Understand real-time trends

A real-time trend shows data from an InTouch application that is currently running. Real-time trends are continuously updated. Real-time trends plot current data values associated with up to four local tags or expressions.

You can:

- Create a real-time trend
- Select the tags for a trend
- Specify the time span and update interval of a trend
- Configure the display options of a trend

Show saved tag values in a historical trend

You can create historical trends by using any of the following WindowMaker utilities:

- Historical Trend tool

- Historical Trend Wizard
- 16-Pen Trend Wizard (Optional)

In addition, you can incorporate a ActiveFactory trend to show InTouch historical trend data saved to a Historian database.

Use historical trend objects

You can create and configure a trend with the WindowMaker Historical Trend tool. You can:

- Create a historical trend
- Select the tags for a trend
- Specify the time span and update interval of a trend
- Configure the display options of a trend

Create a historical trend

You can use the Historical Trend tool to create a trend object in a window. The first time you create a historical trend object, the InTouch default configuration settings are used.

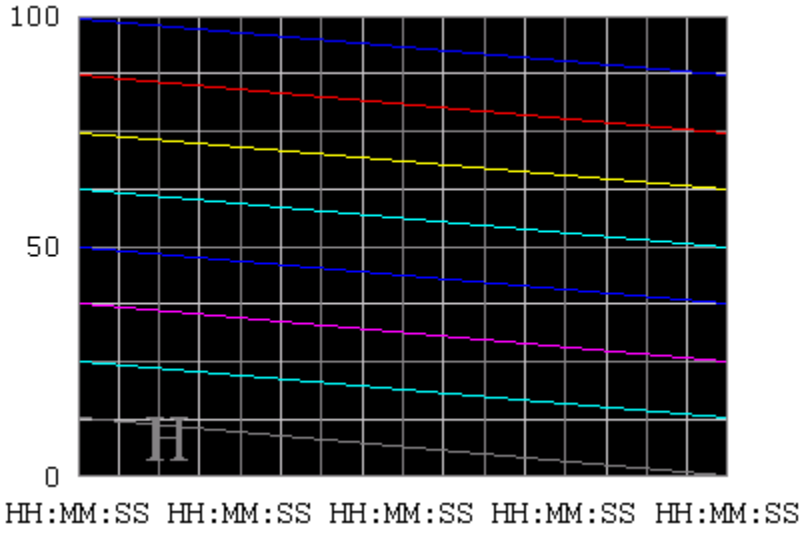
After you configure a historical trend, WindowMaker uses the most recent configuration values as the initial settings for a new trend object.

You can draw the trend chart to any size within the borders of the window.

Create a historical trend

1. Open the window in WindowMaker in which you want to place a historical trend.
2. On the **Draw** menu, in the **Trends** group, select **Historical**.
3. Move the mouse over the window area where you want to place the historical trend. Drag the mouse diagonally to create a rectangle the desired size of the trend.

   The Historical Trend object appears in the window.



4. If needed, adjust the height and width of the trend with its object handles.

Configure which tags to show from a historical trend

A historical trend pen creates a graphical representation of logged data from a specified period. You assign trend pens to the tags that collect historical data.

A historical trend supports up to eight pens.

Note: WindowViewer must be closed. Otherwise, the Pen boxes cannot be selected.

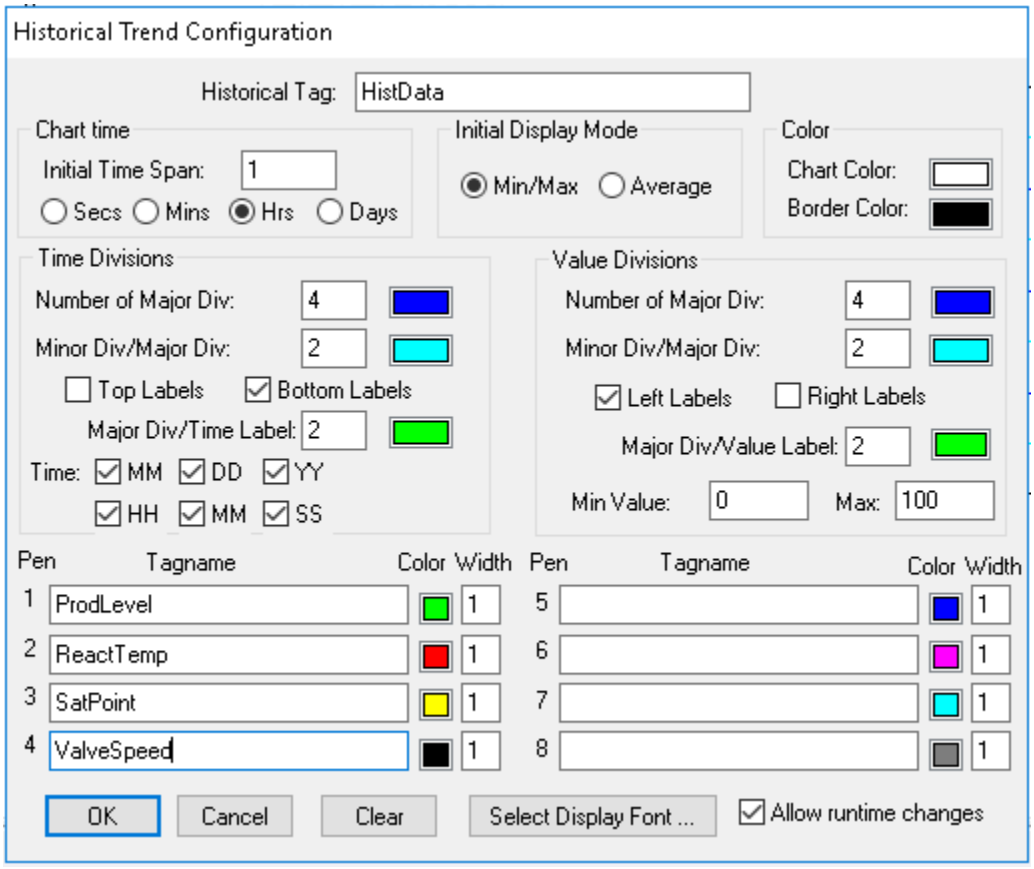
You can select tags from remote history providers, if the providers are configured. For information about setting up a remote history provider, see Distributing Applications in *AVEVA™ InTouch HMI Application Deployment*.

Note: You can also configure a IndustrialSQL Server history provider to visualize historical trend data. For more versatility and other charting options, use the ActiveFactory trend tools to create trends with InTouch historical data saved to a IndustrialSQL Server database.

Configure which tags to display from a historical trend

1. Double-click the trend object in the window.

   The **Historical Trend Configuration** dialog box appears.

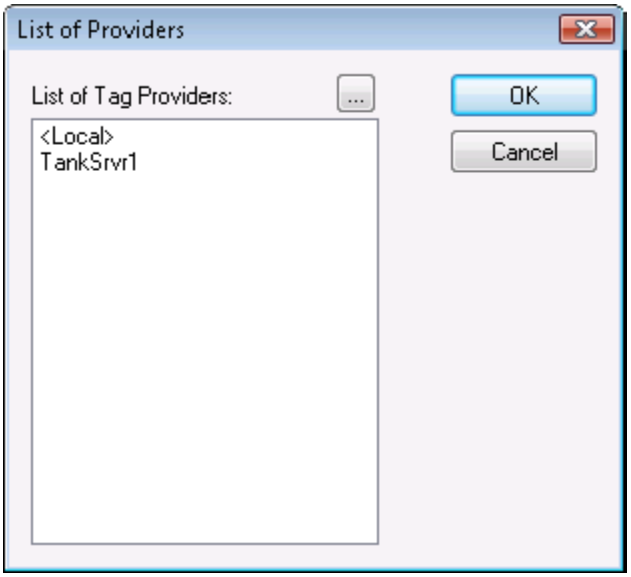


2. In the **Historical Tag** box, type the tag that you want to use for the trend.

   The tag must be defined as a Hist Trend type. You must assign a different Hist Trend tag to each historical trend in an InTouch application.

   If the tag you enter is not currently defined in the Tagname Dictionary, a dialog box appears and asks if you want to create a tag. If you select **OK** to define a tag, the **Tagname Dictionary** dialog box appears. This tag does not support 128 characters.

3. In the **Tagname** area, specify the name of an existing local or remote tag in one or more **Pen** boxes.
4. To assign an existing local or remote tag directly, select a **Pen** box and type the name of the tag.
5. To browse to the tag to assign:
   a. Double-click in a **Pen** box. The **List of Providers** dialog box appears.



   b. Select the tag provider you want to use for the pen.
   c. Select **OK** to show a dialog box listing the tags for the selected provider.
   d. Double-click on a tag from the list to select it.
6. Double-click the color box next to each pen assigned a tag to show the color palette. Select the color for the pen.
7. In the **Width** box, type the line width in pixels for each pen shown in the trend.
8. Repeat steps 3 to 7 for each tag that you want to assign to a historical trend pen.
9. Select the **Allow runtime changes** checkbox to allow an operator to configure a historical trend while the application is running.

   For more information about updating a historical trend during run time, see Change the trend configuration at run time.

Configure the time span of a historical trend

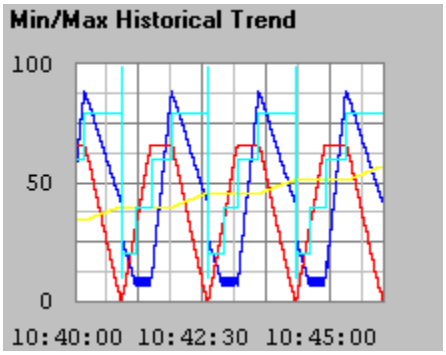
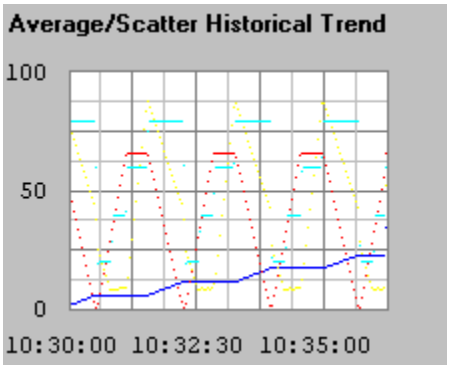
You can configure the time span of a historical trend.

Configure the time span of a historical trend

1. Double-click on the trend object to show the **Historical Trend Configuration** dialog box.
2. In the **Chart Time** area, type the length of time in **Initial Time Span** that you want to appear on the horizontal x-axis of the trend.
3. Select the time unit of measure: Seconds (Secs), Minutes (Mins), Hours (Hrs), or Days (Days).

   For example, it you enter 8 in **Initial Time Span** and then select **Hrs**, the time span shown on the trend is 8 hours.
4. In the **Initial Display Mode** area, select the type of historical trend that appears when WindowViewer

initially shows the window containing the trend.

Initial Display Mode	Description
Min/Max	The chart shows changes in the percentage of engineering units scale as a vertical line over the time span. 
Average	Each pixel within a trend time segment shows the average value of a tag over the period of time within the segment. 

5. Go to Configure historical trend display options to configure the visual appearance of a historical trend.

Configure historical trend display options

You can configure the visual appearance of a historical trend.

Configure historical trend display options

1. Double-click the trend object. The **Historical Trend Configuration** dialog box appears.
2. Set the color options. Do the following:
   - In the **Color** area, select the **Chart Color** box to open the color palette.
   - Select a color in the palette as the background for the trend.

     White is the default background color. Any other background color increases the time needed to print a trend.
   - Select **Border Color** to open the color palette.
   - Select a color in the palette as the border color of the trend.

3. Set the time divisions options. Do the following:
   - In the **Time Divisions** area, type the number of major trend time divisions in **Number of Major Div.**

     The major time divisions appear on the horizontal time axis of the trend. The maximum time between major time divisions is 65536 seconds, or 18 hours, 12 minutes, 16 seconds.
   - Select the color for the major division lines.
   - From **Minor Div/Major Div**, type the number of minor time divisions within each major time division.

     The number of minor time divisions should be evenly divisible within a major division. For example, if the major division is set to 60 seconds, entering a value of 2 in **Minor Div/Major Div** sets the minor time division to 30 seconds.
   - Select the color for the minor division lines.
   - Select **Top Labels** or **Bottom Labels** to specify the placement of time labels on the trend.
   - If you are using time labels, type the number of major time division lines per time label in the **Major Div/ Time Label** box.
   - Select the color of the time division labels.
   - Select the time units shown as the label of the major time division.

Months (MM)	Hours (HH)
Days (DD)	Minutes (MM)
Years (YY)	Seconds (SS)

4. In the **Value Divisions** area, configure the appearance of the vertical axis of the trend.

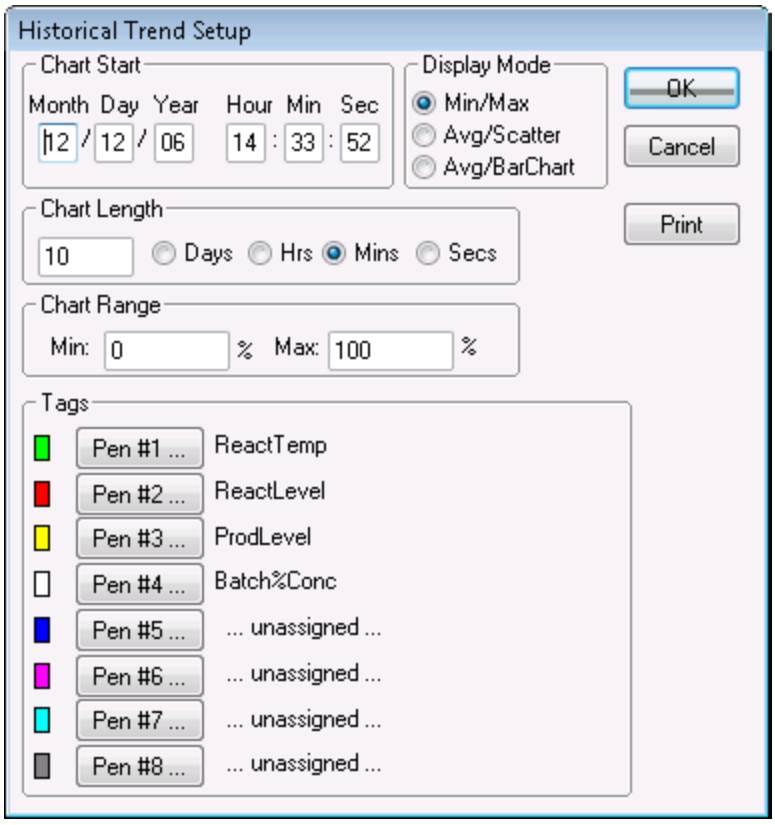
   Value Divisions options are configured the same way as the **Time Divisions** options. The vertical axis specifies the range of data values that appear in the trend based upon engineering units for all tags.
5. Select **OK** to save your configuration changes and close the **Historical Trend Configuration** dialog box.

Change the trend configuration at run time

If you select the **Allow runtime changes** option when you configure your historical trend, operators can configure a historical trend while it is running. Operators configure the trend from a dialog box that appears after selecting the trend from a displayed window.

Configure a historical trend during run time

1. Select the historical trend while it is running. The **Historical Trend Setup** dialog box appears.



2. In the **Chart Start** area, type the starting date and time of the historical trend data collection interval.
3. In the **Display Mode** area, select the type of historical trend chart.

   The trend display mode affects performance. The primary factor that determines trend performance is the length of the lines shown in a trend. The longer the lines, the longer it takes to generate the trend.

   Line width also affects performance. Wide lines take significantly longer to draw. Min/Max or Average/Scatter trends can be created more quickly than an Average/Bar Chart.
4. In the **Chart Length** area, type the duration to be displayed on the trend and then select the unit of measure.

   For example, if you enter 2 and select **Hrs**, the trend duration is 2 hours.
5. In the **Chart Range** area, type the percentage of engineering unit range shown on the vertical axis of the historical trend.

   The scale of the trend is a segment of the trended tag's engineering unit range defined by a percentage range. The values range from 0 to 100. For example, if you want to trend the variance of the selected tags from 40 to 60 percent of their engineering unit range, type 40 and 60 in the **Min** and **Max** range boxes, respectively.
6. In the **Tags** area, select a pen number to assign a tag.

   The **Select Tags** dialog box appears with a list of tags that can be assigned to the historical trend pen.
7. Use the following statement in a QuickScript or button to allow the operator to update the chart:

   ```
   Hist_TrendTag.UpdateTrend = 1;
   ```
8. Use any of the following functions in a QuickScript or on a button:

   HTUpdateToCurrentTime(Hist_Tag);

   HTScrollLeft(Hist_Tag,Percent);

   HTScrollRight(Hist_Tag,Percent);

HTZoomIn(Hist_Tag,LockString);

HTZoomOut(Hist_Tag,LockString);

HTSetPenName(Hist_Tag,PenNum,Tagname);

For more information about using scripts containing trend functions, see Control a historical trend wizard using scripts.

9. Change any of the following trend tag dotfields:

   .ChartStart

   .ChartLength

   .MaxRange

   .MinRange

   .Pen1-.Pen8

   For more information about using dotfields with historical trends, see Control a historical trend using dotfields.

Control a historical trend using dotfields

You can use dotfields to manage historical trends during run time.

.DisplayMode Dotfield

The .**DisplayMode** dotfield specifies the trend format used to display a tag's values.

Category

Historical

Usage

```
tag_name.DisplayMode
```

Parameter

tag_name
Any Hist Trend tag.

Data Type

Analog (read/write).

Valid Values

1 = Shows the min/max value that occurred in each sample period (default).

2 = Shows the average value of each sample period in a scatter historical trend.

3 = Shows the average of each sample period in a bar chart historical trend.

Example

This statement specifies that the values in the historical trend represented by the "HistTrend_Tag" are formatted as a bar chart historical trend.

```
HistTrend_Tag.DisplayMode=3;
```

See Also

.ChartLength, .ChartStart

.MinRange Dotfield

The .**MinRange** dotfield specifies the minimum percentage of the tag's engineering unit range to show for each tag in a Historical trend.

Category

Historical.

Usage

```
tag_name.MinRange
```

Parameter

tag_name

Any Hist Trend tag.

Remarks

A historical trend can show several types of tags at the same time. Specifying the minimum and maximum boundaries of the value range in engineering units is difficult because different types of tags can have different engineering ranges. Therefore, the minimum and maximum range values are expressed as a percentage of the engineering range of each tag. This way, regardless of the tag's true engineering range, the historical trend shows the indicated percentage of that tag's particular engineering range.

Data Type

Real (read/write).

Valid Values

The limits for the .MaxRange and .MinRange dotfields are from 0 to 100. .MinRange is always less than

.MaxRange. If you assign a value less than 0 or greater than 100 to either of these dotfields, the value is clamped at 0 or 100. If .MinRange is greater than or equal to .MaxRange, the trend does not show any data.

Example

This example dotfield statement sets the minimum percentage range of the historical trend to 25 percent of the possible engineering unit range of a Hist Trend tag.

```
HistTrend.MinRange=25
```

See Also

.ChartStart, .ChartLength, .DisplayMode, .EngUnits, .MinEU, .MaxEU, .MaxRange, .MinRaw, .MaxRaw, .RawValue

.MaxRange Dotfield

The .**MaxRange** dotfield specifies the maximum percentage of the tag's engineering unit range to show for each tag in a Historical trend.

Category

Historical.

Usage

```
tag_name.MaxRange
```

Parameter

tag_name

Any Hist Trend tag.

Remarks

A historical trend can show many types of tags simultaneously. Specifying the minimum and maximum boundaries of the range in engineering units can be difficult because tags can have different engineering ranges. Therefore, the minimum and maximum range values are expressed as a percentage of the engineering range of each tag. This way, regardless of the tag's true engineering range, the historical trend shows the indicated percentage of that tag's engineering range.

Data Type

Real (read/write).

Valid Values

The limits for .MaxRange and .MinRange dotfields are from 0 to 100. .MinRange is always less than .MaxRange. If you assign a value less than 0 or greater than 100 to either of these dotfields, the value is clamped at 0 or 100. If .MinRange is greater than or equal to .MaxRange, the trend does not show any data.

Example

This example dotfield statement sets the maximum percentage range of the historical trend to 75 percent of the possible engineering unit range of a Hist Trend tag.

```
HistTrend.MaxRange=75
```

See Also

.ChartStart, .ChartLength, .DisplayMode, .EngUnits, .MinEU, .MaxEU, .MinRange, .MinRaw, .MaxRaw, .RawValue

.UpdateCount Dotfield

The .**UpdateCount** dotfield increments a count each time a historical trend is updated. The .**UpdateCount** dotfield can be used as a trigger for further functions.

Category

Historical.

Usage

```
HistTrendTag.UpdateCount
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Data Type

Integer (read-only).

Valid Values

Any positive integer.

Example

This example uses the **HTGetValueAtScooter()** function to retrieve the value of Pen1 at the right scooter's

current position. A change to any function argument causes the function to be re-evaluated. When the update completes and the value of .**UpdateCount** is incremented, this statement is re-evaluated.

```
MyRealTag=HTGetValueAtScooter( MyHistTrendTag,MyHistTrendTag.UpdateCount, 2,
MyHistTrendTag.ScooterPosRight, 1, "PenValue");
```

See Also

.UpdateInProgress, .UpdateTrend

.UpdateInProgress Dotfield

The .**UpdateInProgress** dotfield indicates the current status of a historical trend update operation. The value of the dotfield is set to 1 if a historical retrieval is in progress; otherwise the dotfield is set to 0.

Category

Historical.

Usage

```
HistTrendTag.UpdateInProgress
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

Whenever new data is requested from the historical trend, this dotfield's value is set to 1. After the process completes, .**UpdateInProgress** is reset to 0. .**UpdateInProgress** can be used in functions related to historical trends.

If the operator scrolls the trend to a period outside the currently shown period, it can take some time to retrieve the historical data. The .**UpdateInProgress dotfield** provides a way to alert the operator that the requested data is being retrieved. Without feedback, the operator may not be aware that the trend is being updated.

Data Type

Discrete (read-only).

Value Values

0 = No update in progress

1 = Update in progress

Example

The .**UpdateInProgress** dotfield is typically used as the expression in a visibility link on a text object near or on the scroll buttons of a Historical Trend. You can use the .UpdateInProgress dotfield to show "Busy" on the window when the data is being retrieved with the following message value display animation link:

```
DText(HistTrend1.UpdateInProgress,"Busy","Ready")
```

See Also

.UpdateCount, .UpdateTrend

.UpdateTrend Dotfield

The **.UpdateTrend** dotfield triggers an update to a historical trend. Using the .**UpdateTrend** dotfield in a button action script, the operator can manually update the trend during run time.

Category

Historical.

Usage

```
HistTrendTag.UpdateTrend
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

Historical trends do not automatically update. A change must be made to either the .**ChartStart** or the .**ChartLength** dotfields to update the chart and show the current values for the specified tags. By using this dotfield in a button action script, the operator can update the chart during run time. You can also use this dotfield in a QuickScript if other dotfields associated with the historical trend are going to be changed.

You should only set the .UpdateTrend dotfield to a value of 1.

Data Type

Discrete (write only).

Valid Values

1

Example

This example triggers the historical trend associated with the **MyHistTrendTag** tag to update with the current values of all parameters.

```
MyHistTrendTag.UpdateTrend=1;
```

.ChartLength Dotfield

The .**ChartLength** dotfield specifies the length of time shown in a Historical trend.

Category

Historical.

Usage

```
HistTrendTag.ChartLength
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

The value assigned to .ChartLength specifies the length of the chart in seconds. The length is defined as the amount of time currently shown on the Historical Trend Chart. More specifically, the calculation retrieved as the Chart Length from a Historical Trend Chart is:

```
ChartLength=(Date/Time Stamp on Right-Hand Side of Chart) - (Date/Time Stamp on Left-Hand Side of Chart);
```

Because Date/Time Stamps are expressed in seconds from midnight on January 1, 1970, the calculation results in seconds of time displayed between the left and right sides of the chart.

Whenever adding or subtracting from .**ChartLength**, time is expressed in seconds. Therefore, to subtract two hours from the current .**ChartLength,** convert hours to seconds before performing the calculation. For example:

```
(2 hours) * (60 minutes/hour) * (60 seconds/minute) = 7200 seconds.
```

Data Type

Integer (read/write).

Valid Values

Any positive integer.

Examples

This example forces the length of the historical trend to one hour.

```
HtTag.ChartLength=3600 {60 minutes * 60 seconds/minute};
```

This example scrolls the trend left by 50 percent.

```
HtTag.ChartStart=HtTag.ChartStart - HtTag.ChartLength / 2;
```

This example scrolls the chart left by 10 percent.

```
HtTag.ChartStart=HtTag.ChartStart - (.10 * HtTag.ChartLength);
```

See Also

.ChartStart

.ChartStart Dotfield

The .**ChartStart** dotfield can be used to set or verify the value of the starting (left side) date/time stamp of a historical trend.

Category

Historical.

Usage

```
HistTrendTag.ChartStart
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

This read/write dotfield is used to set or verify the value of the starting date/time stamp of a historical trend. The .**ChartStart** dotfield is expressed as the number of elapsed seconds after midnight January 1, 1970. The starting point is defined as the first date/time stamp on a historical trend.

Data Type

Integer (read/write).

Valid Values

Any positive integer.

Example

The following statement scrolls the chart to the right by one minute.

```
HtTagname.ChartStart=HtTagname.ChartStart + 60;
```

See Also

.ChartLength

.Pen1-8 Dotfields

The .**Pen1-8** dotfields assign a logged tag to a historical trend pen.

Category

Historical

Usage

```
HistTrendTag{.Pen1 | .Pen2 | .Pen3 | .Pen4 | .Pen5 | .Pen6 | .Pen7 | .Pen8};
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

You assign tags to trend pens using the .Pen1-8 dotfields with the following format:

HistTrend.Pen*X* = Tag_Name.TagID

Where *X* is an integer 1 to 8.

It is recommended that you use the HTSetPenName() and HTGetPenName() functions if possible.

Note: Only local tags can be assigned to a .PenX dotfield. The provider.tag notation cannot be used. The provider.tag can be used only with the HTSetPenName() function.
A good reference to use when learning how these dotfields work is a historical trend wizard placed on the screen and broken apart.

Data Type

TagID (read/write).

Valid Values

This dotfield data type is type **TagID**. This means only the handle of a tag can be assigned to the **.Pen1-8**

dotfields. You cannot directly assign the name of a tag to the **.Pen1-8** dotfields. You must associate the associated .**TagID** dotfield of a tag to a .**Pen1-8** dotfield using the following syntax:

```
HistTrendTag.Pen1=LoggedTag.TagID;
```

In general, a TagID type tag can be equated only to another TagID tag. It cannot be used with any other tag type unless the .**TagID** dotfield extension is added to the other tag.

Although the .**Pen1-8** dotfields are considered read/write, their values cannot be directly shown on the screen.

Examples

The following example assigns a new tag to the .**Pen5** dotfield of the historical trend associated with the Hist Trend tag. You must append the .**TagID** dotfield to the name of the logged tag in order to assign it to .**Pen5** dotfield.

```
HistTrendTag.Pen5=PumpPress.TagID;
```

Working from the previous example, you can show the name of the tag assigned to HistTrendTag.Pen5. Creating a legend that shows the tag assigned to each trend pen is useful information for an operator.

You cannot show the tag assigned to HistTrendTag.Pen5 in a Message Display link. The actual value of the .Pen5 dotfield is an integer that represents a memory location within WindowViewer, which is not useful for display purposes. You need to create a new TagID type tag called **Pen05**. Place the following statement underneath the statement from the previous example:

```
Pen05=HistTrendTag.Pen5;
```

In the first example, the PumpPress tag is assigned to pen 5 of HistTrendTag. In this example, Pen05 is assigned the value of Pen5 of the HistTrendTag, which is the TagID of the PumpPress tag.

The .Pen1-8 dotfields are pointers to the tags that are associated with pens selected to appear in a trend. The .Pen1-8 dotfields are of a special data type, namely .TagID. After you make the assignment, you can use the .Name dotfield of the TagID tag to show the name of the tag.

.TagID Dotfield

The .**TagID** dotfield can be used in conjunction with the .**Pen1** - .**Pen8** dotfields to assign a tag to a historical trend pen.

Category

Historical tag.

Usage

```
tag_name.TagID
```

Parameter

tag_name

Any discrete, integer, real, indirect discrete, or indirect analog tag.

Remarks

The .**TagID** dotfield provides the handle of a tag and is used mainly to assign tags to pens in a historical trend.

Data Type

TagID (read-only).

Example

This example uses the .**TagID** dotfield to assign the **PumpRPM** tag to pen 6 of the historical trend.

```
HistTrendTag.Pen6=PumpRPM.TagID;
```

See Also

.Pen1-.Pen8

.ScooterLockLeft Dotfield

The .**ScooterLockLeft** dotfield specifies whether an operator can move the right scooter further left than the left scooter's current position on the historical trend.

Category

Historical.

Usage

```
HistTrendTag.ScooterLockLeft
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

In general, you should prevent an operator from moving the right scooter further left than the left scooter's current position. When the left scooter is locked, it forces the right scooter position to be equal to the left scooter position whenever the right scooter overtakes the left scooter.

Data Type

Discrete (read/write).

Valid Values

0 = False. Right scooter can move further left than the left scooter’s current position on the historical trend

1 = True. Right scooter cannot move further left than the left scooter’s current position on the historical trend.

Example

The following example prevents the right scooter from moving further left than the left scooter's current position on the historical trend.

```
HistTrendTag.ScooterLockLeft=1;
```

See Also

.ScooterPosRight, .ScooterPosLeft, .ScooterLockRight

.ScooterLockRight Dotfield

The .**ScooterLockRight** dotfield specifies whether an operator can move the left scooter further right than the right scooter's current position on the historical trend.

Category

Historical.

Usage

```
HistTrendTag.ScooterLockRight
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

In general, you should prevent the operator from moving the left scooter further right than the right scooter's current position. When the right scooter is locked, if forces the left scooter position to be equal to the right scooter position whenever the left scooter overtakes the right scooter.

Data Type

Discrete (read/write).

Valid Values

0 = False. Left scooter can move further right than the right scooter's current position on the historical trend.

1 = True. Left scooter cannot move further right than the right scooter's current position on the historical trend.

Example

The following example prevents the left scooter from moving further right than the right scooter's current position on the historical trend.

```
HistTrendTag.ScooterLockRight=1;
```

See Also

.ScooterPosRight, .ScooterPosLeft, .ScooterLockLeft

.ScooterPosLeft Dotfield

The **.ScooterPosLeft** dotfield dynamically controls the position of the left scooter on a historical trend.

Category

Historical

Usage

```
HistTrendTag.ScooterPosLeft
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

This read/write dotfield dynamically controls the position of the left scooter. You can use this dotfield in a QuickScript function to retrieve the current position of the left scooter, or you can assign a value to this dotfield to adjust the position of the left scooter to another location on the trend.

This dotfield is most often used in conjunction with the set of **HTGetValue()** functions. These functions must specify which historical trend is being queried, as well as the current position of the trend's scooters.

Data Type

Real (read/write).

Valid Values

0.0 to 1.0; where 0.0 is the extreme left of the historical trend chart, and 1.0 is the extreme right of the historical trend chart.

Examples

The following example repositions the left scooter. The left scooter moves to a location 34 percent of the chart's total length from the left side of the historical trend chart currently associated with the MyHistTrendTag tag.

```
MyHistTrendTag.ScooterPosLeft=.34;
```

In the following statement, the QuickScript **HTGetValueAtScooter()** function retrieves the value of pen 1 at the left scooter's current position. A change to any value within a function's argument list causes the function to be re-evaluated. Each time the position of the left scooter changes, this statement is re-evaluated.

```
MyRealTag=HTGetValueAtScooter (MyHistTrendTag,MyHistTrendTag.UpdateCount, 1,
MyHistTrendTag.ScooterPosLeft, 1, "PenValue");
```

See Also

.ScooterPosRight, .ScooterLockLeft, .ScooterLockRight

.ScooterPosRight Dotfield

The read/write .**ScooterPosRight** dotfield dynamically controls the position of the right scooter.

Category

Historical.

Usage

```
HistTrendTag.ScooterPosRight
```

Parameter

HistTrendTag

HistTrend tag assigned the name of the trend.

Remarks

This read/write dotfield dynamically controls the position of the right scooter. You can use this dotfield in a QuickScript function to retrieve the current position of the right scooter. You can also assign a value to this dotfield to move the right scooter to another location on the trend.

This dotfield is most often used in conjunction with the **HTGetValue()** functions. These functions must specify which historical trend is being queried, as well as the current position of the trend's scooters.

Data Type

Real (read/write)

Valid Values

0.0 to 1.0; where 0.0 is the extreme left of the historical trend chart, and 1.0 is at the extreme right the historical trend chart.

Examples

The following statement specifies a new location for the right scooter. The right scooter moves to a location 34 percent of the chart's total length from the left side of the Historical Trend chart associated with the MyHistTrendTag tag.

```
MyHistTrendTag.ScooterPosRight=.34;
```

The following statement uses the **HTGetValueAtScooter()** QuickScript function to retrieve the value of pen 1 at the right scooter's new position. A change to any variable within a function's parameter list causes the function to be re-evaluated. Each time the position of the right scooter changes, this statement is re-evaluated.

```
MyRealTag=HTGetValueAtScooter(MyHistTrendTag, MyHistTrendTag.UpdateCount, 2,
MyHistTrendTag.ScooterPosRight, 1, "PenValue");
```

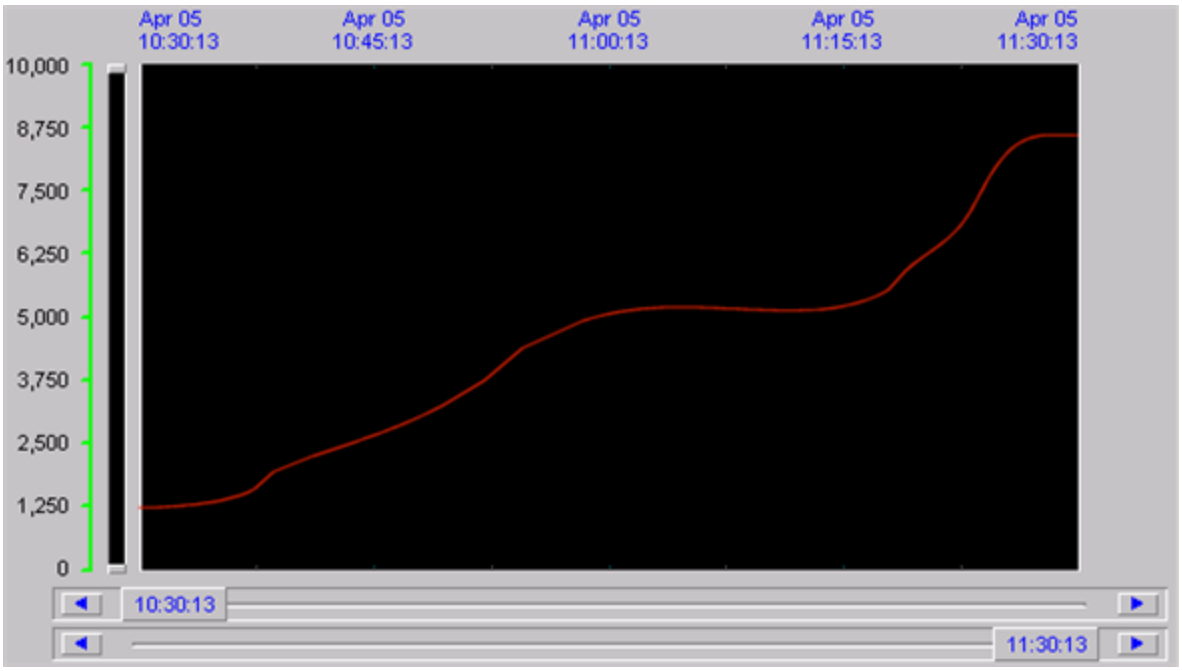
See Also

.ScooterPosLeft,. ScooterLockLeft, .ScooterLockRight

Use the historical trend wizard

The Historical Trend Wizard automatically creates a historical trend. Other than manually assigning tags to historical trend pens, the wizard automatically configures the historical trend using standard values.

The figure below shows the standard trend created with the Historical Trend Wizard. The trend includes sliders called scooters to show data at a specific location on the trend plot or zoom in on a selected range of trend data.



To add zoom and movement functions or pen controls to a historical trend, use the trend Zoom/Pan Panel and Trend Pen Legend wizards.

You can create and configure a historical trend. You can:

- Create a historical trend with wizards
- Select the tags for a trend
- Configure the historical trend time span
- Controlling a trend with QuickScripts

Create a trend with the historical trend wizard

You can create a standard historical trend using the automation features of the Historical Trend Wizard. Other sections describe how to manually configure a historical trend by using wizard options.

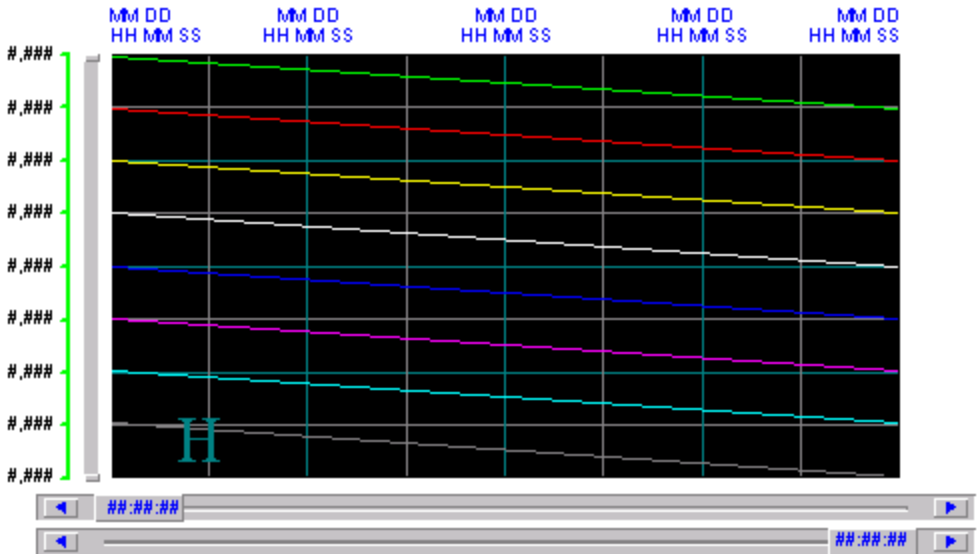
Create a historical trend with wizards

1. Open a window from WindowMaker to place a historical trend.
2. On the **Draw** menu, in the **Insert** group, select **Wizards**.

   The **Wizard Selection** dialog box appears.
3. Select **Trends** from the list of wizards.

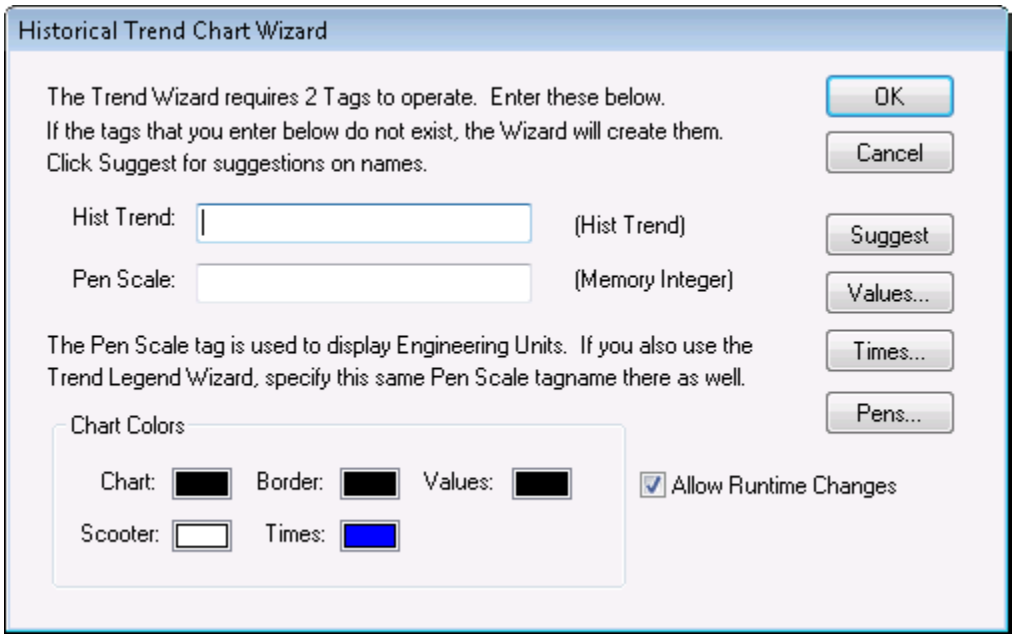
   The right pane of the **Wizard Selection** dialog box shows a set of trend wizard icons.
4. Select the **Hist Trend with Scooters** wizard and select **OK**.

   The **Wizard Selection** dialog box closes and your window reappears.
5. Move the cursor to the window location where you want to place the upper left corner of the historical trend. Select to place the trend in the window.



6. Double-click the trend.

   The **Historical Trend Chart Wizard** dialog box appears.



7. Select **Suggest**. The Historical Trend Chart Wizard automatically assigns default configuration values to the trend.

   The only remaining configuration task is to assign tags to the trend pens.

Configure which tags to display on the trend graph

Assigning tags to trend pens in the Historical Trend Chart Wizard is similar to the procedure for the Historical and Real-Time tools.

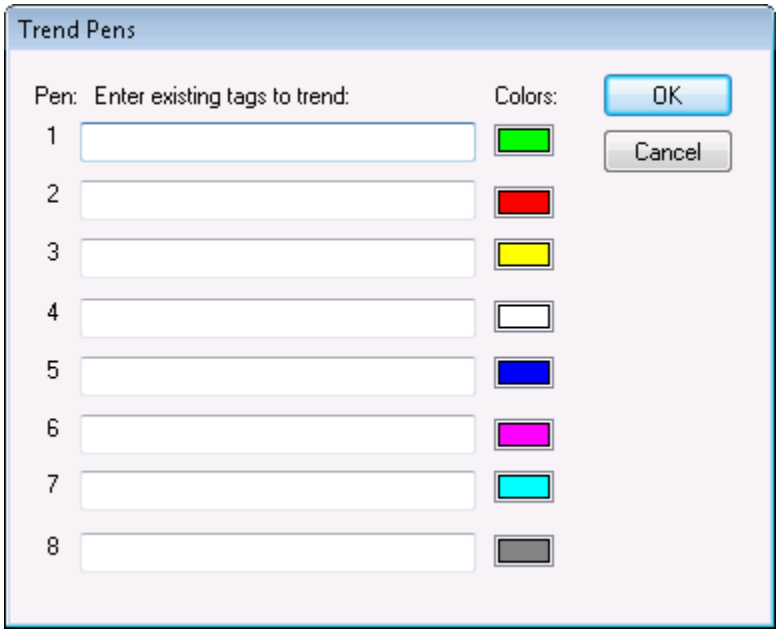
Assign tags from the Historical Trend Chart Wizard

1. Double-click the historical trend.

The **Historical Trend Chart Wizard** dialog box appears.

2. Select **Pens**.

   The **Trend Pens** dialog box appears.



3. Enter the name of an existing local tag in the **Pen** box. You can enter a maximum of 49 characters.

   Note: WindowViewer must be closed. Otherwise, the Pen boxes cannot be selected.

   If you double-click within the **Pen** box, the **Select Tag** dialog box appears with a list of tags assigned the **Log Data** option for the application. You can assign a tag to the pen by selecting it from the **Select Tag** dialog box.

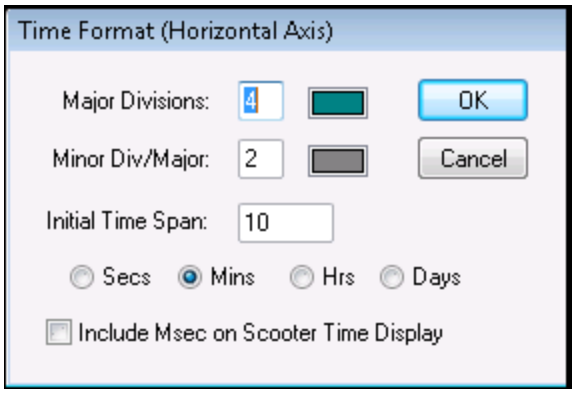
4. Select the color box next to each pen and select another color if you want to change the default pen color. Otherwise, skip this step and accept the default color.
5. Select **OK** to close the **Trend Pens** dialog box.
6. Select **OK** to close the **Historical Trend Chart Wizard** dialog box.

Configure the historical trend time span

The **Historical Trend Chart Wizard** dialog box includes an option to manually configure the time span shown from a trend created with the Historical Trend Wizard. You can manually configure a trend's time instead of accepting the default configuration of the Historical Trend Wizard.

Configure the time span of a historical trend

1. Double-click the historical trend. The **Historical Trend Chart Wizard** dialog box appears.
2. Select **Times**. The **Time Format** dialog box appears.



3. Configure the time format. Do the following:
   a. In the **Major Divisions box**, type the number of major time divisions shown on the horizontal time axis of the trend.
   b. In the **Minor Div/Major** box, type the number of minor time divisions within each major division.
   c. In the **Initial Time Span** box, type the length of the time period shown on the horizontal axis of the trend. Trends created with the Historical Trend Wizard can be updated while the application is running in WindowViewer. Operators can change the length of the trend time period. But a historical trend always starts with the time period set from the **Time Format** dialog box.
   d. Select the unit of measure for the trend time period in seconds, minutes, hours, and days.
   e. Optionally, include milliseconds in the scooter time display. The following example shows a scooter slider with milliseconds appended to the current time.

4. Select **OK** to close the **Time Format** dialog box.
5. Select **OK** to close the **Historical Trend Chart Wizard** dialog box.

Configure display options

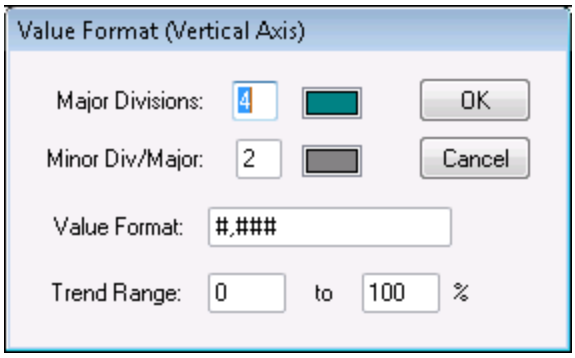
The **Historical Trend Chart Wizard** dialog box includes an option to configure the vertical units of a historical trend. You can manually configure the major and minor value divisions shown on the vertical axis of a trend.

Configure display options with the Historical Trend Chart Wizard

1. Double-click on the historical trend.

   The **Historical Trend Chart Wizard** dialog box appears.
2. Select **Values**.

   The **Value Format** dialog box appears with options to configure the vertical value axis of the trend.



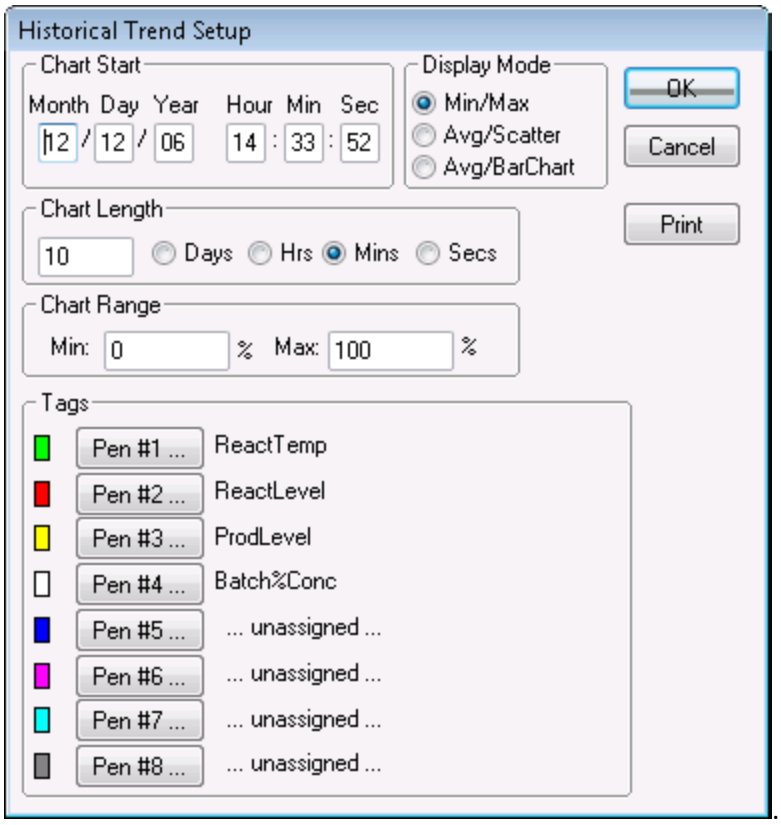
3. Configure the value format. Do the following:
   a. In the **Major Divisions** box, type the number of major value divisions shown on the trend's vertical axis. Select the color box to access the color palette, and then select the color that you assign to the major value axis division lines.
   b. In the **Minor Div/Major** box, type the number of minor divisions that you want to be visible within each major value axis division. Select the color box to access the color palette, and then select the color that you assign to the minor value axis division lines.
   c. In the **Value Format** box, type the format of numbers that appear in the trend's vertical value axis. The default number format is #,###.
   d. In the **Trend Range** boxes, type the lower and upper percentage boundaries of a tag's engineering units that appear in the trend.
4. Select **OK** to close the **Value Format** dialog box.
5. Select **OK** to close the **Historical Trend Chart Wizard** dialog box.

Change the configuration at run time

If you select the **Allow runtime changes** option when you configure your historical trend, operators can change some aspects of a historical trend during run time.

Configure a historical trend during run time

1. Select the trend in WindowViewer. The **Historical Trend Setup** dialog box appears



2. In the **Chart Start** area, type the starting date and time of the chart.
3. In the **Display Mode** area, select the type of historical trend.
4. In the **Chart Length** area, type the length of time to show on the trend, and then select the time increment for the length.
5. In the **Chart Range** area, type the percentage of engineering unit scale shown as the vertical range of the trend.
6. In the **Tags** area, select each **Pen#** to assign a tag to the trend pen. The **Select Tag** dialog box appears and shows those tags for which logging is enabled.
7. Double-click on the name of a tag to assign it to the trend pen.
8. Select **OK** to save your run-time changes to the trend.

Control a historical trend wizard using scripts

You can use QuickScript functions with trend objects or animation link expressions to control a historical trend during run time. For example, you can use QuickScripts to update a trend to the current time, reassign tags to trend pens, connect pens to the chart, replot the grid, and remove or replot the scooters.

Update the trend to the current time

You can create a script to update a historical trend to show recent tag data.

HTUpdateToCurrentTime() Function

The HT**UpdateToCurrentTime()** function retrieves and shows the data with an end time equal to the current time. The start time is equal to end time minus the width of the chart.

Category

Historical

Syntax

```
HTUpdateToCurrentTime(Hist_Tag);
```

Argument

Hist_Tag

HistTrend tag assigned the name of the historical trend.

Example

The following statement retrieves and shows data for the **Trend1** historical tag at the current time:

```
HTUpdateToCurrentTime("Trend1");
```

If the current time is 3:04 PM and the width of the trend is 60 seconds, the new end time is 3:04 PM. The new trend start time is 3:03 PM.

Change the trend configuration

You can use these script functions to change the tags assigned to the pens of a historical trend:

- HTSelectTag() Function
- HTSetPenName() Function

HTSelectTag() Function

The **HTSelectTag()** function opens the **Select Tag** dialog box for the operator to assign a different tag to a trend pen.

Note: The **Select Tag** dialog box only lists the tags that are defined for historical logging with the **Log Data** option selected in the Tagname Dictionary.

Category

Historical

Syntax

```
HTSelectTag();
```

Remarks

The HTSelectTag() function only shows tags in which the **Log Data** option has been selected from the Tagname Dictionary. However, it is possible to use the Tag Browser's filter to display a smaller set of tags. For example all tags that begin with "A". The function returns the selected tag and can be used as function parameter to assign a tag to a pen.

Example

The following QuickScript causes the **Select Tag** dialog box to appear in WindowViewer. The user can then select a tag from the list. This tag is assigned to pen 1 by the Historical Object named HistTrend.

```
HTSetPenName("HistTrend",1,HTSelectTag());
```

See Also

HTSetPenName()

HTSetPenName() Function

The **HTSetPenName()** function assigns a different tag to a trend's pen.

Category

Historical

Syntax

```
HTSetPenName(Hist_Tag,PenNum,Tagname);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

PenNum

Integer tag or value representing the pen number (1-8) of the trend.

Tagname

Name of the new tag to assign to the pen.

Remarks

This QuickScript function is the only method to add tags from a distributed history provider during run time.

You can enter a maximum of 49 characters for a reference in a pen name.

You may see the following error message when you attempt to unassign a trend pen:

```
VIEW /UpdateData: Invalid DBS.TAGNAME handle – 0
```

This error occurs if you're trying to unassign a pen that was previously assigned to a remote tag in the form *histprovider*.*tag_name*. To resolve this error, create a local tag with the **Log Data** option selected. Then, use the following script to unassign the pen:

```
HTSetPenName( "HistTrend", 1, "localtag" );
{assigns the pen to a locally logged tag---localtag}
HistTrend.Pen1=None;
{unassigns the pen}
```

Where None is a TagID type tag.

Examples

The following statement assigns the OutletPressure tag to pen 3 of Trend1.

```
HTSetPenName("Trend1",3,"OutletPressure");
```

The following statement assigns the **HistPrv1.Tag1** tag to TrendPen4 of Trend1.

```
HTSetPenName("Trend1",TrendPen4,"HistPrv1.Tag1");
```

See Also

HTSelectTag()

Retrieve information about the trend and historical data

You can create scripts that retrieve information from a historical trend while it is running. Use the following functions:

- HTGetPenName() Function
- HTGetTimeAtScooter() Function
- HTGetTimeStringAtScooter() Function
- HTGetValue() Function
- HTGetValueAtScooter() Function
- HTGetValueAtZone() Function
- HTScrollLeft() Function
- HTScrollRight() Function
- HTZoomIn() Function
- HTZoomOut() Function

HTGetPenName() Function

The **HTGetPenName()** function returns the name of the tag currently assigned to the specified pen number of the historical trend.

Category

Historical

Syntax

```
MessageResult=HTGetPenName(Hist_Tag,UpdateCount, PenNum);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

UpdateCount

Integer representing the trend's **.UpdateCount** dotfield. The argument value acts as a data change trigger to re-evaluate the function

PenNum

Integer tag or value representing the pen number (1-8) of the trend.

Example

The following statement retrieves the name of the tag assigned to Pen 2 of the Trend1 trend and places the name in the **TrendPen** message tag:

```
TrendPen=HTGetPenName("Trend1", Trend1.UpdateCount,2);
```

HTGetTimeAtScooter() Function

The **HTGetTimeAtScooter()** returns the time in seconds after 00:00:00 hours GMT, January 1, 1970 for the sample at the scooter location specified by the *ScootNum* and *ScootLoc* arguments.

Category

Historical

Syntax

```
IntegerResult=HTGetTimeAtScooter(Hist_Tag, UpdateCount,ScootNum,ScootLoc);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

UpdateCount

Integer representing the trend's **.UpdateCount** dotfield.

ScootNum

Integer representing the left or right scooter:
1=Left Scooter
2=Right Scooter

ScootLoc

Real number representing the value at the .ScooterPosRight or .ScooterPosLeft positions on the trend.

Remarks

Any changes to the values assigned to the UpdateCount, ScootNum, and ScootLoc arguments cause the expression to be evaluated. This ensures the expression is evaluated after new data retrievals or after a scooter is moved.

Example

The following statement retrieves the time in seconds for the value at the current left scooter location of the Trend1 trend:

```
HTGetTimeAtScooter("Trend1",Trend1.UpdateCount,1, Trend1.ScooterPosLeft);
```

HTGetTimeStringAtScooter() Function

The **HTGetTimeStringAtScooter()** function returns the string containing the time/date for the sample at the specified scooter location.

Category

Historical

Syntax

```
MessageResult=HTGetTimeStringAtScooter(Hist_Tag, UpdateCount, ScootNum, ScootLoc,
Format_Text);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

UpdateCount

Integer representing the trend's **.UpdateCount** dotfield.

ScootNum

Integer representing the left or right scooter:
1=Left Scooter
2=Right Scooter

ScootLoc

Real number representing the value at the .ScooterPosRight or .ScooterPosLeft positions on the trend.

Format_Text

String specifying the time/date format to use. The following Format_Text strings are acceptable:
"Date", "Time", "DateTime", "DOWShort" (Wed, for example), and "DOWLong" (Wednesday, for example).

Remarks

Any changes to the values assigned to the UpdateCount, ScootNum, and ScootLoc arguments cause the expression to be evaluated. This ensures the expression is evaluated after new data retrievals or after a scooter is moved. The format of the string determines the contents of the return value.

Example

The following statement retrieves the date and time for the value at the current scooter location for the right scooter of the Trend1 trend. The value is stored in the NewRightTimeString message tag and is in "Time" format:

```
NewRightTimeString=HTGetTimeStringAtScooter ("Trend1",Trend1.UpdateCount,2,
Trend1.ScooterPosRight,"Time");
```

HTGetValue() Function

The **HTGetValue()** function returns a value of the requested type for the trend's specified pen.

Category

Historical

Syntax

```
RealResult=HTGetValue(Hist_Tag,UpdateCount, PenNum,ValType_Text);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

UpdateCount

Integer representing the trend's **.UpdateCount** dotfield.

PenNum

Integer tag or value representing the pen number (1-8) of the trend.

ValType_Text

String indicating the type of value to return:

PenAverageValue = Average for the entire trend.

PenMaxValue = Maximum pen value for the entire trend.

PenMinValue = Minimum pen value for the entire trend.

PenMaxEU = Maximum engineering units value for the entire trend.

PenMinEU = Minimum engineering units value for the entire trend.

PenStdDev = Standard deviation for the entire trend.

Remarks

The function returns the requested value as a real value.

Example

The following statement obtains the standard deviation for the pen 2 data retrieved from the PumpPress trend. The value is stored in the LeftHemisphereSD memory real tag:

```
LeftHemisphereSD=HTGetValue("PumpPress", PumpPress.UpdateCount,2,"PenStdDev");
```

HTGetValueAtScooter() Function

The HTGetValueAtScooter() function returns a value of the requested type for the sample at the specified scooter position, trend, and pen number. The UpdateCount argument causes the expression to be evaluated after function processing is finished.

Category

Historical

Syntax

```
RealResult=HTGetValueAtScooter(Hist_Tag, UpdateCount,ScootNum,ScootLoc,PenNum,
ValType_Text);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

UpdateCount

Integer representing the trend's **.UpdateCount** dotfield.

ScootNum

Integer representing the left or right scooter:
1 = Left Scooter

2 = Right Scooter

ScootLoc

Real number representing the trend's .ScooterPosRight or .ScooterPosLeft dotfields.

PenNum

Integer tag or value representing the pen number (1-8).

ValType_Text

String indicating the type of value to return:
PenValue = Value at scooter position.
PenValid = 0 if value is invalid, 1 if valid.

When the ValType_Text argument is used with the HTGetValueAtScooter() function, use one of the valid types listed.

Example

The following function returns a 1 if the value is an actual sample or a 0 if the value is invalid for pen 3 of the Trend1 trend for the right scooter's current position:

```
HTGetValueAtScooter("Trend1",Trend1.UpdateCount, 2,Trend1.ScooterPosRight,3, "PenValid");
```

HTGetValueAtZone() Function

The HTGetValueAtZone() function returns a value of the requested type for the data located between the right and left scooter positions for a trend's specified pen.

Category

Historical

Syntax

```
RealResult=HTGetValueAtZone(Hist_Tag,UpdateCount,
Scoot1Loc,Scoot2Loc,PenNum,ValType_Text);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

UpdateCount

Integer representing the trend's **.UpdateCount** dotfield. It is used only as a trigger to evaluate the function.

Scoot1Loc

Real representing the trend's .ScooterPosLeft dotfield. It is used only as a trigger to evaluate the function.

Scoot2Loc

Real representing the trend's .ScooterPosRight dotfield. It is used only as a trigger to evaluate the function.

PenNum

Integer tag or value representing the pen number (1-8) of the trend.

ValType_Text

String indicating the type of value to return.
PenAverageValue = Average for zone between the scooters.
PenMaxValue = Maximum value for the zone between the scooters.
PenMinValue = Minimum value for the zone between the scooters.
PenMaxEU = Maximum engineering unit value for the zone between scooters.
PenMinEU = Minimum engineering unit value for the zone between the scooters.
PenStdDev = Standard Deviation for the zone between the scooters.

Remarks

A real value is returned representing the calculated value of the given type. Specifying constant values for the Scoot1Loc and Scoot2Loc arguments has no effect and are only used to trigger the evaluation of the function. The function uses the trend tag's .ScooterPosLeft and .ScooterPosRight dotfield values directly, regardless of the values you specify for the Scoot1Loc and Scoot2Loc arguments.

Example

The following statement calculates the average value for data between the right and left scooters of the Trend1 trend for pen 1. The value is stored in the AvgValue memory real tag:

```
AvgValue=HTGetValueAtZone("Trend1", Trend1.UpdateCount,Trend1.ScooterPosLeft,
Trend1.ScooterPosRight,1,"PenAverageValue");
```

Pan and zoom the trend

You can create QuickScripts containing functions that select specific data from a historical trend during run time.

HTScrollLeft() Function

The **HTScrollLeft()** function sets the start time of the trend to an earlier time than the current start time by a percentage of the trend's total time span. The effect is to scroll the chart to the left to an earlier time by a specified percentage of the trend’s total time span.

Category

Historical

Syntax

```
HTScrollLeft(Hist_Tag,Percent);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

Percent

Real number representing the percentage of the chart's time span to scroll (0.0 to 100.0) left.

Example

The following statement scrolls the time/date left by 10 percent of the PumpPress trend's total width:

```
HTScrollLeft("PumpPress",10.0);
```

If the current display starts at 12:00:00 PM and the display width is 60 seconds, then the new trend starts at 11:59:54 AM after the function is processed.

HTScrollRight() Function

The **HTScrollRight()** function sets the start time of the trend to a time later than the current start time by a percentage of the trend's width. The effect is to scroll the date/time of chart to the right by a specified percentage of the trend's width.

Category

Historical

Syntax

```
HTScrollRight(Hist_Tag,Percent);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

Percent

Real number representing the percentage of the chart to scroll (0.0 to 100.0) right.

Example

The following statement scrolls the PumpPress trend to the right by 20 percent.

```
HTScrollRight("PumpPress",20.0);
```

If the current display starts at 12:00:00 PM and the display width is 60 seconds, then the new trend starts at 12:00:12 PM after the function is processed.

HTZoomIn() Function

The HTZoomIn() function calculates a new chart width and start time. If the trend's scooters are at the left and right sides of the trend, then the new chart width equals the old chart width divided by two. The new start time is calculated based on the value of the *LockString* argument.

If the scooters are not at the left and right sides of the trend, the HTZoomIn() function zooms the trend to the zone defined by the scooters and ignores the *LockString* argument.

Category

Historical

Syntax

```
HTZoomIn(Hist_Tag,LockString);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

LockString

String representing the type of zoom:

StartTime	Keep the start time equal to before zoom
Center	Keep center time equal to before zoom
EndTime	Keep end time equal to before zoom

Remarks

If the scooter positions are not at the left and right sides of the trend, the new chart width is the time between .ScooterPosLeft and .ScooterPosRight positions. In this case, the value of LockString is not used. The minimum chart width is one second. The scooter positions are set to .ScooterPosLeft=0.0 and .ScooterPosRight=1.0 after the zoom.

Example

The following statement zooms the display by a factor of two and maintains the same start time for the Trend1 trend. Trend1.ScooterPosRight is equal to 1.0 and Trend1.ScooterPosLeft is equal to 0.0. If the start time before zooming was 1:25:00 PM and the chart width was 30 seconds, the new start time remains at 1:25:00. The new chart width is 15 seconds.

```
HTZoomIn("Trend1","StartTime");
```

HTZoomOut() Function

The **HTZoomOut()** function calculates a new chart width and start time. The new chart width is the old chart width multiplied by two. The new start time is calculated based on the value of the *LockString* argument .

Category

Historical

Syntax

```
HTZoomOut(Hist_Tag,LockString);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

LockString

String representing the type of zoom:
StartTime = Keep start time equal to before zoom
Center = Keep center time equal to before zoom
EndTime = Keep end time equal to before zoom

Remarks

The current scooter positions have no effect on HTZoomOut(). After the function zoom finishes, the new scooter positions are set to .ScooterPosLeft=0.0 and .ScooterPosRight=1.0.

Example

The following statement zooms out the trend time by a factor of two and maintains the same center time for the Volume trend. If the start time before zooming was 2:15:00 PM and the chart width was 30 seconds, the start time after zooming is now 2:14:45. The chart width is 60 seconds and the center of the trend remains at 2:15:15.

```
HTZoomOut("Volume","Center");
```

Print the trend

You can print a historical trend currently visible from a WindowViewer screen using the PrintHT() function in a script.

PrintHT() Function

The **PrintHT()** function prints the historical trend currently visible on the screen. Usually, the **PrintHT()** function is associated with a screen button included on the historical trend window. Operators select the button to print the visible historical trend with its current values.

Category.

Historical

Syntax

```
PrintHT(Trend_Tag);
```

Argument

Trend_Tag

HistTrend tag.

Example

This example prints the PumpPress historical trend currently visible on the screen.

```
PrintHT(PumpPress);
```

Troubleshoot the trend

You can create QuickScripts to verify if data was successfully retrieved that appears in a historical trend. Use the HTGetLastError() function to troubleshoot the trend.

HTGetLastError() Function

The **HTGetLastError()** function can be used in a script to determine if an error occurred during the last data retrieval for a specified historical trend pen.

Category

Historical

Syntax

```
[Result=]HTGetLastError(Hist_Tag,UpdateCount, PenNum);
```

Arguments

Hist_Tag

HistTrend tag assigned the name of the trend.

UpdateCount

Integer representing the trend's **.UpdateCount** dotfield.

PenNum

Integer tag or value representing the pen number (1-8) of the trend.

Result

Integer assigned to a tag that represents the status of the last script function call for the specified pen.

0 = No error

1 = General server error

2 = Old request

3 = File error

4 = Server not loaded

5 = Trend/Pen passed in function does not exist

6 = Trend tag does not exist in database

7 = Pen number passed to function is invalid (not in range of 1 to 8).

8 = No tag or a non-logged tag assigned to the pen number

Examples

The following statement retrieves the status of the last data retrieval for pen 3 of the Trend1 trend and assigns the result to the ResultCode integer tag.

```
[ResultCode=]HTGetLastError("Trend1", Trend1.UpdateCount,3);
```

In an animation Analog Value Display QuickScript the following statement would be used:

```
HTGetLastError("Trend1",Trend1.UpdateCount,3);
```

Display real-time values in a trend

You can create real-time trends by two methods. The Real-Time Trend object provides a standard set of controls to select the data, set a time range, and specify the physical appearance of the graph. InTouch also includes the 16-Pen Trend Wizard, which is an optional control to create real-time and historical trends. For more information about creating real-time trends with the 16-Pen Trend Wizard, see Create a 16-Pen Trend in *AVEVA™ InTouch HMI Management*.

Use real-time trend objects

You can create a real-time trend to show current values in your application.

Create a real-time trend

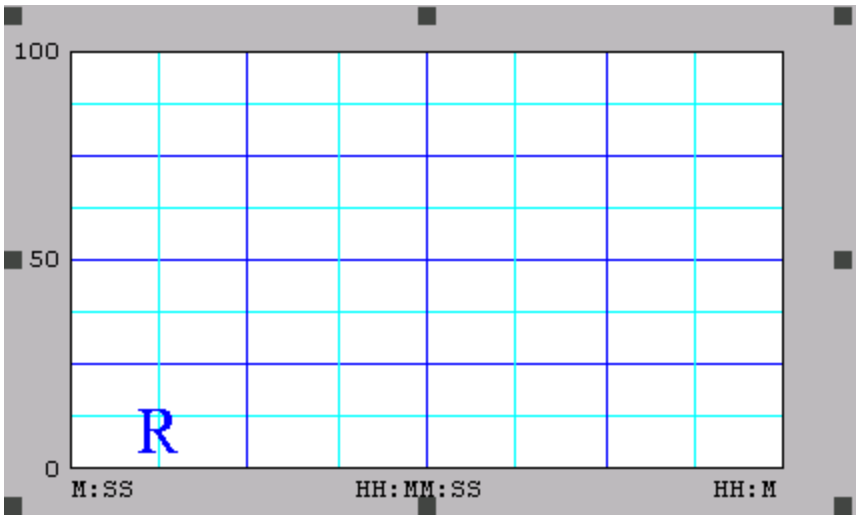
You can use the Real-time Trend tool to create a trend object in a window. The first time you paste a real-time trend object, WindowMaker uses default settings. After configuring a real-time trend, WindowMaker uses the last configuration values as the initial settings for any new real-time trend object.

You can draw the trend chart any size within the borders of the window.

Create a real-time trend

1. On the **Draw** menu, in the **Trends** group, select **Real-time**.
2. Move the mouse over the window area where you want to place the real-time trend. Drag the mouse diagonally to create a rectangle of the desired size of the trend.

   The Real-time Trend object appears in the window.



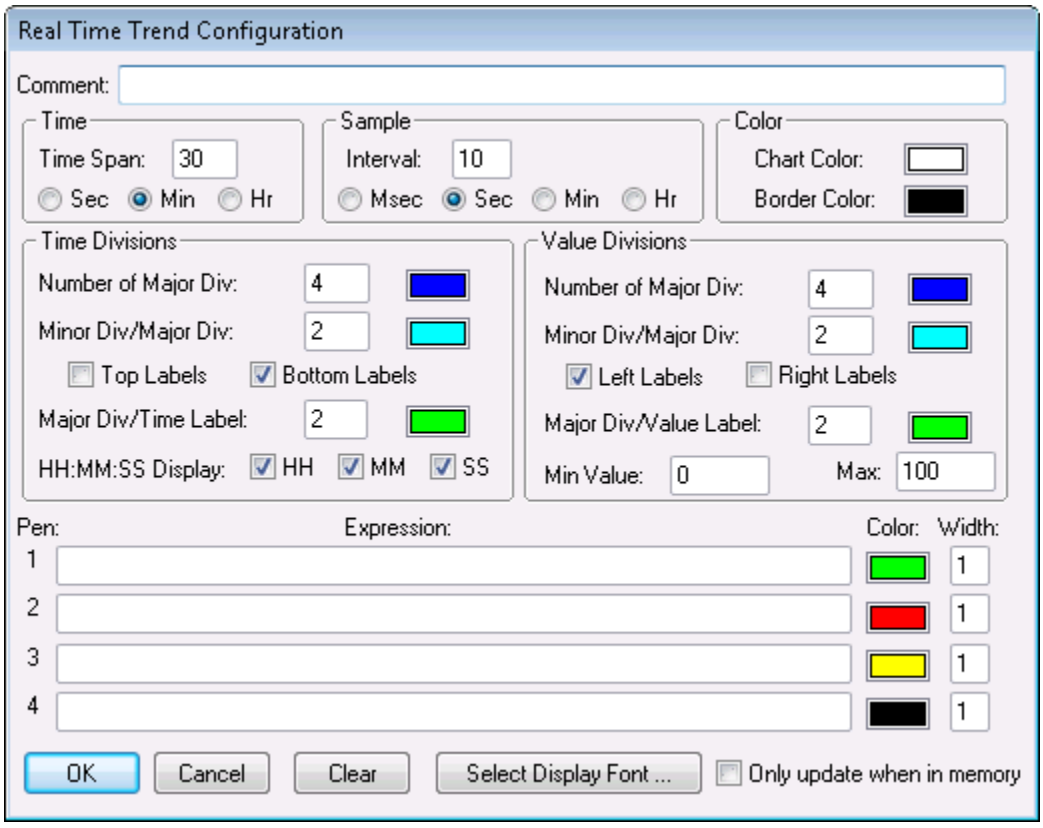
3. If needed, adjust the height and width of the trend with its object handles.

Configure which tags to display on a real-time trend

A real-time trend pen creates a graphical representation of current data from any local tag or an expression that contains one or more local tags. You configure the pens that show tag data in a real-time trend.

Configure real-time trend tags

1. Double-click the trend object in the window. The **Real Time Trend Configuration** dialog box appears.



2. In the **Expression** area, type the name of a local tag or expression that contains one or more local tags.

If you double-click in the **Pen** box, the **Select Tag** dialog box appears showing a list of tags defined for the application. You can assign a tag to the pen by selecting it from the **Select Tag** dialog box.

3. Select the color box next to each pen assigned a tag to show a color palette.
4. Select the color that you want to assign to the pen.
5. In the **Width** box, type the line width in pixels for each pen shown in the trend.

   Selecting a line width greater than 1 increases the time required to update or print a trend.
6. Select the **Only update when in memory** checkbox if you want the trend to update only when shown in the active window.

   If you do not select this option, the trend updates continuously even if the window is closed. Continuous trend updating slows system performance.
7. Keep the **Real Time Trend Configuration** dialog box open and go to the next procedure described in Configure the real-time trend time span and update rate.

Configure the real-time trend time span and update rate

You can configure the time span and update rate of a real-time trend.

Set a real-time trend time span and update rate

1. Double-click the trend object. The **Real Time Trend Configuration** dialog box appears.
2. In the **Time** area, type the length of time in the **Time Span** box that you want to appear on the horizontal x-axis of the trend.
3. Select the unit of measure for trend time.
   - Seconds (Sec)
   - Minutes (Min)
   - Hours (Hr)

     For example, if you enter 30 in the **Time Span** box and then select **Min**, the time span shown on the chart is 30 minutes.
4. In the **Sample** area, type a number in the **Interval** box that the trend expression is evaluated and the chart updates.
5. Select the interval unit of measure.
   - Milliseconds (Msec)
   - Seconds (Sec)
   - Minutes (Min)
   - Hours (Hr)

     For example, if you enter 10 in **Interval** and then select **Sec,** the real-time trend is updated every 10 seconds.
6. Keep the **Real Time Trend Configuration** dialog box open and go to the next procedure described in Configure real-time trend display options.

Configure real-time trend display options

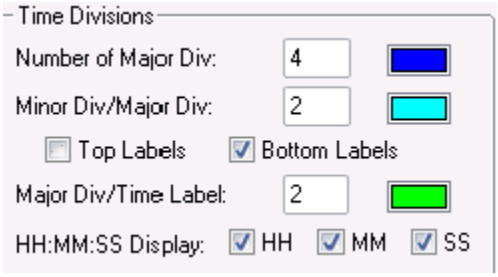
You can configure the visual appearance of a real-time trend.

Configure real-time trend display options

1. Double-click on the trend object. The **Real Time Trend Configuration** dialog box appears.
2. In the **Color** area, configure the color. Do any of the following:
   - Select the **Chart Color** box to open the color palette. Select the background color for the trend.

     White is the default background color. Any other background color significantly increases the time needed to print a trend.
   - Select the **Border Color** box to open the color palette. Select the border color of the trend.
3. In the **Time Divisions** area, configure the time divisions. Do the following:
   - In the **Number of Major Div box,** type the number of major trend time divisions The major time divisions appear on the horizontal time axis of the trend.

     The number of major time divisions must be an even multiple of the **Major Div/Time Label** value. For example, a division number of 20 is an even multiple of the **Major Div/Time Label** value of 4.

   - Select the color for the major division lines.
   - In the **Minor Div/Major Div** box, type the number of minor time divisions visible within each major time division.

     The number of minor time divisions should be evenly divisible within the major division period. For example, if the major division period is set to 60 seconds, entering a value of 2 in **Minor Div/Major Div** creates two minor time division periods of 30 seconds.
   - Select the color for the minor division lines.
   - Select either the **Top Labels** or **Bottom Labels** checkbox to specify the placement of time labels on the trend.

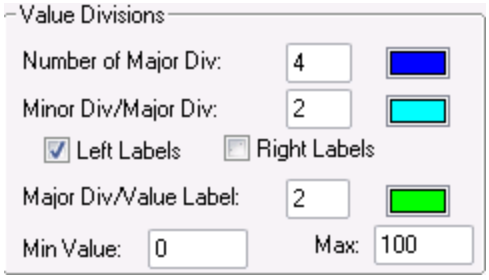
     You can select both options to place time labels at the top and bottom of the trend. Leaving both options blank removes time labels from the horizontal axis of the trend.
   - If you are using time labels, type the number of major time division lines per time label in **Major Div/ Time Label**. The number of major divisions must be an even multiple of the **Major Div/Time Label** value.

     Select the color of the time division labels.
   - Select the time units shown as part of the major time division label.

     Hours (HH)

     Minutes (MM)

     Seconds (SS)
4. In the **Value Divisions** area, configure the appearance of the vertical axis of the trend.



Value Divisions options are configured like **Time Divisions** options. The major and minor divisions on the y-axis show the magnitude of data values rather than time. The vertical axis specifies the range of data values that appear in the trend based upon engineering units for all tags.

To show decimal points for the minor and major value divisions, type real numbers for the **Min Value** and **Max** options. For example, 0.00 to 100.00.

5. Choose **Select Display Font**. The **Font** dialog box appears with options to set the font, style, and size of text that appears in the trend.
6. Select **OK**.

Print a trend at run time

Several factors determine how fast a trend can be printed. The primary factor is the size of the trend on the printed page. The display mode of the trend also affects printing performance. Min/Max or Average/Scatter trends can be printed more quickly than Average/Bar Chart trends. Also, the longer and wider the lines on the trend are, the longer it takes to print.

Another factor that affects printing performance is the background color of the trend. In most cases, a white background prints more quickly than a colored background.

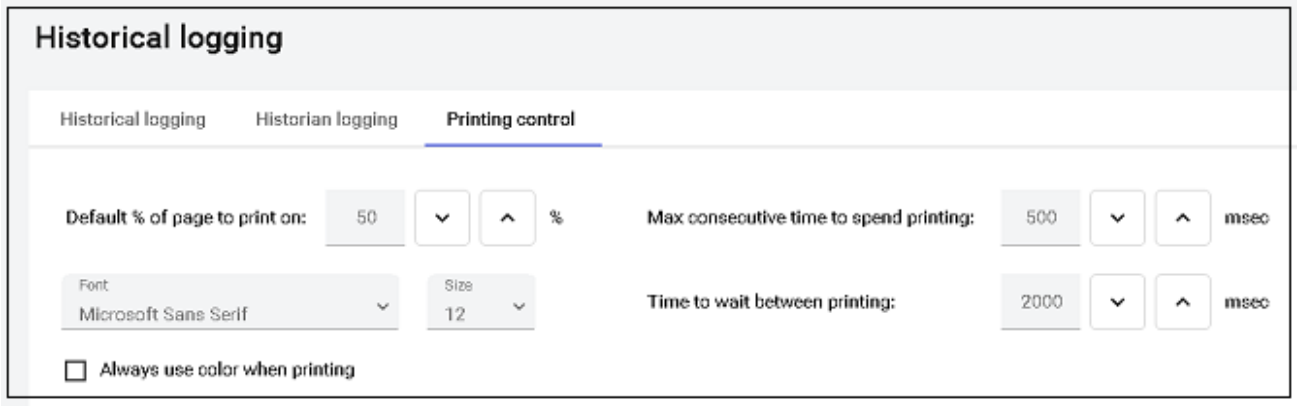
Configure trend printing options

You can configure options that determine how a trend is printed.

Configure historical trend printing

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **Historical Logging**.

   The **Historical Logging** configuration screen appears.
3. Select the **Printing control** tab.



4. Specify the percentage of the page to print the trend in the **Default % of page to print on** box.

   If you enter 50, the trend is printed on half of the page vertically and horizontally. A trend printed at 50 percent takes much less time to print than a full-page trend.

   As a printing alternative, you can use the **PrintWindow()** QuickScript function.
5. In the **Max consecutive time to spend printing** box, type the process time slice in milliseconds.

   A time slice represents the period allocated to the computer processor to run the print module process in the foreground and print the trend. A longer time slice enables the trend to be printed more quickly at the expense of other processes running on the computer.
6. In the **Time to wait between printing** box, type the time in milliseconds the print module waits between processor time slices.

   A shorter waiting period between processor time slices enables the trend to be printed more quickly.
7. From the **Font** dropdown list, select the **Printer Font** characteristics appearing in a trend.
8. Select the checkbox **Always use color when printing** for a colored print.
9. Select **Save**.

Display historical tag values from other InTouch nodes or Historian Server

If you want to use data stored remotely to create historical trends, the remote provider must be registered in the InTouch history provider list. This list specifies the name and network location of each history provider. These names are referred to whenever historical trend pens point to tags at the remote history provider.

You can define a remote history provider and assign historical trend pens to tag data stored at the remote location. You can:

- Configure remote history providers.
- Configure pens to display data from a remote history provider.
- Assign pens to tag data stored at a remote history provider using the Tag Browser.
- Assign a pen to a remote tag using a QuickScript.

For more information about using data from a remote history provider, see Distributing Applications in *AVEVA™ InTouch HMI Application Deployment*.

Use the InTouch HMI with the Historian Server

The Historian Server is a real-time, relational database designed specifically for industrial applications. You can optionally configure the Historical Logger to store InTouch historical data to a historian in the AVEVA history server database.

Note: For more information about setting up the InTouch HMI with a remote history provider, see Distributing Applications in *AVEVA™ InTouch HMI Application Deployment*.

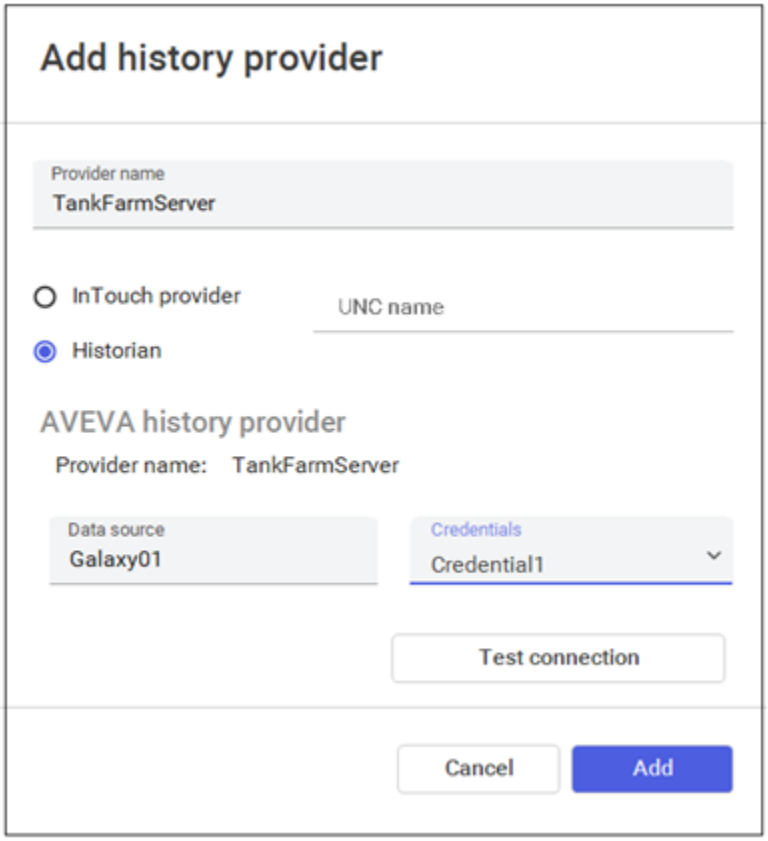
If you use the the Historian to store historical data, you must use the Distributed Name Manager from WindowMaker to specify a connection to the database.

Configure a connection to a Historian Server database

1. Open WindowMaker.
2. On the File menu, highlight **Configure**, and then select **Distributed Name Manager**.

   The Distributed Name Manager configuration screen appears, with separate tabs for the **Distributed Alarms** and the **Distributed History**.
3. On the **Distributed History** tab, select the **+** icon to add a history provider, or press Alt+A.

   The **Add history provider** dialog appears.



4. In the **Provider Name** box, type the name you want to use for the new historical provider.

   A provider name can be 16 alphanumeric characters or fewer.
5. Select **Historian**.
   a. In the **Data Source** box, type the node name of the server where the Historian Server is installed.

b. From the **Credentials** drop-down, select the credentials for authentication.

Note: For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credential Manager in AVEVA™ InTouch HMI Application Development.

a. Select **Test connection** to verify the connection to the Historian Server database. A message appears indicating whether the connection to the database is successful or not.

6. Select **Add** to close the dialog box. The Historian Server node appears in the History Providers list.
7. Select **Save**.

Configure pens to display remote trend data

Historical trends can display tag data from both local and remote history providers. You can assign trend pens to display data from a remote history provider.

Display a tag from a remote history provider

1. Double-click on the historical trend to show the **Historical Trend Configuration** dialog box.
2. In each pen's **Tagname** box, type the reference to a remote history provider. The format of the reference to a remote history provider is:

   history_provider_name.tag_name

   Example:

   ```
   TankFarm1.Pump1RPM
   ```

   Each pen of a historical trend can refer to a different remote history provider.
3. Select **OK** to save your configuration changes.

Note: The .TagID dotfield cannot be used in remote history provider tag references.

Use the tag browser to assign pens to remote history providers

The following procedure explains how to use the Tag Browser to assign a trend pen to tag data from a Remote History provider. Using the Tag Browser to select tags eliminates the need to manually enter each tag name and reduces the likelihood of errors.

The remote node name you specify in the Access Name does not have to be the actual name of the node where the tag resides. But, you must create an Access Name to define the remote history provider as a tag source. For more information about creating an Access Name, see Set up access names.

Define a remote history provider as a tag source

1. Create an Access Name that specifies the node name where the history provider is located.
2. Double-click the historical trend to open the **Historical Trend Configuration** dialog box.
3. Double-click a pen's **Tagname** input box to show the **Select Tag** dialog box.
4. Select **Define Tag Sources** to define the remote history provider as a tag source.
5. Select the **Tag Source** arrow and select the new remote history provider tag source in the list, or select the **Tree View** button and select the tag source in the tree view pane. The **Select Tag** dialog box refreshes and shows the tags from the selected remote history provider.

6. Select the tag that you want to assign to the historical pen and select **OK**. The **Historical Trend Configuration** dialog box reappears with the selected tag listed in the pen's **Tagname** box as: *AccessName*:*Item*.
7. Replace the AccessName: portion with the history provider name you defined in the **Distributed Name Manager**.

   For example, HistPrv1.*tag_name*

   This process may seem cumbersome, but after you have defined the history provider as a tag source in the Tag Browser, each time you double-click another tag input box, you simply double-click the tag name in the **Tag Browser**, and then replace the AccessName: portion with the history provider name.

In WindowViewer, if run-time changes are allowed for the historical trend, when the user selects a pen button to change the tagname, the Tag Browser appears but only the local application's tags are accessible.

Use a QuickScript to assign a pen to a remote history provider

While an InTouch application is running, you can configure a trend pen to show tag data from a remote history provider. Create a QuickScript that specifies the remote history provider tag reference in the HTSetPenName() function. For example:

```
HTSetPenName("HistTrendTag", 1, "HistPrv1.Boiler1");
```

In this example, the number 1 specifies the number of the pen in the historical trend that plots the remote **Boiler1** tag values from the HistPrv1 remote history provider.

The run-time **Historical Trend Setup** dialog box and **Pen** dotfields are not supported for remote history providers.

User Defined Types

A User Defined Type (UDT) is a hierarchical data structure that consists of individual user-defined members. Each member can have its own data type as Integer, Boolean, string, other common data type, or another UDT.

UDTs can reduce engineering effort by providing an option to create multi-level nested structures. For example, you can create a reactor UDT in InTouch, instead of creating a set of individual tags representing that reactor.

About UDTs

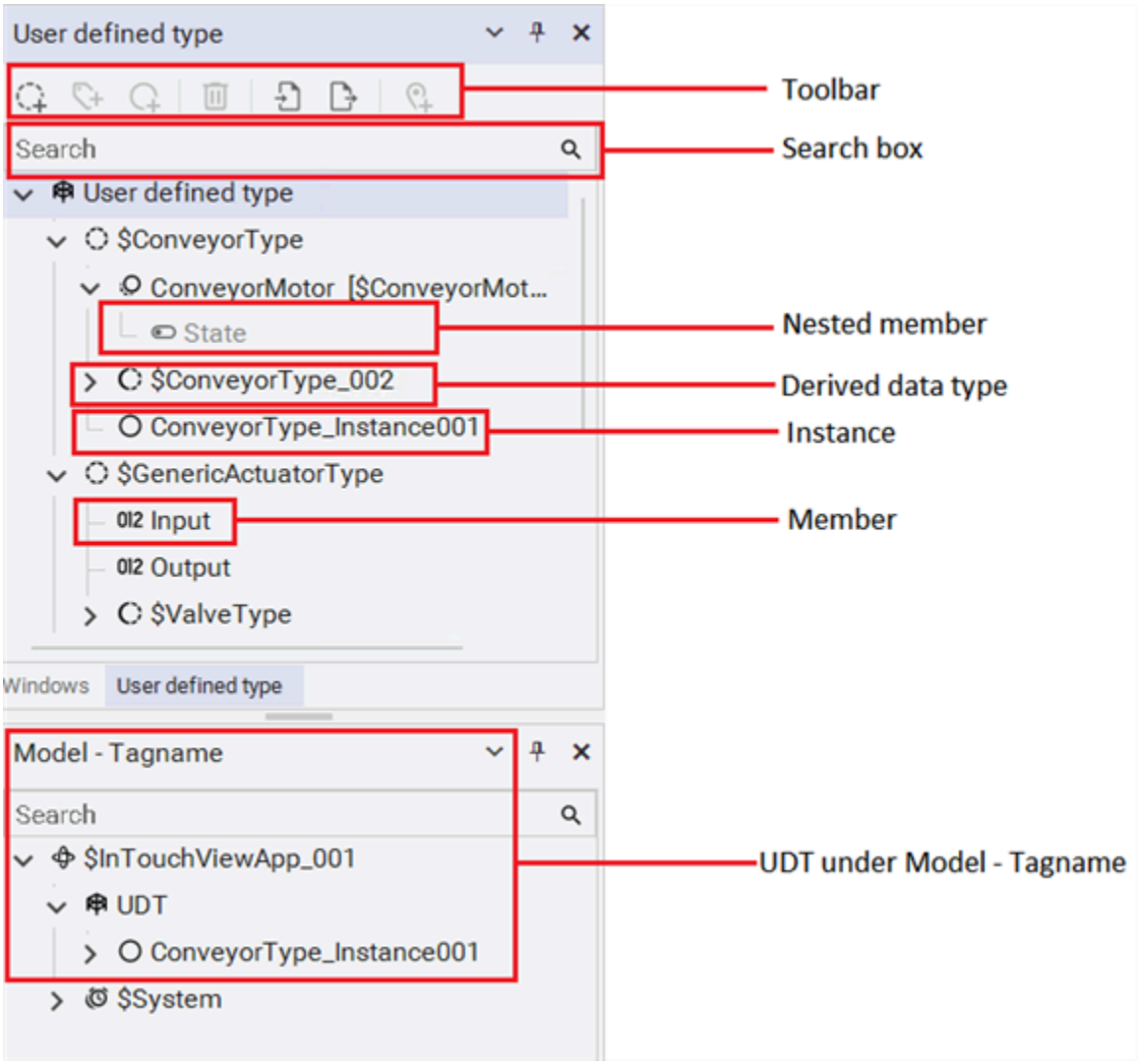
InTouch HMI provides support for UDTs. You can see the **User defined type** pane to the left side of the WindowMaker canvas. It includes the following:

- **Search box** - search for the required data type, derived data type, instance, or a member.
- **Toolbar** - the toolbar includes the following options:
  - **Add new data type**
  - **Add new member**
  - **Add new instance**
  - **Delete**
  - **Import data types**
  - **Export data types**
  - **Add location members**

The User Defined Type hierarchy can include these items in the following order:

1. **Direct member** - Members that are defined within the current user defined type.
2. **Inherited members** - Members that are inherited from other data types. These are shown as grayed-out.
3. **Derived data type** - A data type that is derived from a base data type (base data types are read-only and are predefined in InTouch). It includes all the members that are under the base data type from which it was derived. You can add additional members to the derived data type, apart from the existing items. The base data type and derived data type are prefixed with $ sign by default. If you rename a derived type, it still must have $ as the first character of its name.
4. **Instance** - An instance represents a physical or virtual object of the user defined type. You must create instance to be able to use the user defined type in graphic animations or scripting. An instance tag member is an InTouch basic tag.

Note: You can edit UDTs in the **Properties** grid. You cannot view or edit UDTs in the Tagname dictionary.



UDT specification

User Defined Types (UDTs) must comply with the following specifications:

Name length

- Maximum name length for data type, derived data type, and instance name is 32 characters.
- Maximum name length for the member data type and member is 32 characters.

Valid names

- Instance names can include alphanumeric characters, and the special characters $ (dollar sign), # (pound sign), and _ (underscore).
- The $ character cannot be used as the first character.

Maximum nested level

- A maximum of six nested UDT levels is allowed.

  For example, "Datatype1.Datatype2.Datatype3.Datatype4.Datatype5.Datatype6.Member1"

  where:
  - "Datatype1.Datatype2.Datatype3.Datatype4.Datatype5.Datatype6" are six UDT nested levels.
  - "Member1" is a basic InTouch tag type, such as memory integer.

  Note: Circular referencing is not allowed.

Create User Defined Types manually

You can create and manage User Defined Types in WindowMaker with the **User defined type** pane. You can create individual UDTs and build a hierarchical structure of UDTs by using a set of discrete operations. These operations are described in the following topics:

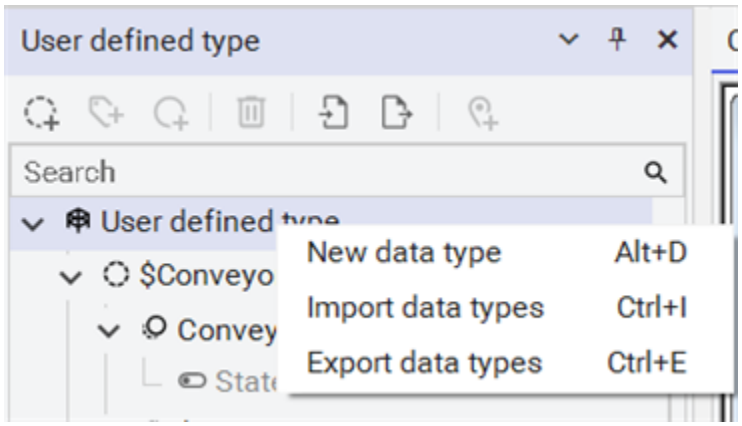
- Create a new data type
- Create a new member
- Create a new derived data type
- Create a new instance
- Nest a member under another data type
- Build a set of User Defined Types by importing data types
- Export data types
- Manage User Defined Types
- Edit User Defined Types in bulk

Note: The UDT names, member names, and property names shown throughout this section are for illustration only. You should name your types and members as appropriate for your application.

Create a new data type

Create a new data type

1. In the **User defined type** pane, right-click **User defined type** and select **New data type** from the context menu, or select the **New data type** icon on the toolbar.



   OR

   Select New data type  on the toolbar.

   OR

   Use the shortcut key **Alt+D**.

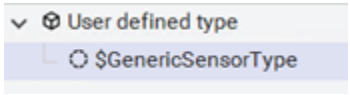
   The data type is created.

   The default data type name is $Data type_00*n,* where the initial value of *n* = 1. The data type name is prefixed with a $ sign. You cannot delete the $ sign. The data type name can use alphanumeric characters, plus $, #, and _ (underscore) characters. The maximum name length is 32 characters.

2. To rename the data type, right-click the data type and select **Rename**.

   Example:

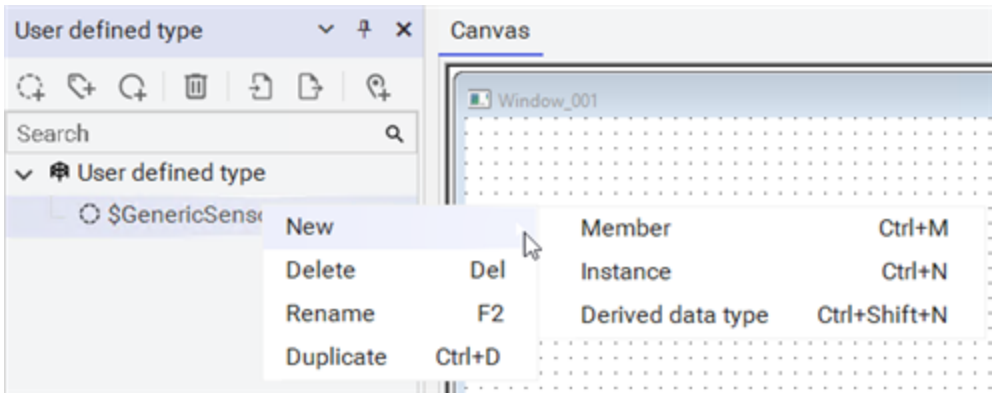
   Rename the new data type as GenericSensorType.



Create a new member

Create a member

1. In the **User defined type** pane, right-click the data type, select **New** from the context menu, and then select **Member**, or select the **New member** icon on the toolbar.



OR

Select New member  on the toolbar.

OR

Use the shortcut key **Ctrl+M**.

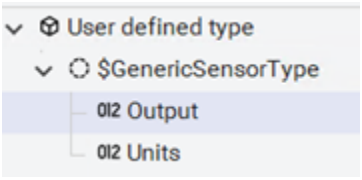
A new member, **Member001**, is created. The member name can use alphanumeric characters, plus $, #, and _ (underscore) characters. The maximum member name length is 32 characters.

2. To rename the member, right-click the member and select **Rename**.

Example:

You can add two members of basic InTouch tag types, for example:

- Output - type: integer
- Units - type: message (string)



You can also set the tag properties in the **Property** grid on the right pane.

Example:

Set the “Initial value” of the Output member tag to 50.

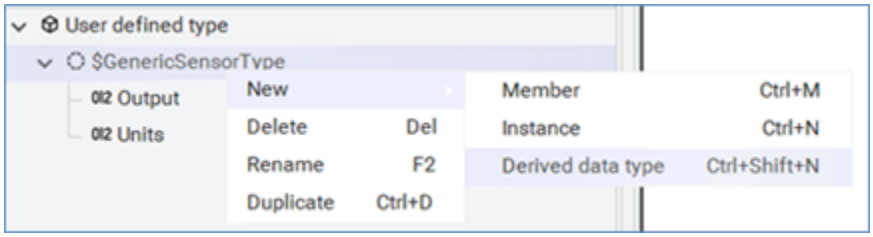
For more information on properties, see Edit User Defined Types in the Properties grid.

Create a new derived data type

You can derive a data type and inherit all of its members.

Create a derived data type

1. In the **User defined type** pane, right-click the required data type, select **New** from the context menu, and then select **Derived data type**.



   OR

   Use the shortcut key **Ctrl+Shift+N**.

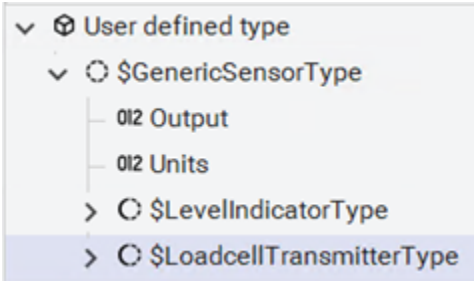
   A new derived data type is created.

   The default name of a derived data type is <Parent data type name>_00*n*, where the initial value of *n* = 1. For example, if you create a derived data type from $Pump, the default name of the first derived data type is “$Pump_001”. The derived data type name is prefixed with a $ sign and the $ sign cannot be deleted. The derived data type name can use alphanumeric characters, plus $, #, and _ (underscore) characters. The maximum name length is 32 characters.

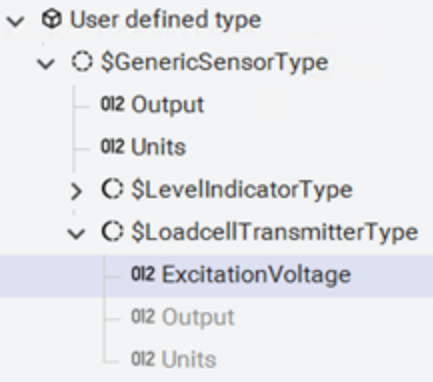
   The members in the base data type are added to the derived data type. You can modify the properties of the members of the parent data type except Name and Type property. You can add new members to a derived data type. The inherited members are grayed out. You can modify properties of the derived data type in the **Properties** pane.

Example:

Create two derived data type from GenericSensorType and call them LevelIndicatorType and LoadcellTransmitterType.



Next, add a new member to the LoadcellTransmitterType data type and call it ExcitationVoltage of type integer.

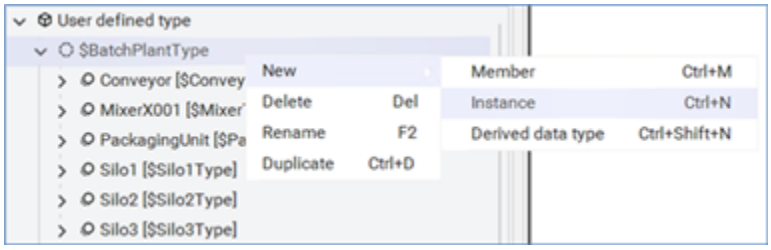


The derived data type inherits all data members of the parent and are shown as grayed-out.

Create a new instance

Create an instance

1. In the **User defined type** pane, right-click the required data type, select **New** from the context menu, and then select **Instance**. Or, you can select the **New instance** icon on the toolbar.



   OR

   Select New instance  on the toolbar.

   OR

   Use the shortcut key **Ctrl+N**.

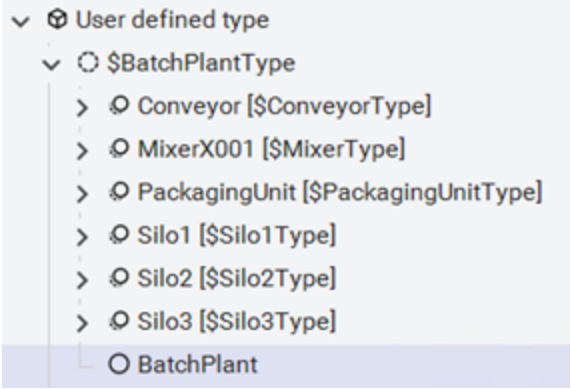
   A new instance is created.

2. Rename the UDT. The default instance name is <data type name>_00*n*, where the initial value of *n* = 1. Instance names are not prefixed with the $ character. The instance name can use alphanumeric characters, plus $, #, and _ (underscore) characters. The maximum name length is 32 characters.

The members in a base data type are added to the instance. You cannot add new members to an instance. The inherited members will be in Grey color. You can modify properties of the instance in the **Properties** pane. You can modify the properties of the members except Name and Type properties. All the members under a data type or derived data type will belong to instances under the same data type or derived data type. You can confirm what members belong to which instance from the **Model – Tagname** pane. All the instances along with their members are listed under **UDT** of the **Model – Tagname** pane.

Example:

Create an instance from $BatchPlantType and call it BatchPlant.

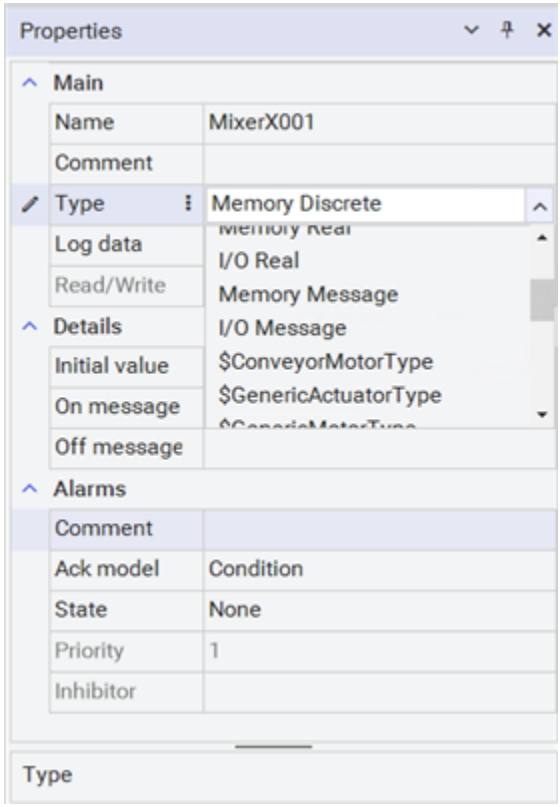


Nest a member under another data type

UDT nesting is accomplished by creating a member of another data type. You can point a member of one data type to another data type in the **Properties** pane. Nesting can reduce the time and effort required to create tags. Nesting UDTs up to six levels is supported.

Nest a member under another data type

1. Select the required member that you want to nest.
2. In the **Type** field of the **Properties** pane, select the required data type or derived data type under which you want to nest. You can also type the name of the data type in the **Type** field, to filter the data type.



3. The member is nested under the selected data type or derived data type. You can rename it. For example, rename the nested member "InletValve".

Note: This name is not another instance of a data type. It is another way to reference the data type member of the data type instance.

Example:

“InletValve” is not an actual instance. It does not appear in the root of the tagname dictionary, and it will not appear in the root of a tag hierarchy in the cross-reference utility. You can map the nested member to an instance to reference it in script and animation .

The maximum number of nested UDT levels is six. Consider the following example:

Datatype1.Datatype2.Datatype3.Datatype4.Datatype5.Datatype6.Member1

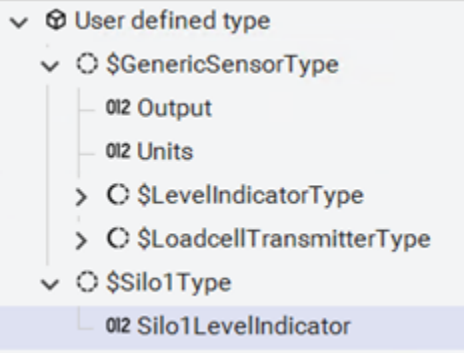
In this example:

- "Datatype1.Datatype2.Datatype3.Datatype4.Datatype5.Datatype6" are six levels of the UDT.
- "Member1” is a basic InTouch tag type such as a memory integer.

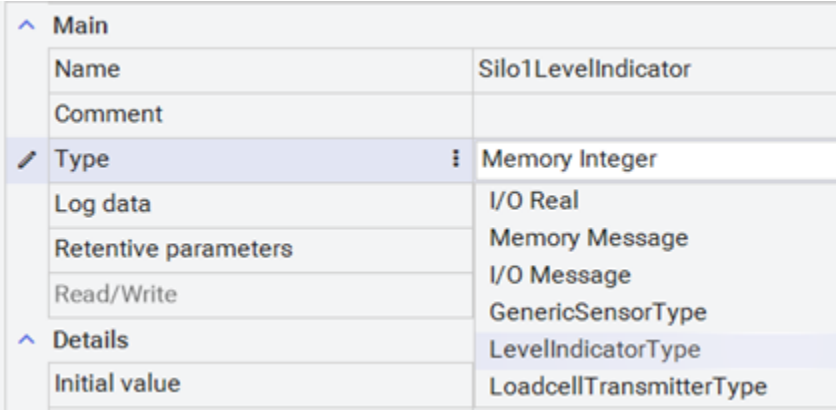
  Note: Circular referencing is not allowed.

Example:

Create a new data type and call it Silo1Type and then add a member called Silo1LevelIndicator.



Now, from the **Property** Grid, you can change the property **Type** to be another data type. Say you select LevelIndicatorType.



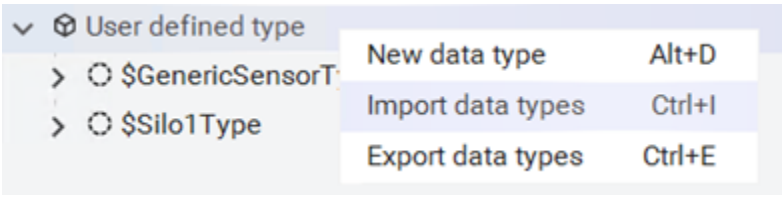
The nested member is added.

Build a set of User Defined Types by importing data types

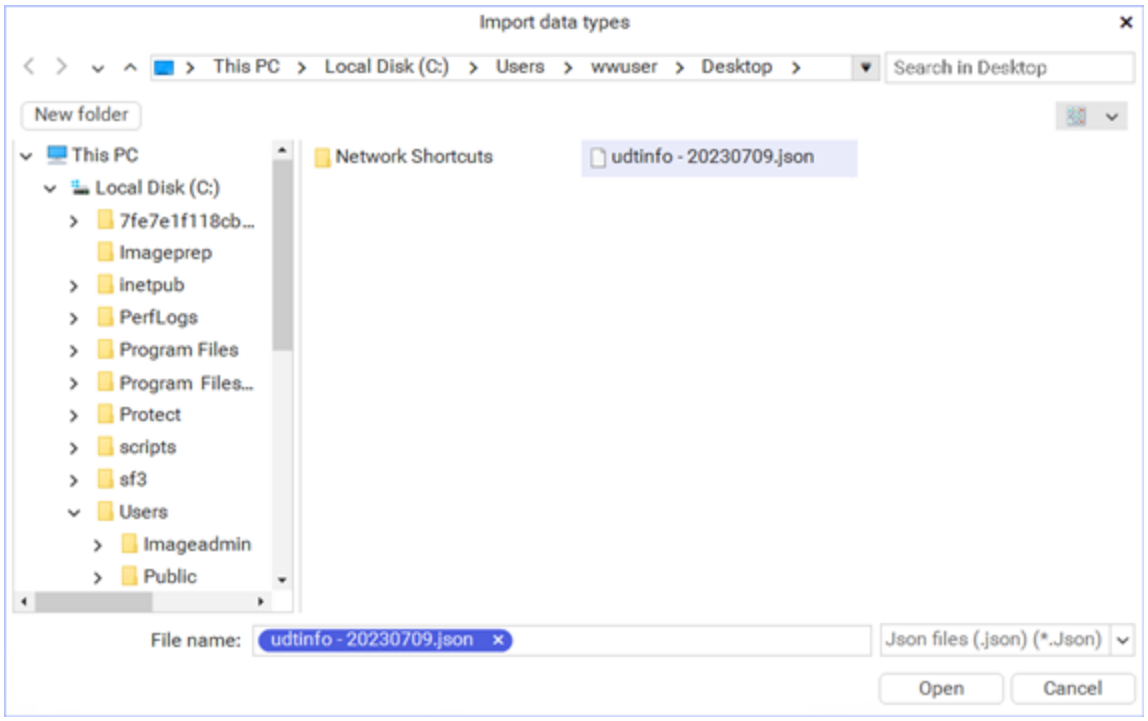
You can efficiently build a set of UDTs by importing a .json file containing a previously created set of UDTs into your application.

Import data types

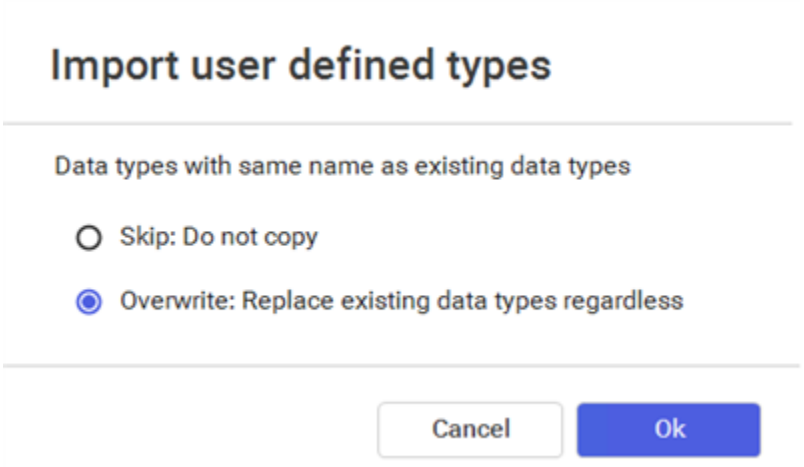
1. Open the **User defined type** pane.
2. Right-click **User defined type** and select **Import data types** from the context menu, then select the **Import** icon on the toolbar.



3. Navigate to and select the .json file with data type configuration details, and then select **Open**.



4. In the **Import user defined types** dialog box,
   - Select **Skip: Do not copy** to skip the import of data types that has same name as the existing data types.
   - Select **Overwrite: Replace existing data types regardless** to overwrite and replace the existing data types, when data type name is same as the existing data types.

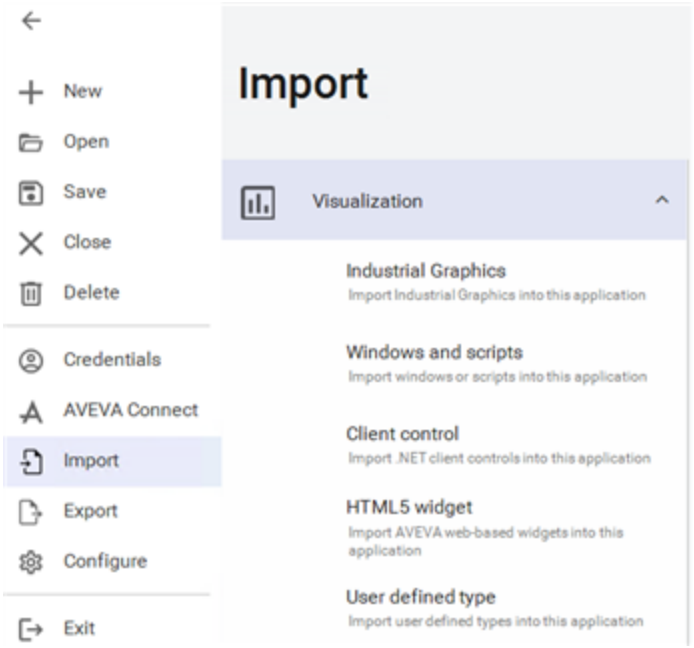


5. Select **OK**.

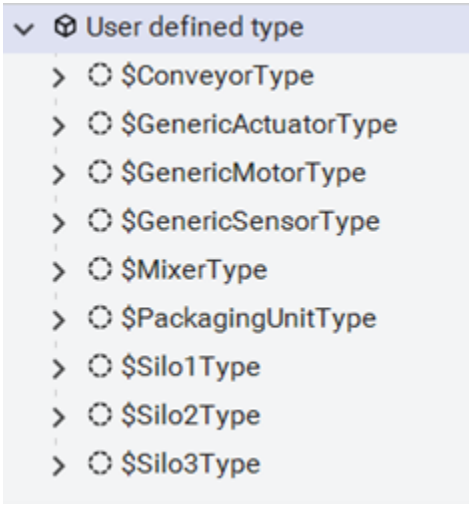
   All data types and other items available in the .json file appear according to the option that you have selected in the previous step.

   Note: Overridden instance values are not retained when you import the file.

You can also import the data type using the **Import** > **Visualization** > **User defined type** option under **File** menu.



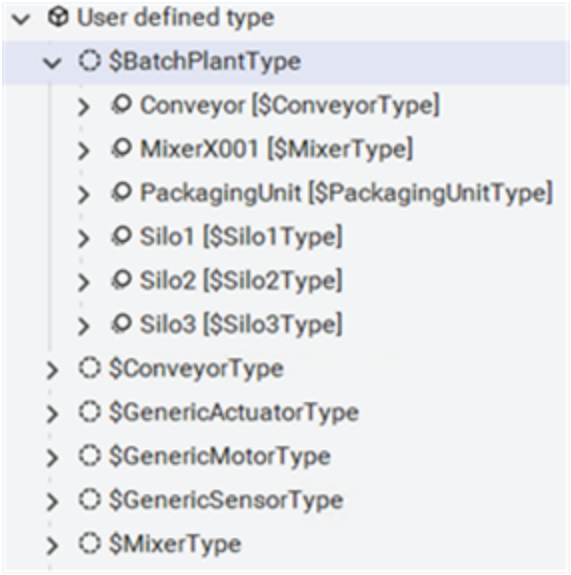
After importing the file, the UDT hierarchy is shown.



You can add more UDTs to the hierarchy. To do that, create a new data type called "BatchPlantType", add members, and change the type in the properties grid for each member as shown below:

- Conveyor - type $ConveyorType
- MixerX001 - type $MixerType
- PackagingUnit - type $PackagingUnitType
- Silo1 - type $Silo1Type
- Silo2 - type $Silo2Type
- Silo3 - type $Silo3Type

The resulting hierarchy reflects the new members.



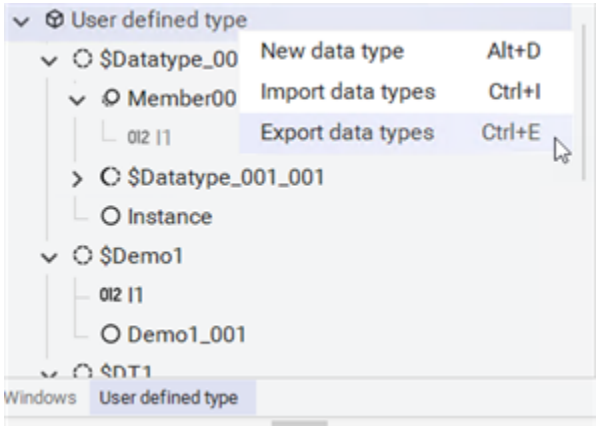
Export data types

You can export data types for use in other applications, on other computers in your system, or in other systems.

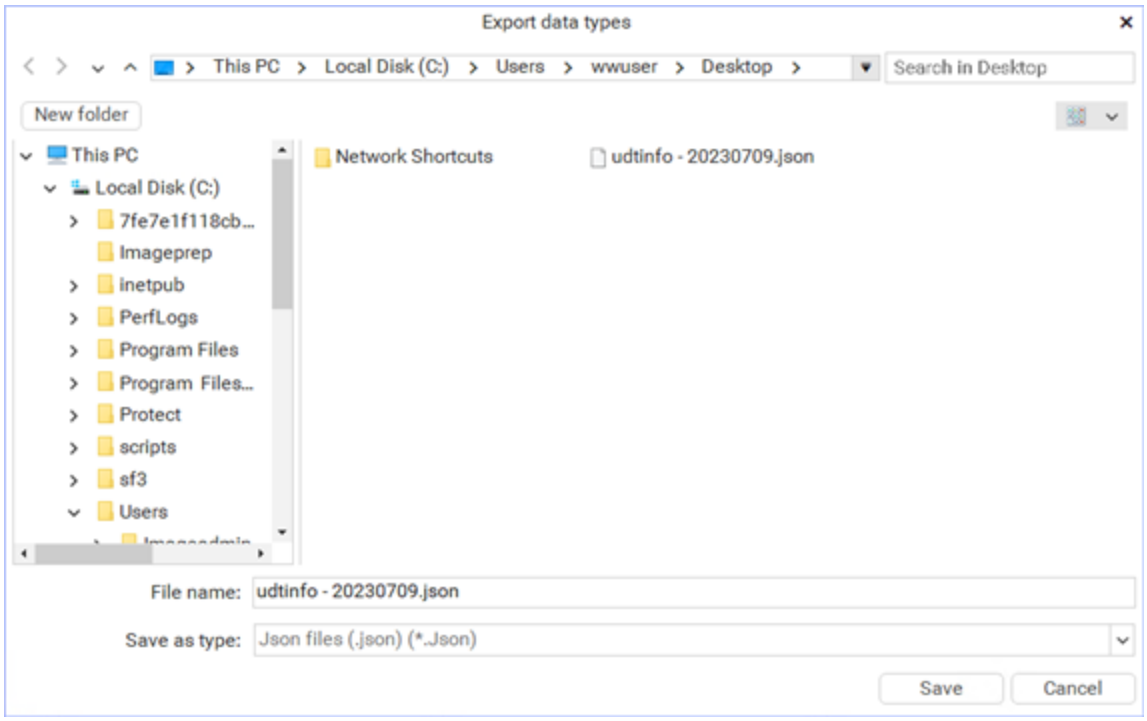
Export data types:

1. Open the **User defined type** pane.

   Right-click **User defined type** and select **Export data types** from the context menu, or select the **Export** icon on the toolbar.

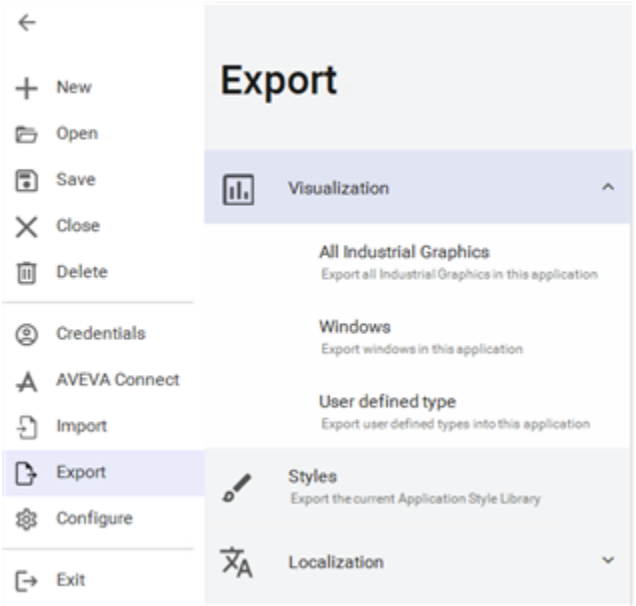
   The **Export data types** window opens.



2. Modify the file name if required, and then select **Save**.

   The .json file is saved to the chosen location. You can edit the exported file using an external json editor and import it back into your application or into another application.

You can also export the data types using the **Export** > **Visualization** > **User defined type** option under the **File** menu.



Note: During export all UDTs will be included in the JSON file.

Manage User Defined Types

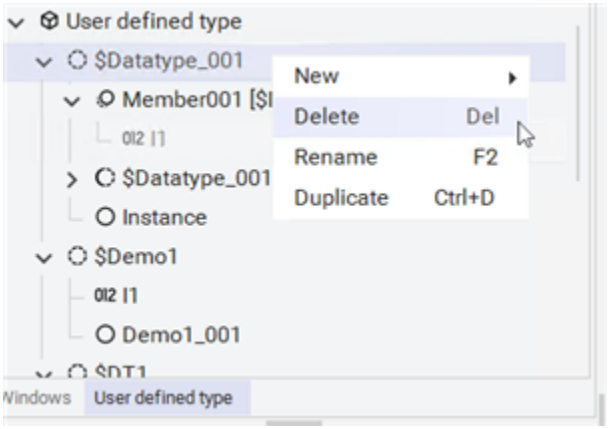
You can perform basic management operations in addition to operations to create and update UDTs described in the preceding topics.

Delete a member

- Right-click the member that you want to delete, and select **Delete** from the context menu**.** You can also select the **Delete** icon present on the toolbar to delete the member. Use the **Ctrl** key to select multiple members.

Delete a data type, a derived data type, or an instance

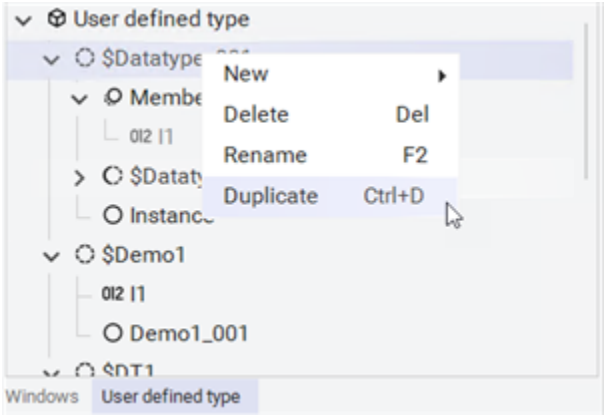
Right-click the data type, derived data type, or an instance and select **Delete** from the context menu**,** or select the **Delete** icon on the toolbar.This deletes the data type and all its members.



Note: You cannot delete a data type unless you delete the instances and derived data types under it.

Duplicate a data type

- In the **User defined type** pane, right-click the base data type and select **Duplicate** from the context menu. A copy of the base data type with all its members is created.



Note: You can only duplicate data types and derived data types. You cannot duplicate instances and members.

Add location attributes

1. Open the **User defined type** pane.
2. Highlight the data type or derived data type, and then select **Add location**  icon on the toolbar.

   New attributes for Latitude and Longitude are added to the data type or derived data type. For more information on using location attributes, see the MapApp topic later in this section.

Edit User Defined Types in bulk

You can edit UDTs in bulk using a JSON file. To edit the exported JSON file, use any a publicly available popular JSON editor to add, delete, or edit the UDTs.

Edit or create UDTs in bulk

1. Export the existing UDTs to a JSON file. For more information on how to export UDTs, see Export data types.
2. Open the JSON file in a text or JSON editor.
3. Edit or create UDTs by copying the existing UDTs and pasting back into the file.
4. Rename or edit other properties as required.
5. Save the JSON file.
6. Import the JSON file back into the User defined type pane of WindowMaker. For more information on importing, see Build a set of User Defined Types by importing data types.

Example:

View User Defined Types

You can view the User Defined Type in two ways:

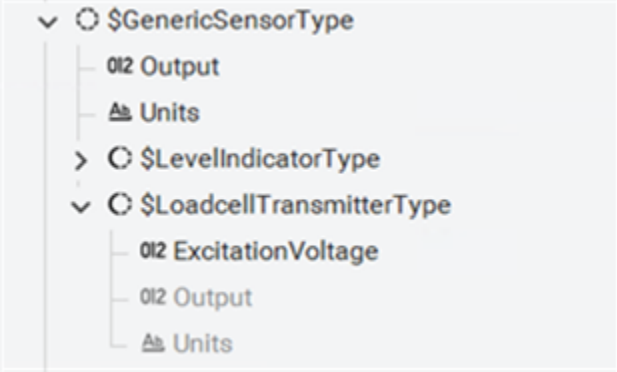
- Through the User Defined Type view
- Through the Model - tagname view

User Defined Type view

The **User defined type** pane displays the list of User Defined Types, their member tags and member data types, derived data types, and instances.

- Data type is prefixed with the “$” character.
- Member data types are appended with “[$<data type name>]”.
- All member and derived data types can be fully expanded to show the member.
- The instance name cannot be expanded in this view.
- For each data type, all child items are sorted in the order shown below. Within the group, items are sorted alphabetically.
  1. Direct members
  2. Inherited members
  3. Derived data types.
  4. Instances.

Example:



Model - tagname view

Model – Tagname view is the InTouch tag view with the addition of the UDT instance view. You can view UDT instances and their members.

Edit User Defined Types in the Properties grid

Note: You must edit UDTs in the **Properties** grid. You cannot view or edit UDTs in the Tagname dictionary.

Update the Properties

You can view and modify UDT properties from the **Properties** grid. Properties of the selected item in the User defined type pane or **Model - tagname** pane are displayed in the **Properties** pane at the right-hand side of WindowMaker.

- Highlight each property to select and update the property, or to view the corresponding property description at the bottom of the **Properties** pane.
- Any changes you make are automatically saved.
- The property fields change, based on the option that is selected in the **Type** property field.
- The **Type** property field lists UDTs along with the other data types.
- The list of properties is updated to reflect the properties associated with the selected **Type**.
- You can edit all tag property values where the member is defined except the **Name** and **Type** properties.

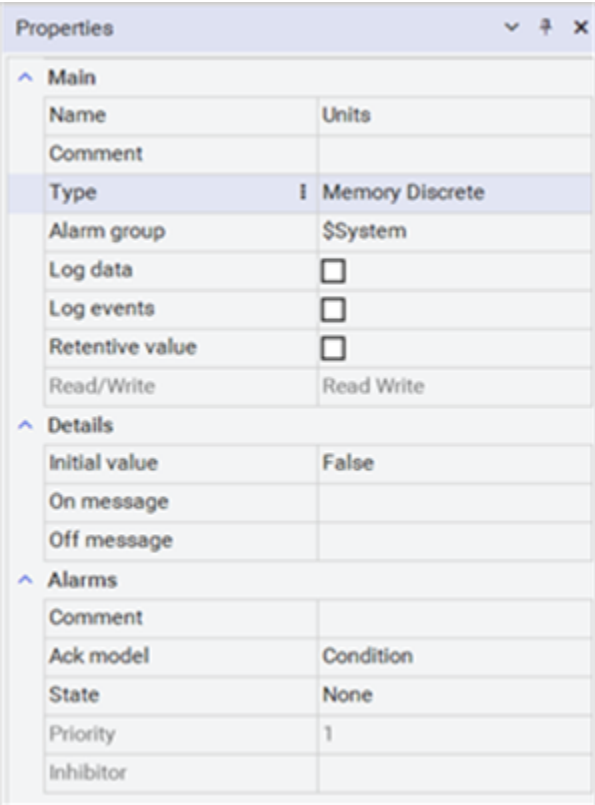
Member data type

If you change a member to be another data type, then the property will only show the following properties. In the example, you can select the $BatchPlantType > Silo1 member.

Edit member tag

If you change the member type to a basic InTouch tag, then all the relevant tag properties become editable.

For example, select a UDT named $GenericSensorType > Units and change the type to Memory Discrete. The properties grid refreshes to show all related tag properties for this discrete tag.



Invalid Values and Required Fields

Invalid values of a property are indicated by a red icon. After you update the property value, and then select any other user defined type, a red error icon is displayed in front of the property name. If you hover over the error icon, a tooltip with the reason for the error is displayed.

The properties for which values are required are indicated with a red flag. You are not forced to enter the value at data type level. But you are recommended to enter or override value at the instance level.

Example:

Value override and change propagation

You can edit all tag properties value at data type level where the member is defined except the **Name** and **Type** properties.

Example 1:

1. Select $GenericSensorType > Output.
2. Change the Initial value to be 50.
3. Then select $LevelIndicatorType > Output.
4. Notice that the Initial value is 50 as well.
5. Select $GenericSensorType > Output.
6. Change the Initial value to 60.

   Return to $LevelIndicatorType > Output. Notice that the Initial value is now updated to 60 as well. This is an example of change propagation.
7. You have the option to override the initial value. Set the initial value to be 75.

   Notice that the initial value is now in bold to indicate that it has been overridden.

8. Select $GenericSensorType > Output and change the Initial value to 65.

   In this instance, the value will not propagate down to $LevelIndicatorType because its value has been overridden.

Example 2:

This example is extended from the preceding example 1.

1. In **Model - Tagname** view, Select BatchPlant > Silo1 > Silo1LevelIndicator > Output. The Initial value is 75. This is from the $LevelIndicatorType data type
2. Enter an override value of 25..
3. Select BatchPlant > Silo2 > Silo2LevelIndicator > Output and override the initial value to 55.
4. Select BatchPlant > Silo3 > Silo3LevelIndicator > Output. Override the initial value to 80.

Change propagation

Changes in a data type are reflected immediately in the **User Defined Type** view.

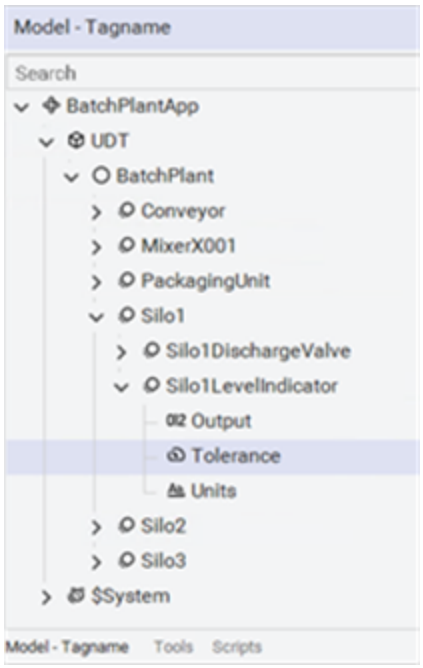
Add new member

Adding a new member is reflected in the derived data type, member data type, and instance.

Example:

1. Select $GenericSensorType.
2. Add a new member called "Tolerance" of type Memory Real.
   This new member is now available in its derived data type $LevelIndicatorType.

   This new member is also available in the instance member data type.



Change member type

Changing the existing member type is also reflected immediately in the derived data type, member data type, and instance.

Example:

Extended from the previous example:

1. Go to $GenericSensorType > Tolerance.
2. In the properties grid, change the data type from Memory Real to Memory Message.

   $LevelIndicatorType > Tolerance is also updated to be data type Memory Message.

UDT support for existing functionalities

Note: The UDT names, the member names, and the properties as shown in the examples and images are for illustration purposes only. You should name your types and members as appropriate for your application.

UDTs are supported in the following InTouch functionalities:

- Intellisense
- Animation
- Scripting
- Substitute references
- Browse for UDT members
- Drag and drop the UDT member of an instance to the Industrial Graphic Editor canvas
- Animation and scripting
- Alarm and Alarm Client Control

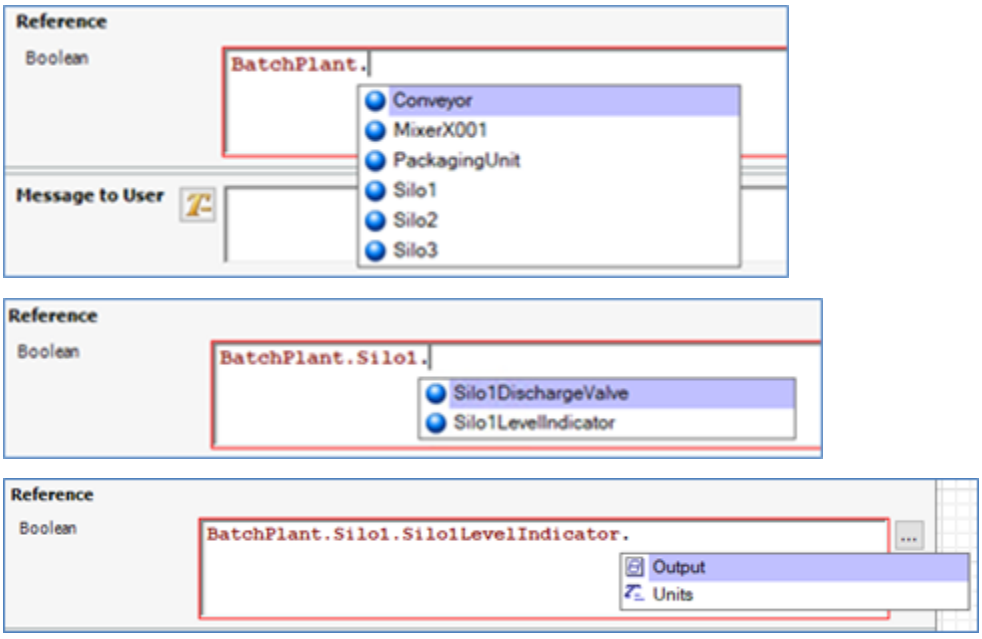
- History and trend client
- Configure a UDT member as an I/O tag
- Use a UDT instance as an owning object
- Web Client
- MapApp
- Cross-reference
- Delete unused tags
- Co-existence with Supertags
- Licensing
- Run-time tag viewer

Intellisense

Intellisense for UDT instances lists all members and member data structures for the immediate level only. The Industrial Graphic Editor supports intellisense in scripting, animation, and custom properties. Native InTouch supports intellisense in scripting only.

- Industrial Graphics: Intellisense support in scripting, animation, and custom properties.
- Native InTouch: Intellisense support in native InTouch scripting only.

Example:



Animation

You can configure UDTs in Industrial Graphic animations and native InTouch animations.

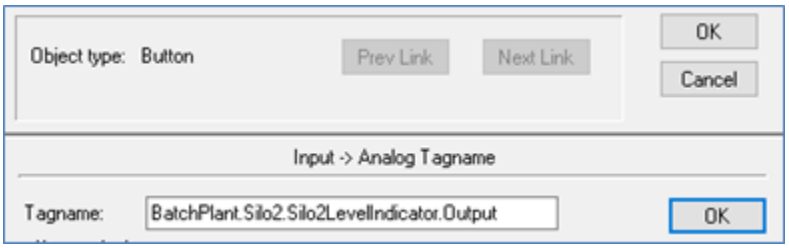
Note: Substitute tags for UDTs with value display animation behaves the same as remote tags. You need to

replace the complete reference with either the UDT tag or the UDT tag with dot field.

Example:

Configure a UDT in an Industrial Graphic animation:

Configure a UDT in a native InTouch animation:

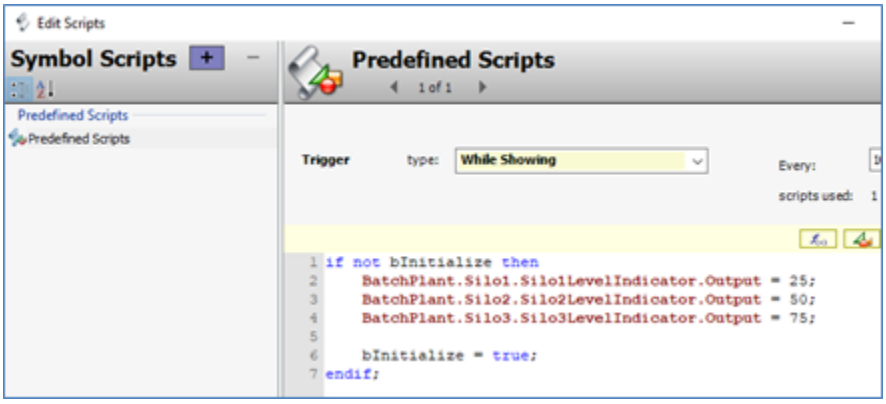


Scripting

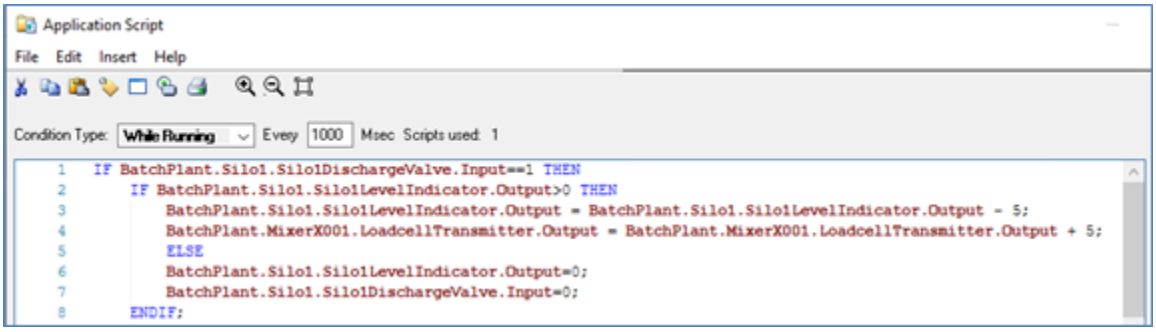
You can configure UDTs in an Industrial Graphic script and a native InTouch script.

Example:

Configure a UDT in an Industrial Graphic script:



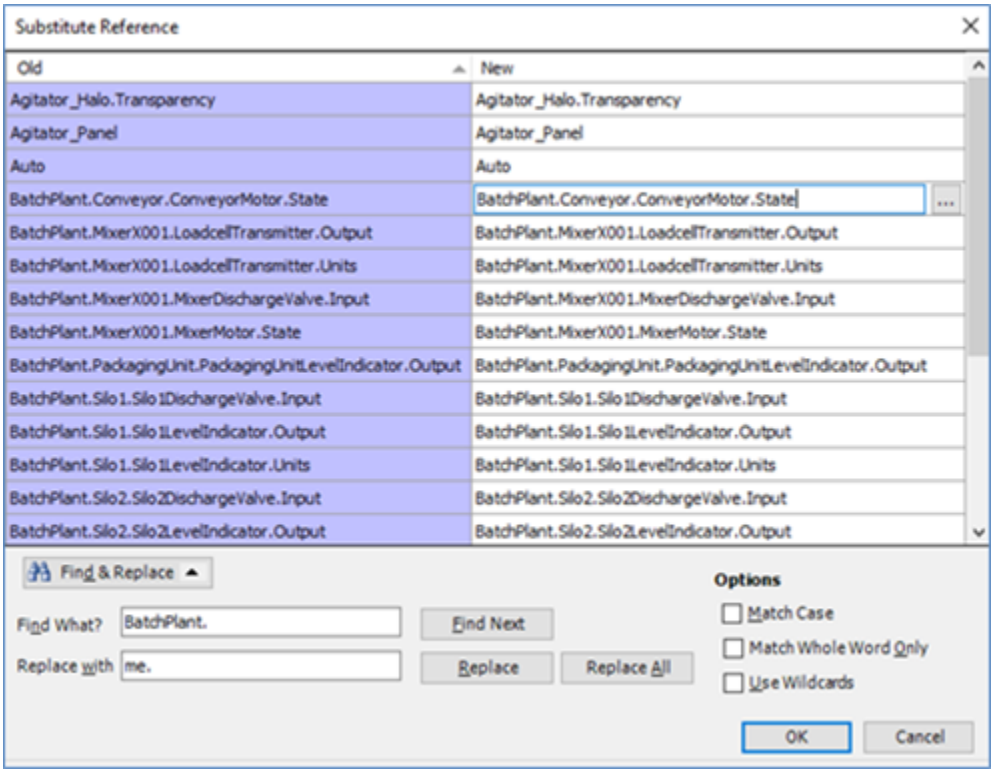
Configure a UDT in a native InTouch script:



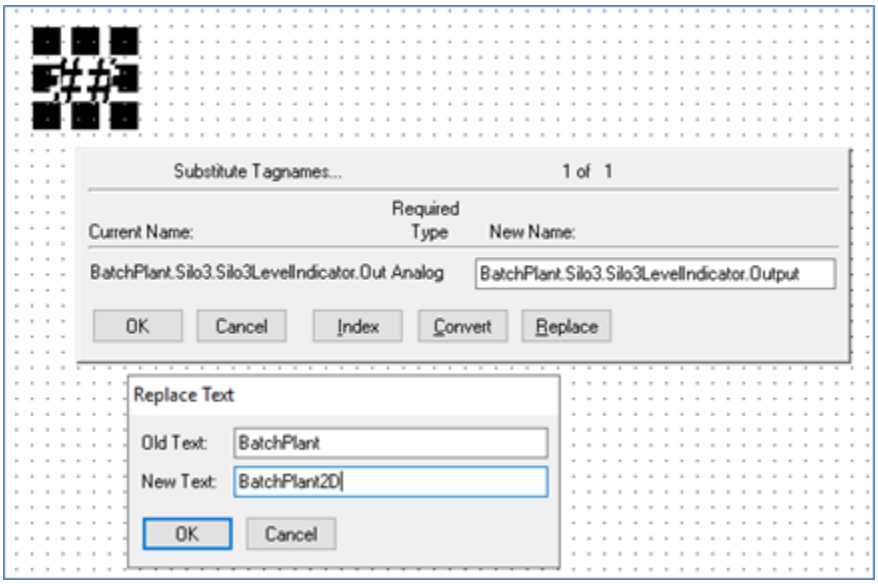
Substitute references

You can substitute UDT instance member tag references in the same way as other InTouch tags, both in the Industrial Graphic Editor and in native InTouch.

Following is an example of Substitute references in an Industrial Graphic:



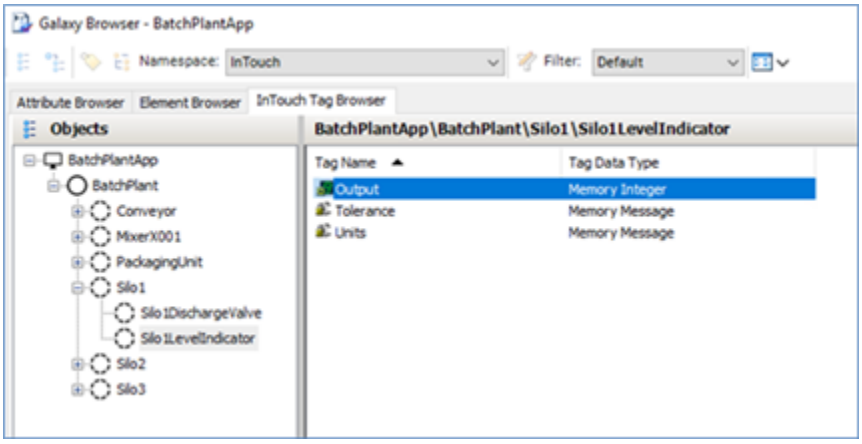
Following is an example of a substitute reference in native InTouch:



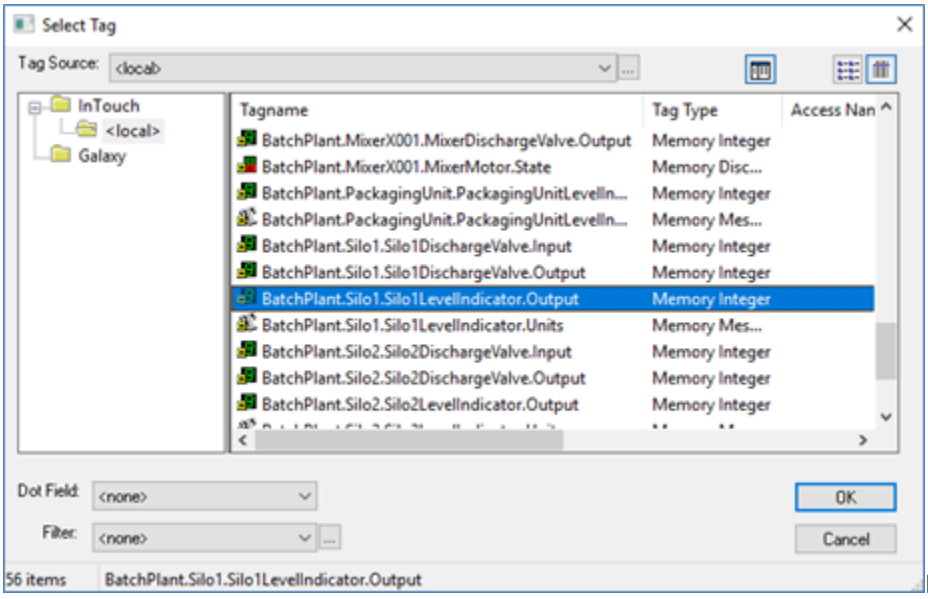
Browse for UDT members

When configuring an animation or using scripting in the Industrial Graphic Editor, use the ellipses button to open the **Galaxy Browser** to browse the UDT member. In the animation or scripting options of native InTouch, open the **Tag Browser** to browse the UDT member.

Following is an example of browsing in the Industrial Graphic Editor:



Following is an example of browsing in native InTouch:

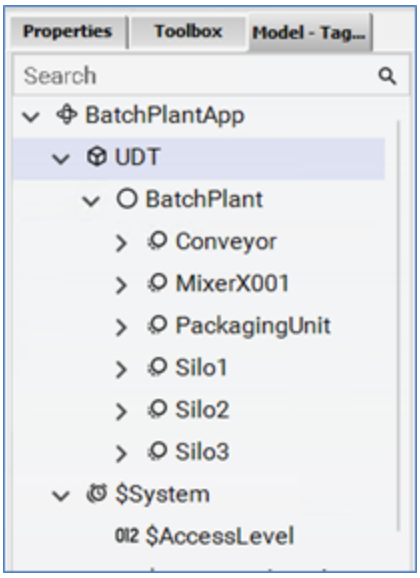


Drag and drop the UDT member of an instance to the Industrial Graphic Editor canvas

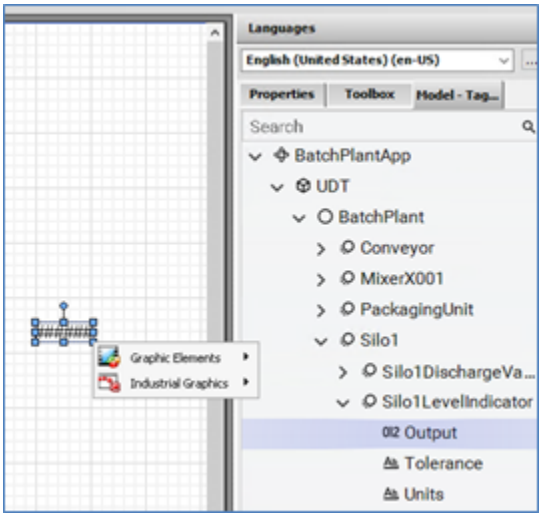
In the Industrial Graphic Editor, the **Model - Tagname** pane also shows UDT instances. You can drag and drop single or multiple UDT members to the canvas to create animations in the Industrial Graphic Editor.

Example:

Drag and drop in the **Model - Tagname:**



Drag and drop in the Industrial Graphic Editor:



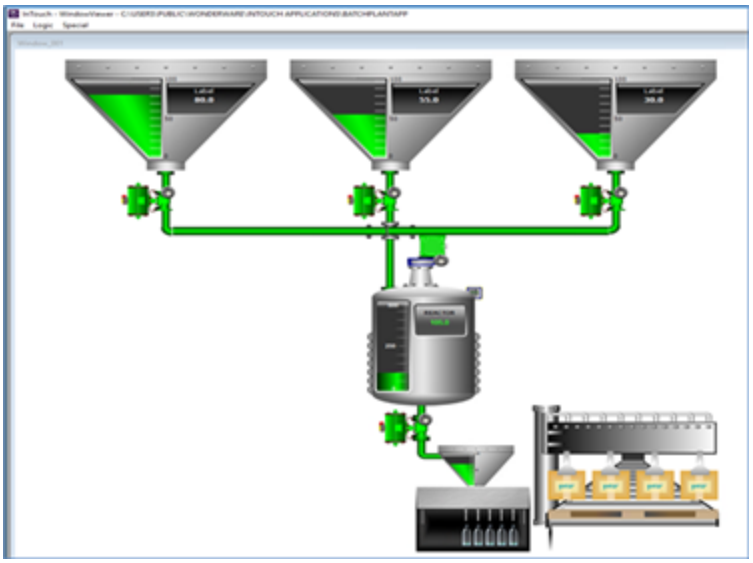
Animation and scripting

UDT instance members are supported in WindowViewer if the graphic is configured with:

- Animation and scripting in an Industrial Graphic and in native InTouch Graphics.
  or
- Scripting in native InTouch.

Example:

In WindowViewer you can see as following:

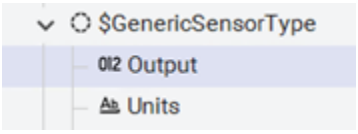


Alarm and Alarm Client Control

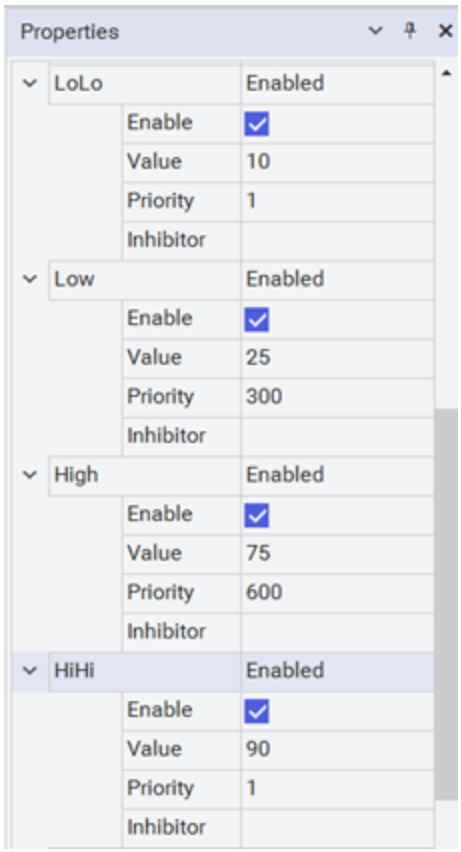
The UDT member of an instance supports alarms in the same manner as other InTouch tags. You can update the alarm properties in the **Properties** grid. In the WindowViewer, you can view the members of an instance that are generating alarms and display them in an Alarm Client Control.

Example: Configure an alarm for a UDT member

1. In **User defined type** view, select $GenericSensorType > Output member.



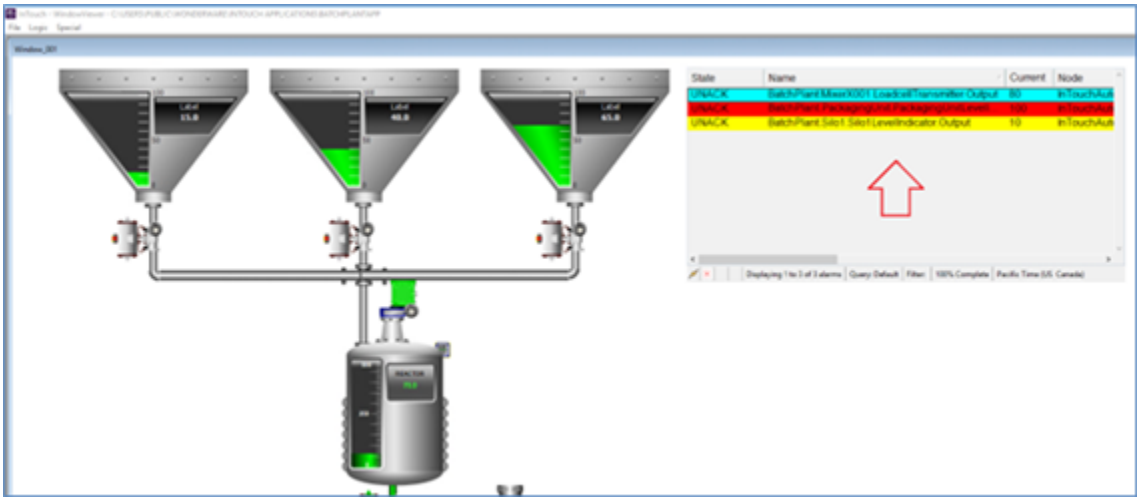
2. In the **Properties** grid, update the alarm properties for this tag as shown in the following illustration.



The alarm properties are then propagated down to the UDT instance level. The following UDT members can have alarms configured as shown in the preceding example.

- BatchPlant.Silo1.Silo1LevelIndicator.Output
- BatchPlant.Silo2.Silo2LevelIndicator.Output
- BatchPlant.Silo3.Silo3LevelIndicator.Output

The following illustration is an example of alarm support for UDTs in WindowViewer:

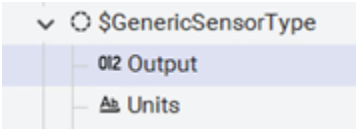


History and trend client

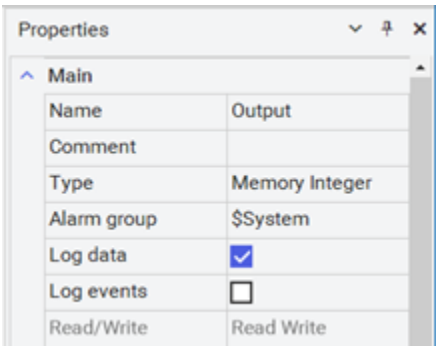
The UDT members of an instance can log data to LGH files or the Historian, as with other InTouch tags.

Example of Enabling Log Data

1. In the **User defined type view**, select the $GenericSensorType > Output member.

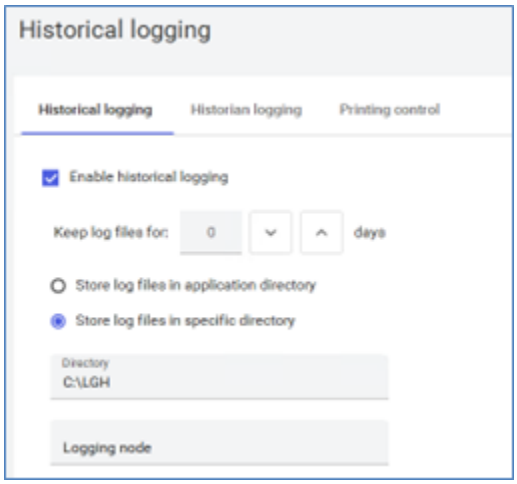


2. In the **Properties** grid, check if the Log data property is updated to True (checked) as shown below:



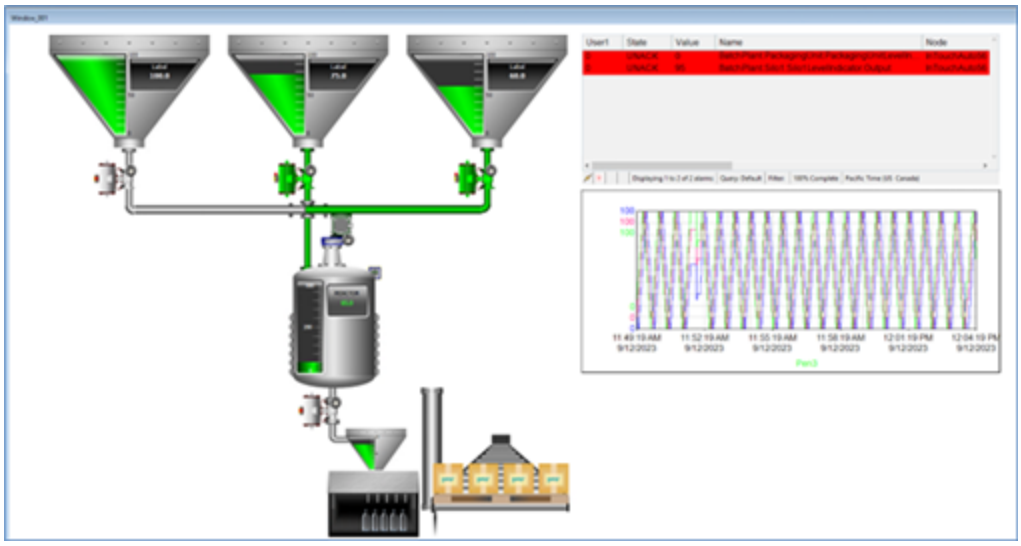
Example of enabling historical logging for LGH:

1. Go to the backstage: File > Configure > Historical Logging



2. If the specified directory does not exist, use File Explorer to create it.
3. Open WindowViewer to go to run time and verify data historizing to the LGH file.

   In the following image, the UDT members are shown logging historical data to the LGH file, and the Trend Client is plotting real time and historical data. The historical data is back filled into the trend.

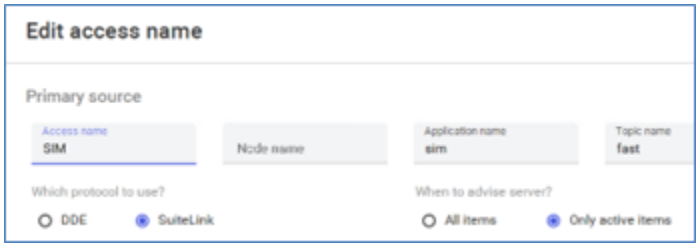


Configure a UDT member as an I/O tag

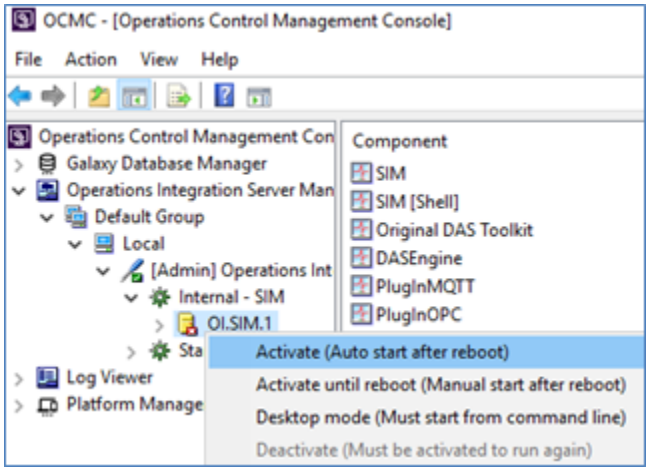
You can configure a UDT member as an I/O tag in the same manner as other InTouch I/O tags. In the **Properties** grid, configure the **Type**, **Access name**, and **Item name** fields to set the member as an I/O tag.

Example: configure a UDT member as an I/O tag

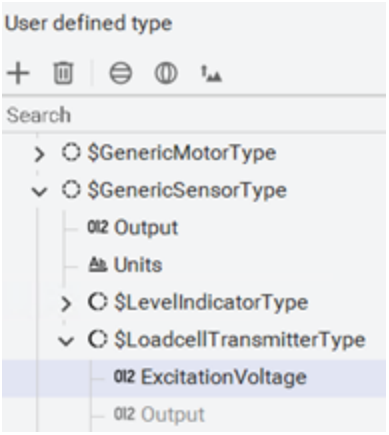
1. Create the access name “SIM” for the local Simulator server.



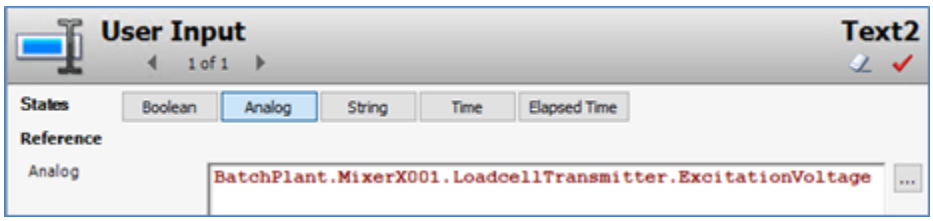
2. Activate the SIM server from the **Operations Control Management Console** (OCMC).



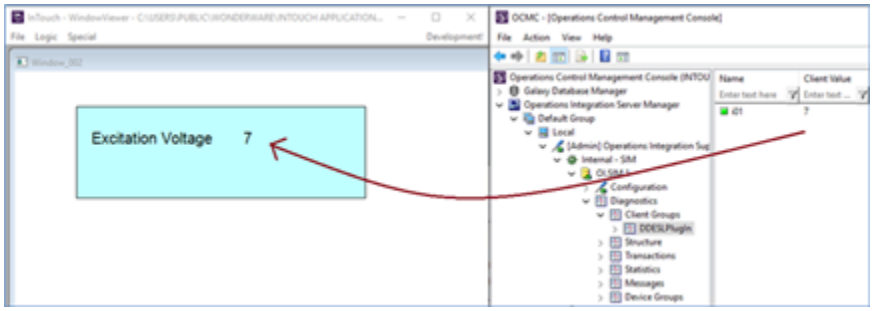
3. Select and configure the ExcitationVoltage example member.



4. Configure properties to make the member an I/O Tag. You can also override the Access name and/or Item name at the instance level.
   a. Set Type to I/O Integer
   b. Set Access name to SIM (you can select from the drop down list)
   c. Set Item name to i01
5. Create a graphic and add an animation with the reference shown in the following illustration.



6. Embed the graphic you just created and view it in run time. The value should bind to the I/O reference.



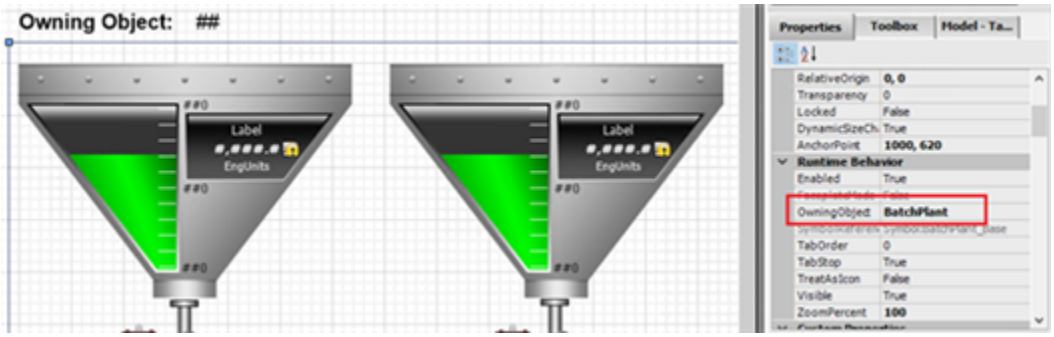
Use a UDT instance as an owning object

You can configure the owning object of an embedded graphic and point to an UDT instance. During run time, the **me.** relative reference is replaced by the UDT instance. You can also update the owning object to a different UDT instance during run time.

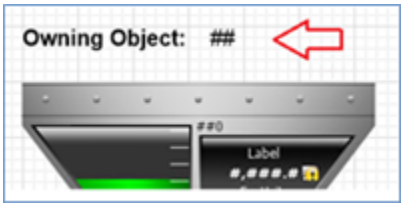
Example: use a UDT instance as an owning object

1. In the Visualization folder, navigate to ReactorDemo Symbols > MainDisplays
2. Edit the BatchPlant_OwningObj graphic.
3. Select the embedded graphic BatchPlant_RelativeRef1.

In the properties grid, notice that the OwningObject attribute is set to BatchPlant, which is the UDT instance.



You can create another UDT instance from BatchPlantType so that in run time, you can switch to a different UDT instance using the input animation.



Example: switch the owning object to a different UDT instance in run time

1. Embed the BatchPlant_OwningObj graphic in another frame window.
2. Close WindowViewer, if it is running.
3. Switch to WindowViewer run time.
4. Verify that the “me.” relative reference is bound to BatchPlant.

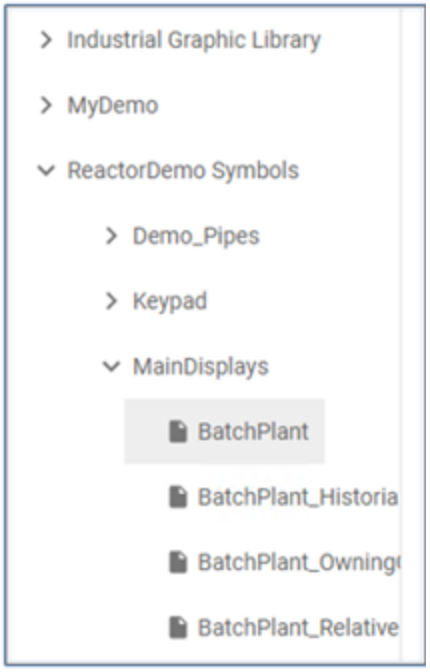
You can change the owning object to a different instance.

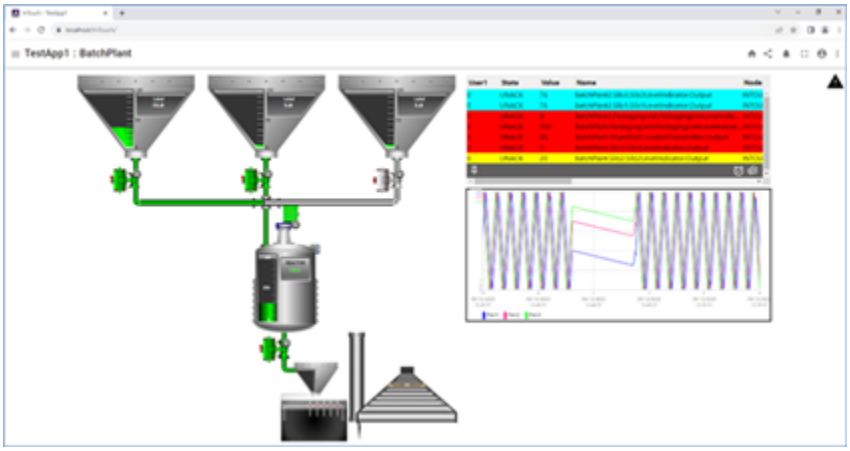
Web Client

The UDT member of an instance works the same as other InTouch tags in the Web Client.

Example:

With the application running in WindowViewer, launch the Web Client from WindowMaker, then open a BatchPlant graphic from the menu.





MapApp

The UDT member of an instance behaves the same as other InTouch tags in the MapApp.

Example for adding location members

1. Select "BatchPlantType" in this example and add new members Latitude and Longitude using the **Add location** option in the toolbar

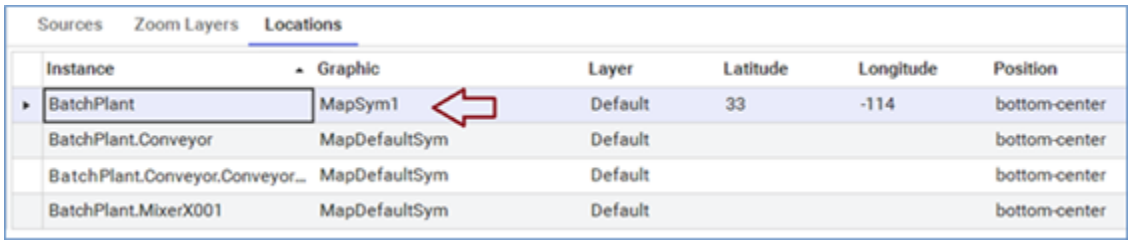
2. Set the following values in the properties grid:
   - Latitude = 33

- Longitude = -114

Example for configuring Map settings

1. In the Visualization folder, go to Widgets and double-click the Map_App.
2. Associate the graphic “MapSym1” with “BatchPlant”. MapSym1 is a graphic that you want to show in the MapApp for this UDT instance BatchPlant.

   The MapApp configuration lists the instances and all members of that instance which are a member data type

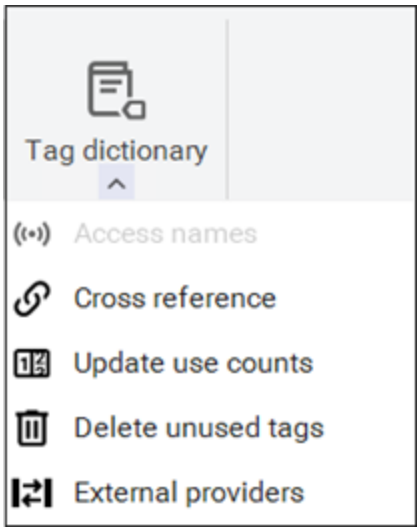


Example for Using the MapApp in run time

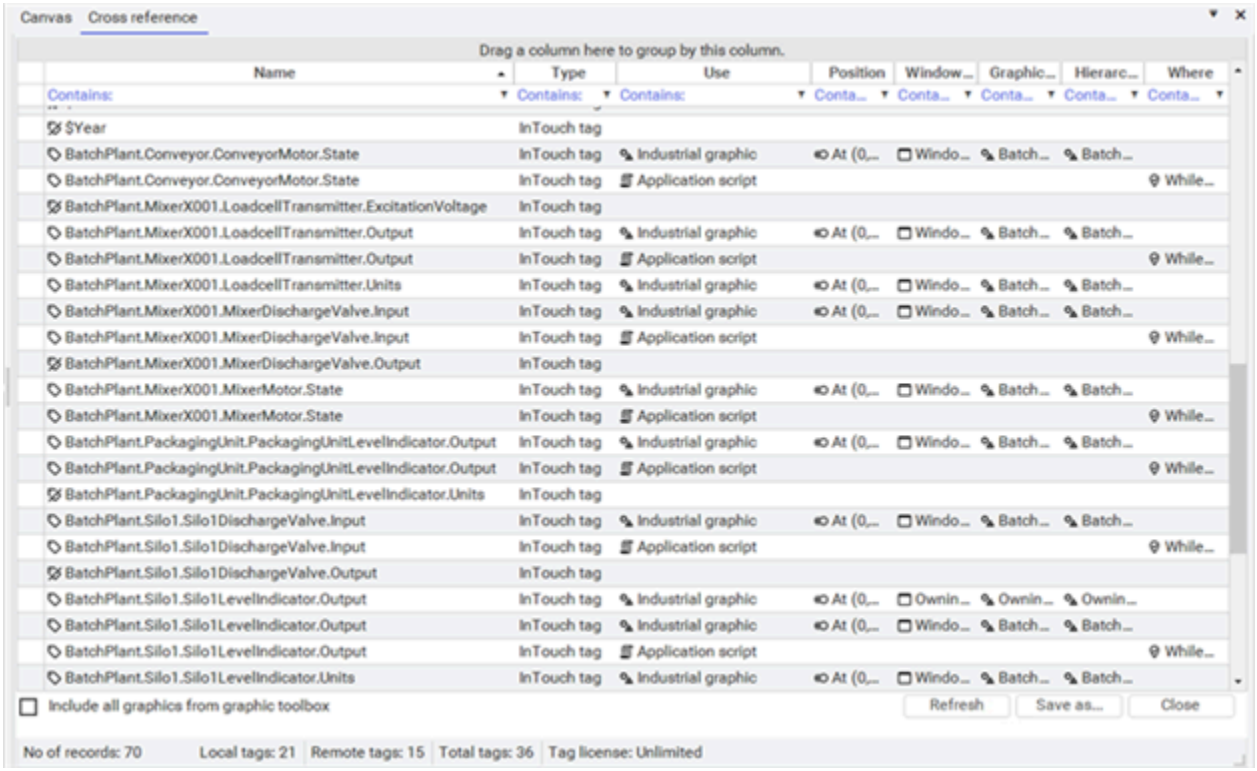
- Embed a graphic “MapAppMain” in another frame window. Close WindowViewer if it is already running, then switch to run time.

Cross-reference

You can use the cross-reference utility to find out where the UDT members of an instance are being used. Select **Tag dictionary** on the WindowMaker ribbon, then select **Cross reference**.



Cross-references example:

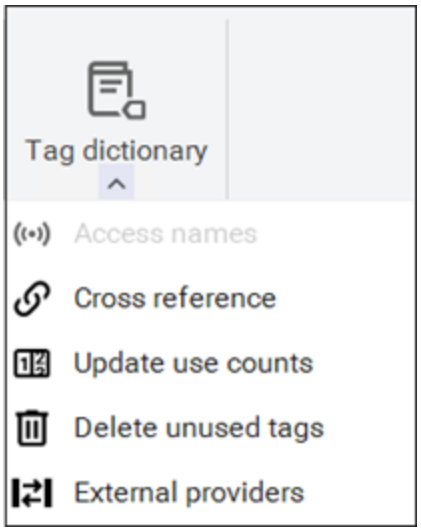


Delete unused tags

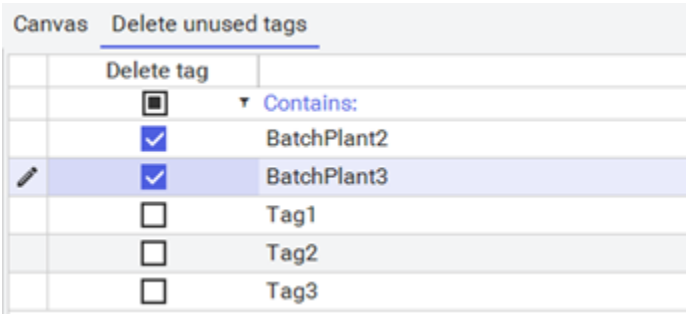
The unused UDT members of an instance are shown in the **Delete unused tag** page and you can delete them from there.

Example:

1. Create two instances of $BatchPlantType and name them BatchPlant2 and BatchPlant3.
2. Select **Delete unused tags**.



You can see that the unused tags BatchPlant2 and BatchPlant3 can be deleted. BatchPlant is not shown because it is being used.

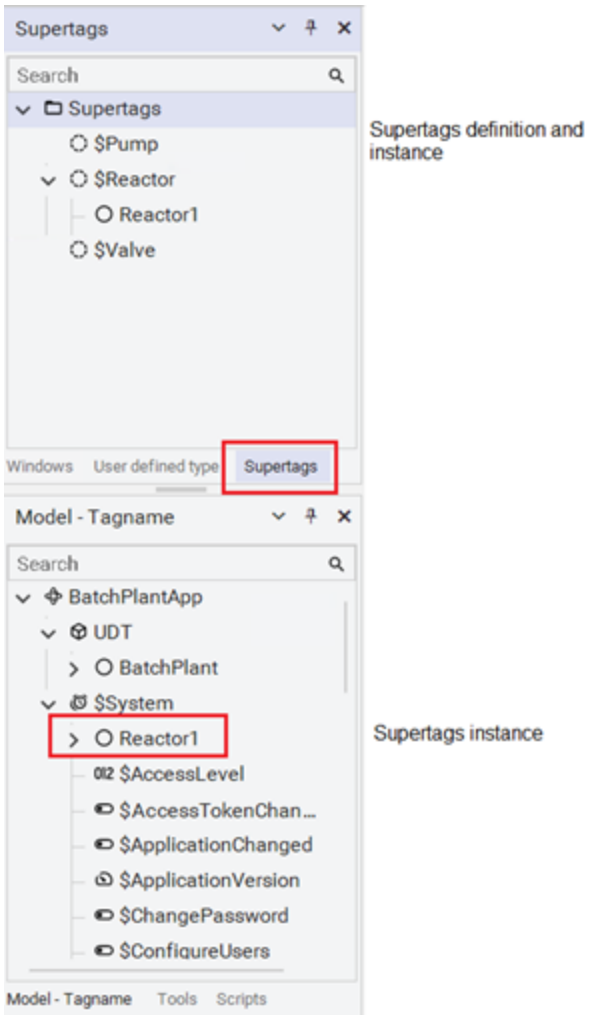


Co-existence with Supertags

UDTs can co-exist with Supertags. You can create Supertags and UDTs with the same name. Supertag instances and UDT instances can work together or separately in scripting, animation, alarms, and trends.

Co-existence behavior is as follows:

- By default, Supertags view is off.
  - This applies to new applications as well to applications that are migrated to the latest release.
  - You can revert back to the Supertags view if desired.
- Supertags continue to use “\” syntax.
- Supertags continue to work the same as in System Platform 2023.
- There is no name collision between Supertag definitions and User Defined Types.
  - You can create Supertags and UDTs with the same name, for example, “Reactor”.
- Supertag instances and UDT instances can work together or separately in scripting, animation, alarms, and trend.
- There is no owning object support for Supertags. This means that there is no “me.” support for Supertags.

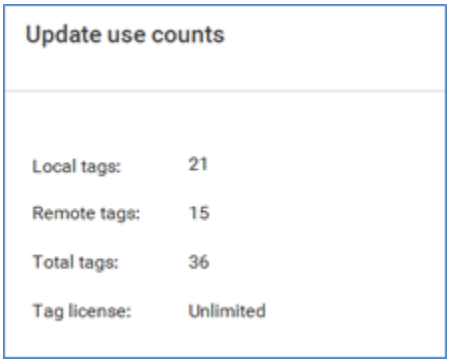


Licensing

Each UDT member of an instance is counted as a local tag. The UDT data type or derived data type does not account for any local tag usage.

Example:

In the below example, the total number of local tags is 21.

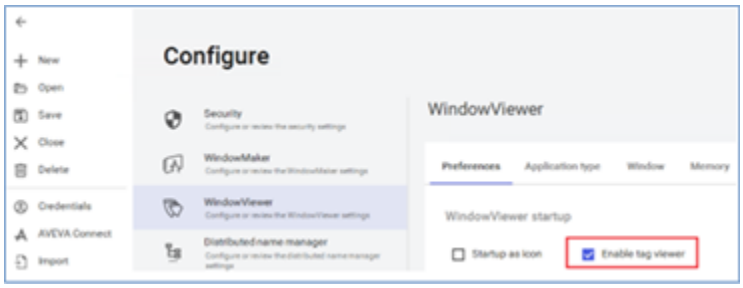


Run-time tag viewer

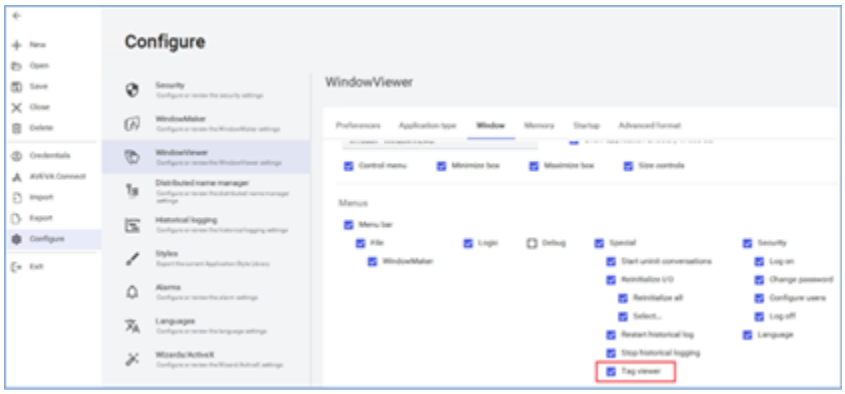
The run-time tag viewer supports UDT members.

Before opening the **Tag Viewer**, make sure that:

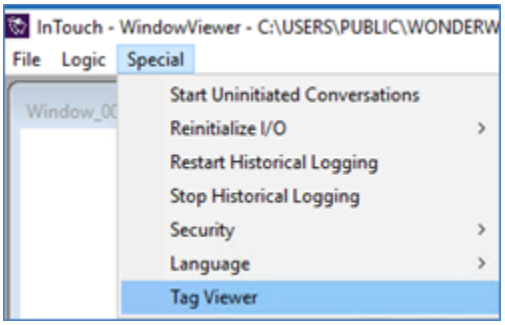
- The **Enable tag viewer** checkbox is selected under **File** > **Configure** > **WindowViewer** >**Preferences**.



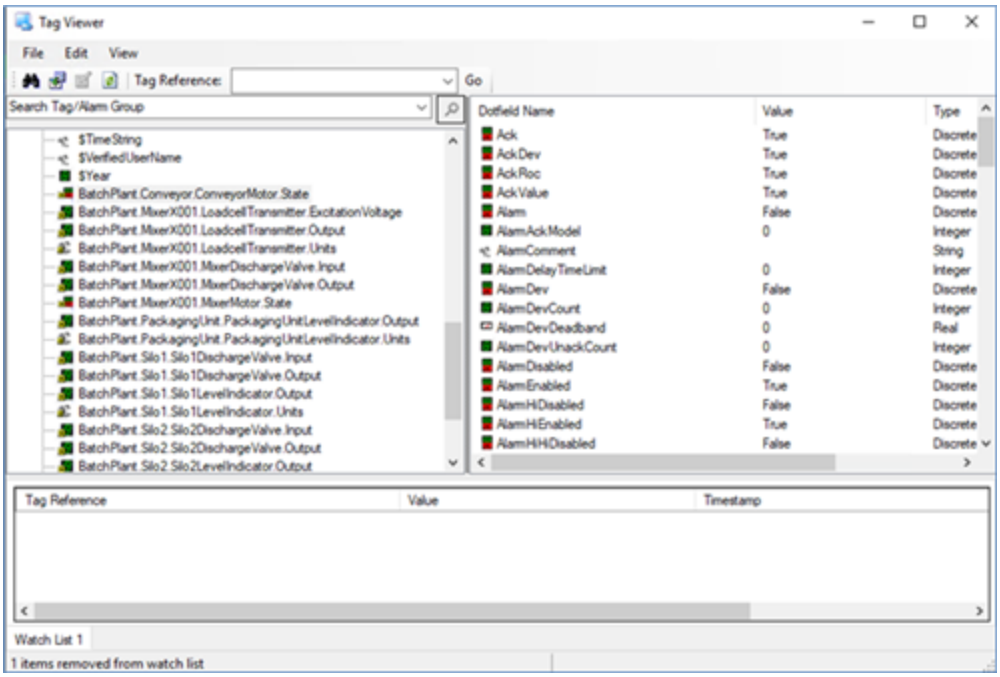
- The **Tag viewer** checkbox is selected under **File** > **Configure** > **WindowViewer** >**Window**.



Run the application in WindowViewer and launch the Tag Viewer.



The Tag Viewer shows the UDT instance members.



UDT limitations

The following functionalities are not supported by UDTs in the current release:

- UDT Tags usage in ActiveX control properties.
- UDT Tags usage in Wizards (Buttons, Clocks, Frames, etc.).
- UDT Tags usage in Trends (Historical Trend).
- Overriding Instance References of a Smart Symbol with UDT Tags.
- UDT Tag usage in the SQL Access Manager Bind List.
- UDT Tag usage in Recipe Manager.
- UDT Tag usage in Minimum and Maximum fields of Native User input animations.
- Tag Browser from the InTouch Proxy Object.
- UDT tags usage in alarm comments.
- UDT tags usage in language switching.
- Retentive properties of a tag.
- UDTs usage in native animations like discrete alarm and analog alarm

Alarms

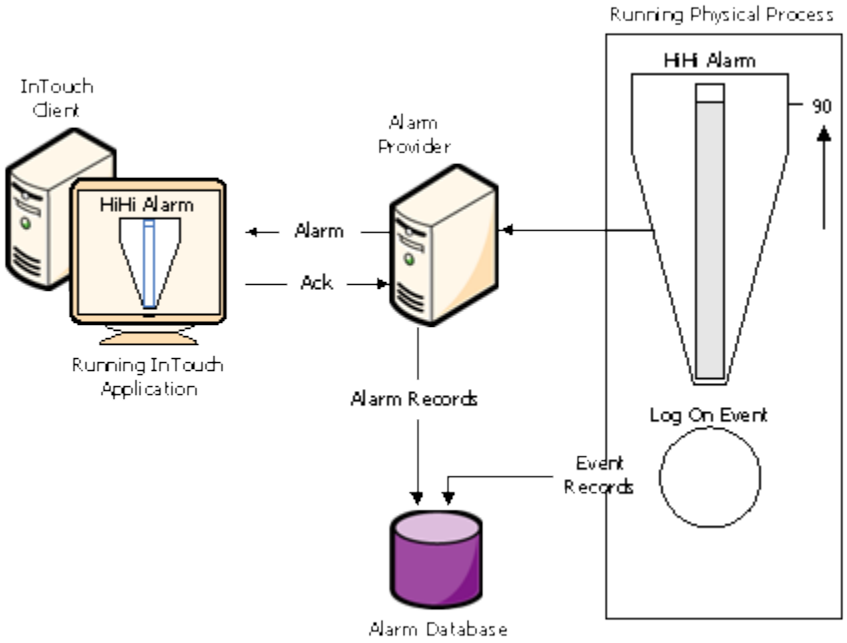
You can create InTouch applications that generate alarms and events to notify operators about the status of process activity.

- Alarms warn run-time operators about process conditions that could potentially cause problems. Typically, you set up an alarm to trigger when a process value exceeds a defined limit. An operator must usually

acknowledge the alarm.

- Events represent normal system status messages. A typical event is when a system condition occurs, such as an operator logging on to an InTouch application. Operators do not have to acknowledge events.

The following figure shows how the InTouch HMI handles alarms and events while an application is running. Alarm and event data is saved to the alarm database.



You can configure any tag for event monitoring. An event message is logged to the alarm system each time the tag value changes. The event message includes how the value changed and whether the operator, I/O, scripts, or the system initiated the change.

Configure alarms

To configure alarms, you simply configure tags with alarm conditions.

If required, you can also:

- Define alarm hierarchies.
- Disable and inhibit alarms.
- Configure alarm comments.
- Configure miscellaneous alarm and event properties.

Define alarm hierarchies

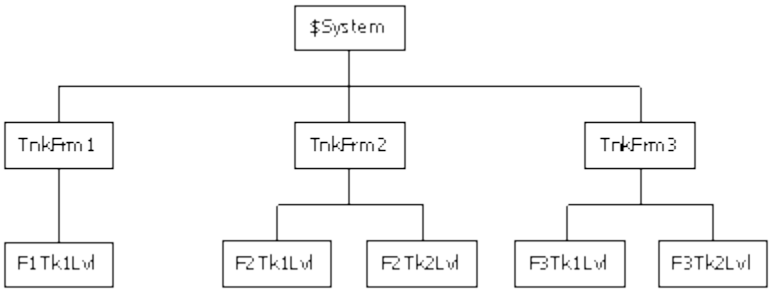
Each InTouch alarm belongs to an alarm group. Organizing related alarms into groups makes it easier for an operator to filter, show, and acknowledge alarms. For more information, see Alarm Groups.

The Distributed Alarm system uses alarm groups as the basis for its alarm group lists. For more information about creating Distributed Alarm system group lists, see Create an alarm group list file.

Create an alarm group

Before you start creating alarm groups, have a plan for how your alarm groups should be organized and what you want the alarm group names to be. Using a consistent group naming convention enforces a logical ordering of groups within the hierarchy.

In the following figure, notice the similarity of names assigned to groups at the same level within the hierarchy.



Also, notice that subordinate group names reference their parent groups by including a portion of the parent name. For example, the third-level alarm group name F1Tk1Lvl references its alarm group parent TnkFrm1 by including the F1 prefix in its group name. Develop a naming convention that suggests the parent-child relationship between alarm groups at different levels within the hierarchy.

Note: While alarm groups do not count as tags for InTouch licensing, they do count as tags in the database. Therefore, the total number of alarm groups plus actual tags cannot exceed the maximum limit set by your InTouch license.

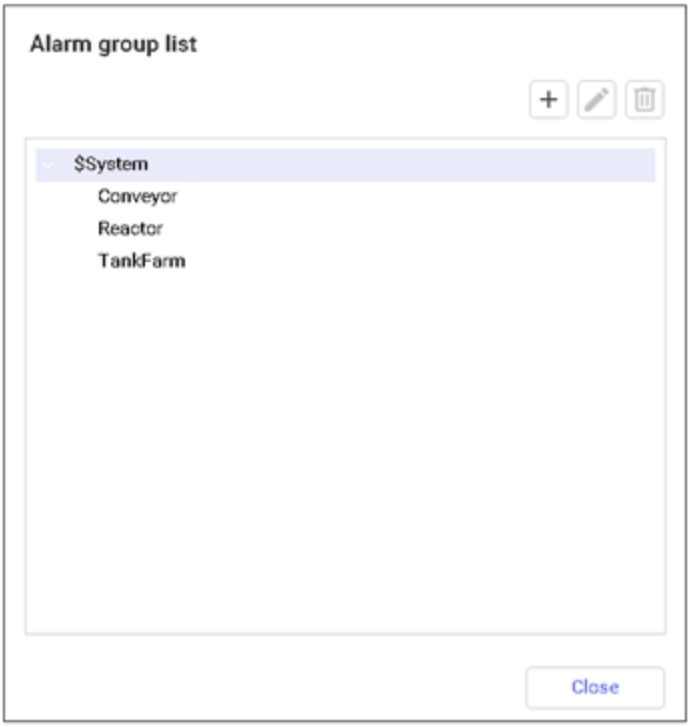
An alarm group name must meet the following requirements:

- A name must be 32 characters or fewer.
- A name must begin with an alphanumeric character (A-Z, a-z, or 0-9).
- A name can contain the following keyboard characters (@, #, $, %, &, -, _, ?, !, \) beginning at the second character position within the name.
- If a name contains a hyphen (-), the name must begin with an alphabetic character.
- A name cannot contain a blank space.
- A name must have at least one alphabetic character.

Create an alarm group

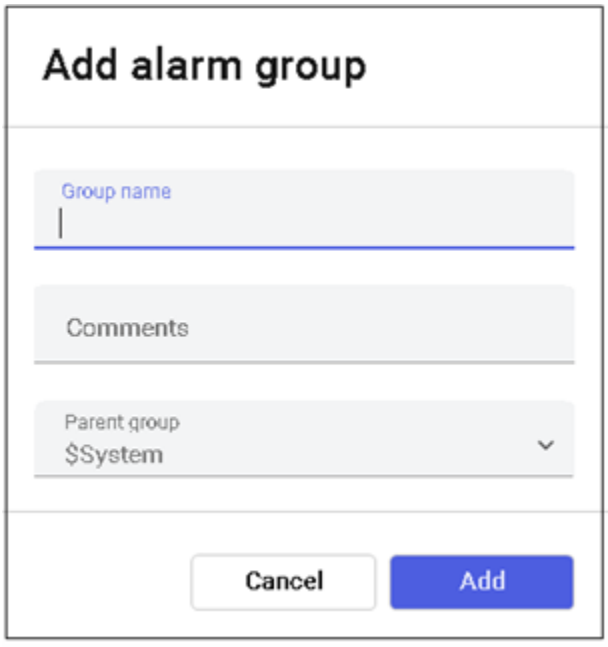
1. On the **Home** menu, in the **Alarms** group, select **Alarm groups**.

   The **Alarm group list** window appears.



2. Select the **+** icon to Add, or press Alt+A.

   The **Add alarm group** dialog box appears.



3. In the **Group Name** box, type a name for the new alarm group.
4. In the **Comment** box, type an optional comment up to 49 characters for the new alarm group.
5. To reassign the alarm group to another parent group:
   a. Select the **Parent group** name from the list. If this is the first alarm group defined for the InTouch application, the group is automatically assigned to the parent $System group.
   b. Select a new parent group from the list.

6. Select **Add**.
   The **Alarm groups** list window displays the new alarm group added to the list.
7. Select **Close**.

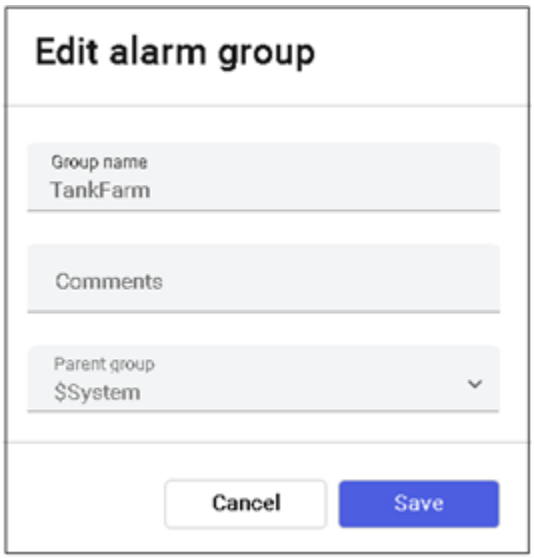
Modify an alarm group

You can modify an alarm group to:

- Rename it.
- Change the associated comment.
- Reassign it to another group.

Modify an alarm group

1. On the **Home** menu, in the **Alarms** group, select **Alarm groups**.
   The **Alarm group list** window appears.
2. Highlight the alarm group to modify and select the Edit icon, or press Alt+E.
   The **Edit alarm group** window appears.



3. Make any changes to the alarm group's name or comment.
4. To reassign the alarm group to another parent group:
   a. From the **Parent group** dropdown list, select a new parent group.
5. Select **Save**.
6. Select **OK**.

Delete an alarm group

You can delete an alarm group and remove the group from the hierarchy. Alarms and tags that belong to deleted alarm groups are automatically reassigned to the immediate parent group above the deleted group in the hierarchy. Also, child groups of the deleted group are reassigned to the immediate parent group above the

deleted group.

Delete an alarm group

1. On the **Home** menu, in the **Alarms** group, select **Alarm groups**.
   The **Alarm group list** window appears.
2. Highlight the alarm group and select the **Delete** icon, or press Alt+D. When a message appears, select **Yes**.
3. Select **Close**.

Configure tags with alarm conditions

You can configure any tag for an alarm by specifying the type of alarm and one or more alarm thresholds. Whenever the value of a tag reaches a defined threshold, an alarm occurs. Any transitions of a tag's value in and out of an alarm state are reported to the Distributed Alarm system.

Configure discrete alarms

A discrete alarm corresponds to a discrete tag. You can configure whether the alarmed state corresponds to the discrete tag's true (On, Yes, 1) state or false (Off, No, 0) state.

Define alarm conditions for a discrete tag

1. Open the Tagname Dictionary.
2. Select an existing discrete tag or create a new discrete tag.
3. Select either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog box to show the discrete alarm details dialog box.

4. In the **ACK Model** area, select the alarm acknowledgement model for the tag.
   - Select **Condition** for acknowledgment to count against all transitions into the alarmed state or a sub-state up to the time of the acknowledgement. This is the default acknowledgement model.
   - Select **Event Oriented** for an acknowledgment to only be for a particular transition to the alarmed state or a sub-state; an acknowledgment is accepted only if it refers to the most recent transaction.
   - Select **Expanded Summary** for an acknowledgment to only be for a particular transition, whether to an alarmed state, to a sub-state, or a return to normal. Each transition from the normal state marks the beginning of a new "return to normal" (RTN) group. All transitions in an RTN group must be acknowledged individually before the overall RTN group is considered acknowledged.
5. In the **Alarm Comment** box, type an alarm comment up to 131 characters.

   Note: The Alarm Comment box should not contain the double-quote character ("). A process that uses the double-quote as a delimiter will fail if it is fetching an alarm comment that includes a double-quote.
6. In the **Alarm State** area, select the active alarm state to be the discrete tag's **On** or **Off** value.
7. In the **Priority** box, assign an alarm priority number between 1 to 999. The default priority number is 1, which is the highest alarm priority.

8. Optionally assign an alarm inhibitor tag for the discrete alarm.
   a. In the **Alarm Inhibitor** box, select the button to show the **Select Tag** dialog box containing a list of defined tags.
   b. Highlight a tag in the list and select **OK**. The name of the tag you selected as the inhibitor tag appears in the **Alarm Inhibitor** box.

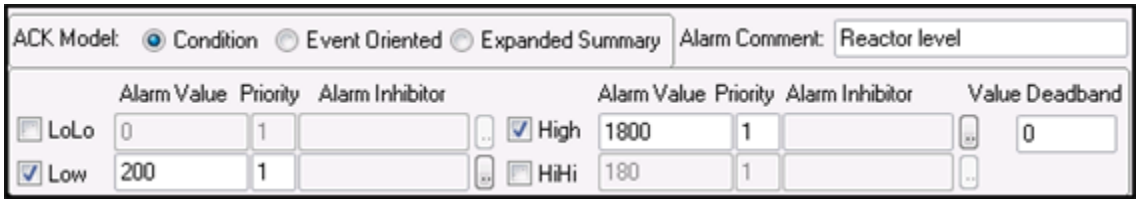
   For more information on inhibiting alarms, see Inhibit alarms.
9. Select **Save**.
10. Select **Close** to close the **Tagname Dictionary** dialog box.

Configure value alarms

A value alarm is associated with integer or real tags. You can set alarms when the tag value transitions from a set of predetermined thresholds that range from LoLo to HiHi. You can configure whether the alarmed state corresponds to any value of the tag and the associated priority of that alarm.

Configure a value alarm

1. Open the Tagname Dictionary.
2. Select an existing real or integer tag or create a new tag.
3. Select either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog box to show the alarm details dialog box.



4. In the **ACK Model** area, select the alarm acknowledgement model for the tag.
   - Select **Condition** for acknowledgment to count against all transitions into the alarmed state or a sub-state up to the time of the acknowledgement. This is the default acknowledgement model.
   - Select **Event Oriented** for an acknowledgment to only be for a particular transition to the alarmed state or a sub-state. An acknowledgment is accepted only if it refers to the most recent transaction.
   - Select **Expanded Summary** for acknowledgment to only be for a particular transition, whether to an alarmed state, to a sub-state, or a return to normal. Each transition from the normal state marks the beginning of a new "return to normal" (RTN) group. All transitions in an RTN group must be acknowledged individually before the overall RTN group is considered acknowledged.
5. In the **Alarm Comment** box, type a default comment up to 131 characters. The comment is assigned to the tag's .AlarmComment dotfield.

   Note: The Alarm Comment box should not contain the double-quote character ("). A process that uses the double-quote as a delimiter will fail if it is fetching an alarm comment that includes a double-quote.
6. Select the alarm types (**LoLo**, **Low**, **High**, **HiHi**) to detect when the value of the tag is beyond an absolute limit.
7. In the **Alarm Value** boxes, type the limit values for the alarm types.

   For example, in the case of **LoLo** and **Low** alarms, an alarm condition exists whenever the value of the tag is less than the **Alarm Value**. In the case of **High** and **HiHi** alarms, an alarm occurs whenever the value of the

tag exceeds the **Alarm Value**. You can use real numbers for the limits.

8. In the **Value Deadband** box, type the number of engineering units the tag value must drop below or above the alarm value before it transitions out of an alarm state.

   For example, to return-to-normal from an alarm condition, a tag value must not only return inside its alarm limit, but also return through your specified Value Deadband. The Value Deadband prevents nuisance alarms caused by repetitive re-annunciation of an alarm where the tag value hovers around the limit, continually fluctuating in and out of an alarm state.

9. Optionally assign an alarm inhibitor tag for the tag's alarm types (LoLo, Low, High, HiHi).

   a. In the **Alarm Inhibitor** area, select the button to show the **Select Tag** dialog box containing a list of defined tags.

   b. Highlight a tag in the list and select **OK**. The name of the tag you selected as the inhibitor tag appears in the **Alarm Inhibitor** box.

   For more information on inhibitor tags, see Inhibit alarms.

10. Select **Save**.

11. Select **Close** to exit the **Tagname Dictionary** dialog box.

Configure deviation alarms

A deviation alarm is associated with integer or real tags. You can set an alarm by comparing the current tag value to a target value, and then the absolute value of the difference is compared to one or more limits, expressed as a percentage of the range of the tag value.

For example, the following values set the conditions for a tag's minor and major deviation alarms:

Minimum Value = -1000

Maximum Value = 1000

Minor Deviation % = 10

Major Deviation % = 15

Target Value = 500

Using these values as an example, the minor and major deviation alarms points are calculated by the following steps:

1. Calculate the total value range of the tag.

   1000 - (-1000) = 2000

2. Multiply the total value range of the tag by minor and major deviation percentages.

   2000 x 0.10 = 200 = minor deviation limit

   2000 x 0.15 = 300 = major deviation limit

3. Add and subtract the minor and major deviation limits from the target value.

   500 - 200 = 300 = minor deviation lower limit

   500 + 200 = 700 = minor deviation upper limit

   500 - 300 = 200 = Major deviation lower limit

   500 + 300 = 800 = Major deviation upper limit

Configure a deviation alarm

1. Open the Tagname Dictionary.
2. Select an existing real or integer tag or create a new tag.
3. Select either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog box to show the alarm details dialog box.

4. Select the deviation (**Minor** and **Major Deviation**) alarm types you want to use to detect when the value of an analog type tag is in a major or minor deviation from the specified target value.
5. In the **%Deviation** box, type the percentage that the analog tag can deviate from the target value to trigger a minor or major deviation alarm condition. It is expressed as a percentage of the range of the tag. For an I/O tag, the **Min EU** and **Max EU** values entered in the tag's details dialog box define the range. For memory tags, the range is defined by the minimum value and maximum value.
6. In the **Target** box, type the tag reference value that minor and major deviation percentages are based.
7. In the **Deviation Deadband %** box, type the deviation percentage the tag value must drop below the limit before the tag is taken out of its alarm condition.
8. Select **Save**.
9. Select **Close** to close the **Tagname Dictionary** dialog box.

Configure rate of change alarms

A rate of change alarm detects when the absolute value of an alarm exceeds a specified limit over a measured interval. A tag is tested for a rate-of-change alarm whenever its value changes. The change rate is calculated using the previous tag value, the time of the previous change, the current tag value, and the current time.

The calculated rate of change over time is compared to a rate-of-change percentage limit specified for a tag. If the calculated rate-of-change is greater than the percentage limit, an alarm condition is set for the tag. A rate-of-change alarm remains active until the rate at which the tag is changing is less than the alarm limit.

Configure a rate of change alarm

1. Open the Tagname Dictionary.
2. Select an existing real or integer tag or create a new tag.
3. Select either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog box to show the alarm details dialog box. The following figure shows only those options that apply to rate-of-change alarms.

4. Select the **Rate of Change** box.
5. In the **% per** box, enter the maximum allowable percentage change limit.
6. Select **Sec**, **Min,** or **Hr** as the time interval unit.
7. In the **Priority** box, type a number between 1 and 999 to set the alarm priority.
8. Optionally assign an alarm inhibitor tag for the rate of change alarm.

a. In the **Alarm Inhibitor** area, select the button to show the **Select Tag** dialog box containing a list of defined tags.

b. Highlight a tag in the list and select **OK**. The name of the tag you selected as the inhibitor tag appears in the **Alarm Inhibitor** box.

For more information on inhibiting alarms, see Inhibit alarms.

9. Select **Save**.
10. Select **Close** to close the **Tagname Dictionary** dialog box.

Disable alarms

You can disable or enable all alarms of a tag at once using the .AlarmEnabled or AlarmDisabled dotfields. For an alarm that has sub-states, each sub-state can be individually disabled. For example, an analog value alarm can have Hi enabled and HiHi disabled.

During run time, the Alarm Provider does not generate alarms for an alarm or sub-state that is disabled. Changes to whether an alarm is disabled or enabled can be made at run time.

Whenever an alarm transitions from disabled to enabled, the checking logic determines whether the item should be put in the alarmed state by the Alarm Provider.

If an alarm becomes disabled or actively inhibited while the item is in an alarmed state, the item will be forced to a different (valid) state. What that state should be depends upon which states are available and whether they have also been disabled. This activity is handled by the Alarm Provider according to the type of alarm and limit values.

Inhibit alarms

You can optionally assign to each alarm or alarm sub-state an inhibitor alarm tag that prevents the alarm from transitioning into an active state.

- When the inhibitor tag value becomes and remains TRUE (non-zero or non-NULL), the alarm is inhibited.
- Likewise, when the inhibitor alarm tag becomes and remains FALSE (zero or NULL), the alarm is not inhibited.

You can only change the inhibitor tag in WindowMaker. You can change the value of an inhibitor tag at run time.

You can assign inhibitor tags to individual alarm sub-states. Each sub-state can be inhibited by a different tag, and you can leave some sub-states with no inhibitor tag assigned.

An alarm that is inhibited (and for which the tag is TRUE) is not waiting for an acknowledgment. If the alarm has sub-states, it can only be waiting for an acknowledgment on sub-states that are still available.

An alarm or sub-state can be independently disabled, inhibited, or both. Only if the alarm is both enabled and not actively inhibited is the alarm capable of becoming active.

If an alarm or sub-state has no inhibitor tag assigned to it, the effect is the same as if it had an inhibitor tag that is always FALSE.

Whenever the transition causes an alarm to change from being actively inhibited, the checking logic determines whether InTouch should put the item in the alarmed state.

If an alarm becomes actively inhibited while the item is in an alarmed state, the item must be forced to a different (valid) state. What that state should be depends upon which states are available and whether they have also been disabled or actively inhibited. This activity is handled by InTouch according to the type of alarm, limit

values, and so on.

If an alarm (or an alarm sub-state) becomes actively inhibited while waiting for an acknowledgment, the item must be forced to a different (valid) state. As with whether the item is alarmed, InTouch must determine what this state should be.

Alarm inhibitor tags are included in use counts and license limitations.

Use the following read-only tag dotfields to get the name of the alarm inhibitor tag:

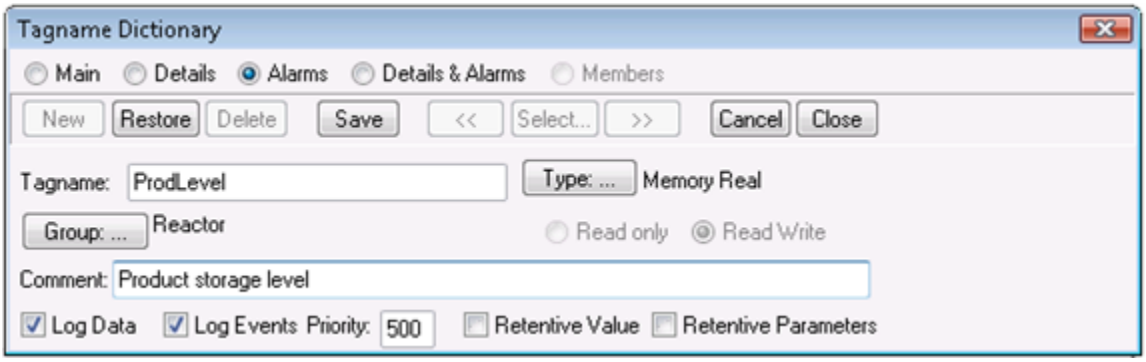
- AlarmDscInhibitor
- AlarmLoLoInhibitor
- AlarmLoInhibitor
- AlarmHiHiInhibitor
- AlarmHiInhibitor
- AlarmMajDevInhibitor
- AlarmMinDevInhibitor
- AlarmRocInhibitor

These fields return the name of a tag. Therefore, you can use the name in an indirect tag reference in an InTouch QuickScript to find out the current value of the alarm inhibitor tag, or to change the value of the alarm inhibitor tag. By doing this, you can force groups of alarms to be enabled or actively inhibited during run time.

Set event properties for individual tags

When you define a tag to do event monitoring, an event message is logged to the alarm system each time the tag's value changes. The event message logs how the value changed. For example, whether the operator, I/O, QuickScripts, or the system initiated the change.

1. Open the Tagname Dictionary.
2. Select an existing tag or create a new tag associated with data that will be recorded as an event.
3. Select **Log Events**. The Priority box becomes available. The value you enter for the Priority determines the event priority level for the tag.



4. In the **Priority** box, assign a number from 1 to 999 as the event priority. 1 is the highest event priority and 999 the lowest.
5. Select **Save**.
6. Select **Close** to close the **Tagname Dictionary** dialog box.

Configure global settings for alarms and events

You can configure the following global settings that apply to all alarms and events generated by an application:

- Internal alarm memory (buffer) size.
- Whether an alarm's return to normal implies acknowledgement. For more information, see Use automatic acknowledgement when the tag value returns to normal.
- Event logging.
- Whether alarm enabling is retentive when WindowViewer is restarted.
- Whether to use alarm acknowledgement comments as the general alarm comment. For more information, see Use alarm and acknowledgement comments.
- Whether to display LATCHED alarms in the Alarm Client Control grid. For more information, see Enable latched state.

Configure the alarm buffer size

Communication in the Distributed Alarm system consists largely of alarm queries and alarm records sent between nodes. Within a node, alarm queries and records are held in the InTouch internal alarm memory, also called the alarm buffer, to minimize network traffic. The alarm buffer size is maximum number of alarms the node can store for summary or historical alarm queries. The alarm buffer deletes the oldest records to make room for new records.

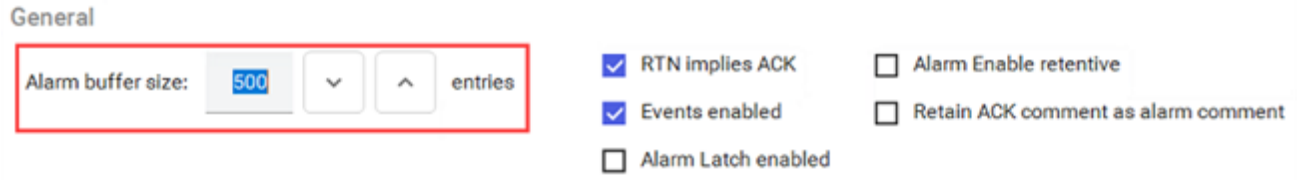
Only alarm events stored in memory can be shown in an application window. If your InTouch application does not show any alarm status, you can set the buffer size to 1 to conserve node memory.

Assigning a large value to the alarm buffer can potentially affect node performance. For a Distributed Alarm system, we recommend the default value of 500.

Configure the alarm buffer size

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Alarms**.

   The **Alarms** configuration screen appears.



3. In the **Alarm buffer size** box, type the maximum number of alarm entries that can be stored in the memory alarm buffer for summary or historical queries.
4. Select **Save**.

Enable events

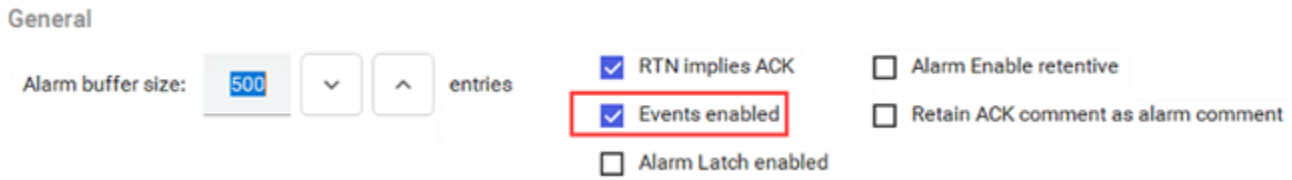
You can enable events to be logged within the application. An event represents a recognized change in application data resulting from an operator action, QuickScript, or I/O.

A tag's **Log Events** property must be set from the Tagname Dictionary before the tag's associated events are stored in the internal alarm memory or logged to the alarm database. For more information about specifying event logging for a tag, see Set event properties for individual tags.

Enable events

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Alarms**.

   The **Alarms** configuration screen appears.



3. Select the **Events enabled** checkbox to log all events that occur while an InTouch application is running.
4. Select **Save**.

Make alarm enabling retentive

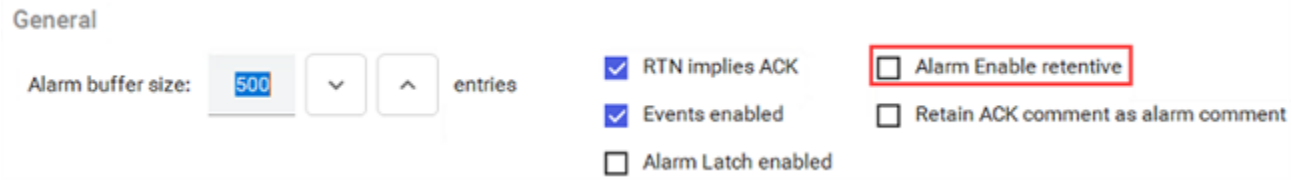
You can select to retain the current value assigned to a tag's .**AlarmEnabled** dotfield when the InTouch application is stopped and then restarted.

The value assigned to the .**AlarmEnabled** dotfield toggles alarm and event logging on or off. The .**AlarmEnabled** dotfield can be assigned to tags or alarm groups. When the .**AlarmEnabled** dotfield is assigned to an alarm group, it determines whether all alarms associated with the tags within the specified alarm group are logged or not.

Make alarm enabling retentive

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Alarms**.

   The **Alarms** screen appears.



3. Select the **Alarm event retentive** checkbox to retain the current state of the **.AlarmEnabled** dotfield as the initial value when the InTouch application re-started.
4. Select **Save**.

Enable latched state

You can enable the LATCHED state to view the LATCHED alarms in the Alarm Client Control grid. Alarms go to the LATCHED state if:

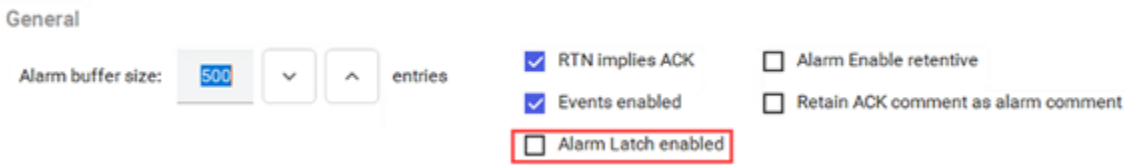
- You ACK an alarm from UNACK_RTN state.

Or

- An Alarm Returns to Normal (RTN) from the ACK state.

Enable the latched state:

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Alarms**.
   The **Alarms** configuration screen appears.



3. Select the **Alarm Latch enabled** checkbox to view the LATCHED alarms in the Alarm Client Control grid.
4. Select **Save**.

Create an alarm group list file

You use the Distributed Name Manager to create an alarm group list. Then, you add existing alarm groups from the local and remote nodes to the list.

The following table shows the syntax to specify alarm groups from the Distributed Name Manager.

Node	Alarm Group Syntax
Local	\InTouch!Group_Name or .Group_Name
Remote	\\Node_Name\InTouch!Group_Name or Node_Name.Group_Name

For these examples, Node_Name is the name of the InTouch remote node. Group_Name is the name of the alarm group. If the alarm group is defined on the local node where the alarm group list is being defined, you can simply enter the alarm group name with a preceding period. For example, .Group_Name.

Create an alarm group list

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **Distributed Name Manager**.
   The **Distributed Name Manager** configuration screen appears.
3. In the **Distributed Alarms** tab, select the Add (**+**) icon.
   The **Add distributed alarm** dialog appears.

4. In the **Members** box, type the list of InTouch nodes and alarm groups to be included in the query.

   You can enter the node names and alarm group names using Standard Group Entry syntax or as short cut entries using periods. Short cut entries are converted to Standard Group Entries when you save the alarm group list.

   Note: The Node.Group and .Group syntax can be used only in this configuration dialog box. It is not valid in the alarm display configuration or any alarm QuickScript function.

5. Select **Add** to add the list to your alarm group file.

   The syntax of the members is automatically converted.

6. Select **Save**.
7. Add the name of the alarm group list to a query in an Alarm Viewer control. The Alarm Viewer control now shows alarms for all groups specified in the list.

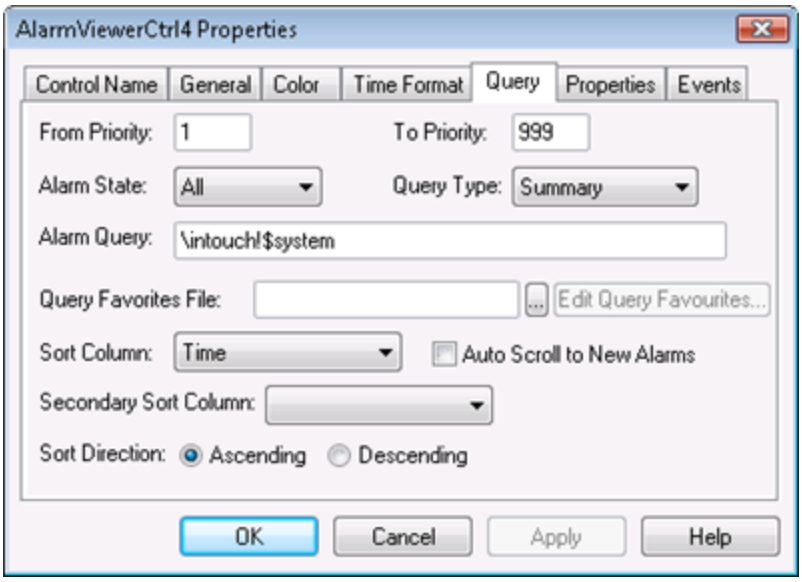
Alarm queries

An alarm query retrieves either:

- Alarms and events (historical alarms) from the InTouch internal alarm memory or the alarm database.
- Current alarms (summary alarms) from the InTouch internal alarm memory.

When you configure an InTouch alarm ActiveX control, you specify the query source. You can also select query options to filter the query results.

The following figure shows the **Query** tab for the Alarm Viewer ActiveX control.

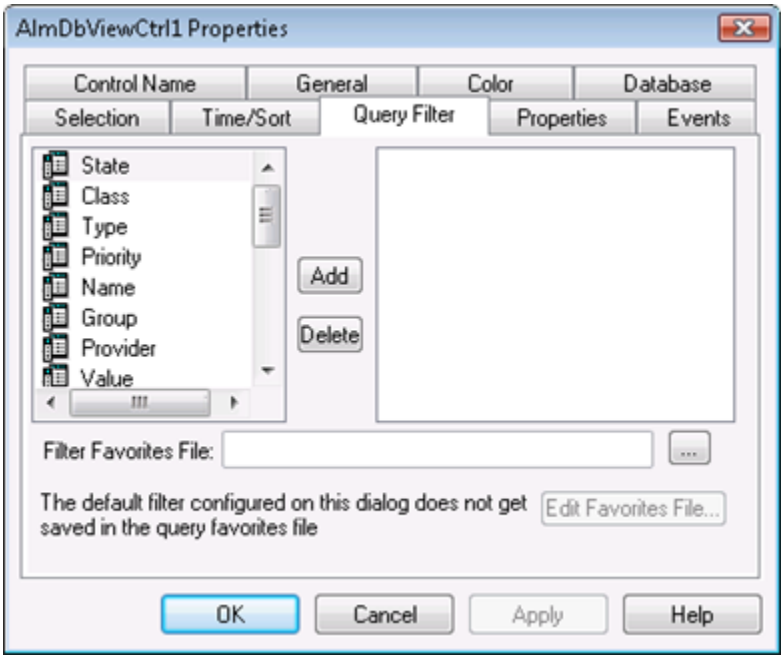


In this example, you create an alarm display that shows alarm data selected by the following criteria:

- Alarm priority (1-999)
- Alarm state (All, Acknowledged, or Unacknowledged)
- Query type (Summary or Historical)
- Alarm group (Local or remote data sources)

You can save your queries to an .xml file, called a "query favorites" file. During run time, you can update the alarm display with new alarm data by running another query using selection criteria saved to the file.

Other InTouch alarm ActiveX controls provide more extensive query criteria. The following figure shows the **Query Filter** tab of the Alarm DB Viewer control.



You build your queries by selecting alarm or event attributes from the list shown in the left pane of the dialog box. Then, you assign a value to the selected attributes. Finally, you can combine attributes using Boolean

operators to set your query filter conditions.

You can write QuickScripts that include query functions or dotfields to select alarm and event records from alarm memory. The following Alarm Viewer control statement uses the ApplyQuery() method to query the alarm memory.

```
#AlarmViewerCtrl1.ApplyQuery ("\InTouch!$System",500,600,"All", "Historical");
```

This statement retrieves all historical alarms specified by the "\InTouch!$System" query with a priority between 500 and 600. The selected alarm records appear in the Alarm Viewer control display.

Example alarm queries

Alarm queries follow this syntax for the local node:

\Provider!AlarmGroup

For example:

\InTouch!$System

Use the following query syntax for remote nodes:

\\NodeName\Provider!AlarmGroup

For example, on a node called MyNode1:

\\MyNode1\InTouch!$System

Use the following syntax for querying alarms from a Galaxy with the **Register using "Galaxy_<Galaxy name>" instead of "Galaxy"** checkbox selected. This syntax gets alarms from a specific alarm name of an object in a specific area on a specific computer. The alarm name may be an attribute name or an alarm primitive name. Galaxy names are displayed in the alarm window's Provider column.

\\NodeName\Galaxy_GalaxyName!AreaName!ObjectName.AlarmName

The following syntax gets all alarms from a specific area:

\Galaxy_GalaxyName!AreaName

The following syntax gets alarms from two areas:

\Galaxy_GalaxyName!Area1 \Galaxy_GalaxyName!Area2

The following syntax gets all alarms in the specified Area from the Platform on the specified computer node (by default):

\\NodeName\Galaxy_GalaxyName!AreaName

You can also use single wildcard to match alarm names within a specified area. The following syntax gets all alarms from all objects starting with the characters "Tank" in the "AreaName" area:

\Galaxy_GalaxyName!AreaName!Tank*

The following syntax gets all alarms named "Hi" from all objects in the "AreaName" area:

\Galaxy_GalaxyName!AreaName!*.Hi

Get more InTouch query information

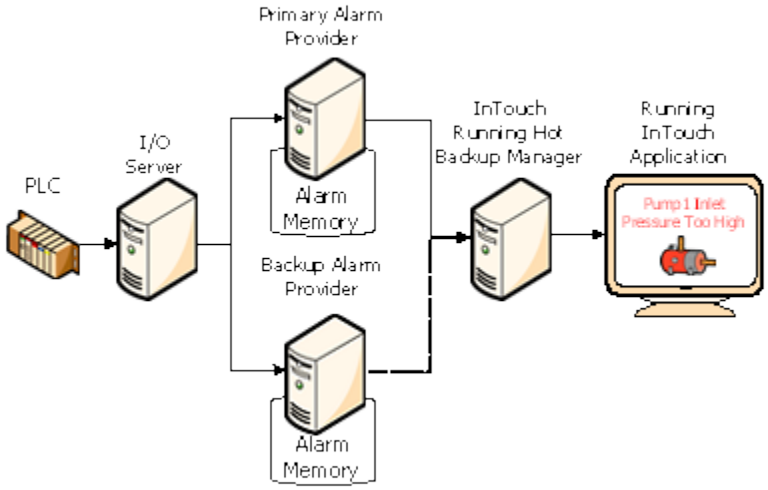
The following table lists the references for more information about queries for each InTouch query source.

Query Source	See
Alarm DB Logger Manager	Configure which alarms to log
Alarm Printer Utility	Configure which alarms to print
Alarm Viewer Control	Configure which alarms to show
Alarm Tree Viewer Control	Configure which providers and groups to show
Alarm Pareto Control	Configure which alarms to analyze
Distributed Alarm Display Object	Configure which alarms to show
Hot Backup Manager	Create an alarm record mapping file

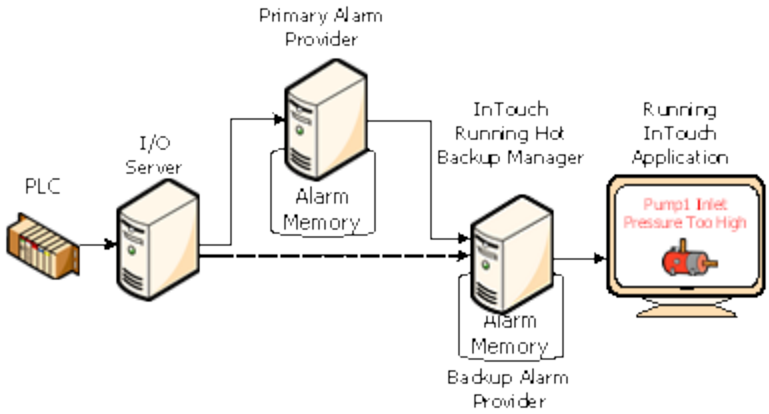
Enhance plant security through alarm redundancy

The InTouch Distributed Alarm system issues notifications and receives alarm acknowledgments from applications running on remote nodes within a network. Alarm provider applications store alarm data within their memory. Alarm consumer applications run as clients on other nodes to remotely query, show, and acknowledge alarms from the alarm providers.

You use the Alarm Hot Backup Manager to create a duplicate alarm provider. The following figure shows how the Hot Backup Manager uses a secondary current alarm repository as a backup provider.



You can also run the Hot Backup Manager and the backup provider on the same node, as shown in the following figure:



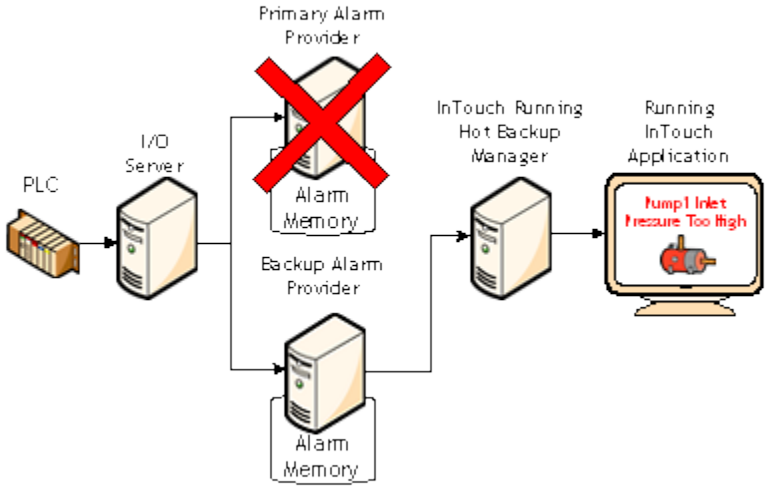
Understand hot backups

The hot backup provides a single name (hot backup pair name) that points to two alarm providers, the primary and the backup (the hot backup pair). The InTouch HMI alarm consumers can reference this single hot backup pair name and retrieve alarms from either the primary or the backup alarm provider. The following alarm clients support querying the hot backup pair:

- InTouch HMI Alarm DB Logger Manager
- InTouch HMI Alarm Printer
- InTouch HMI Alarm Viewer Control
- Alarm Client Control

If both provider nodes are operating normally, the alarm consumer receives alarm data from the primary provider. If the primary provider fails, however, the alarm consumer receives alarm data from the backup instead. You can also configure the Hot Backup Manager to continue using the backup provider even when the primary provider is available. If this backup option is not selected then, when the primary provider is available the Backup manager will switch to the primary provider automatically. For more information, see Create a hot backup pair.

The following figure shows how the alarm consumer still receives alarms after the primary alarm provider fails. The alarm consumer still references the hot back pair, but the backup provider provides the alarm data.



Specify the alarm providers for hot backup

You can specify any of the following alarm providers:

- InTouch

  When you specify InTouch as the alarm provider, the provider must be InTouch for both the primary node and the backup node. However, the alarm group of the backup node can be different from that of the primary node.
- Galaxy

  To specify a Galaxy or Galaxy_<GalaxyName> alarm provider, you must configure AppEngine redundancy in the IDE.
- Galaxy_<GalaxyName>

  The valid characters for GalaxyName are alphanumeric and special characters $, #, and _.

  When specifying Galaxy_<GalaxyName>, the Galaxy name must be the same for both the primary and the backup node.

  When you specify Galaxy or Galaxy_<GalaxyName> as the alarm provider, the provider of the backup node can be Galaxy or Galaxy_<GalaxyName>.

  The alarm group of the backup node must be the same as that of primary node.

Note: The system does not support hot backup pairs for alarms generated from two different Galaxies.

About the Hot Backup Manager

The Hot Backup Manager synchronizes alarm acknowledgements between the primary and backup providers. If an alarm is acknowledged on the primary provider, the same alarm is acknowledged simultaneously on the backup provider.

The Hot Backup Manager:

- Provides a configuration utility to create a backup pair.
- Provides a configuration utility to map alarm records between the primary and backup providers of a hot backup pair.
- Synchronizes alarm acknowledgments between InTouch alarm provider backup pairs.
- Establishes communication between all nodes that belong to a Hot Backup system when the Distributed Alarm system starts and stops.

Use the Alarm Hot Backup Manager in TSE sessions

You can start only one Alarm Hot Backup Manager using **Tools, Applications** in a different Terminal Services Edition (TSE) session of WindowMaker.

You can start one Alarm Hot Backup Manager in each TSE session using the **Start, Programs** shortcut. The last saved configurations will overwrite earlier configurations.

Querying the alarm hot backup pair is supported in a TSE session at run time.

Configure a hot backup pair

The Alarm Hot Backup Manager creates a backup pair from two host nodes running provider applications. You can start the Hot Backup Manager from WindowMaker. To configure a hot backup pair:

- Create a hot backup pair.
- Set key fields for alarm records.
- Map alarm record key fields.
- Import the alarm record map to Hot Backup Manager.

Create a hot backup pair

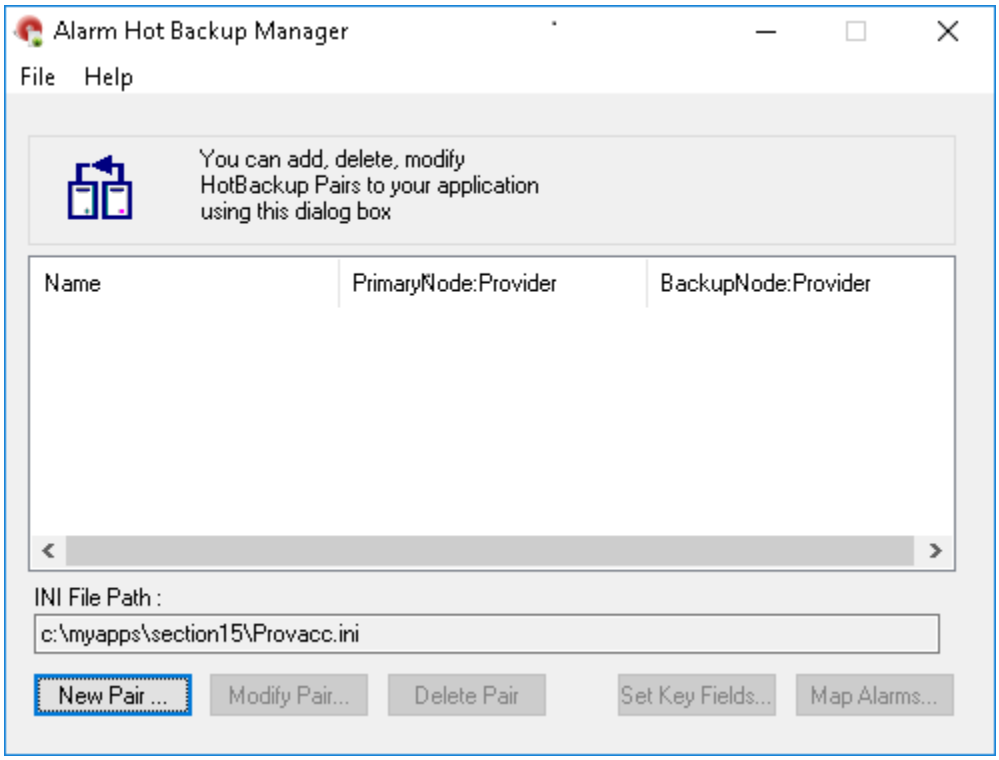
To create a hot backup pair, you:

- Assign a name to the hot backup pair.
- Identify the primary alarm provider.
- Identify the backup alarm provider.

You can also specify a Provacc.ini file that contains the configuration information.

Configure a hot backup pair

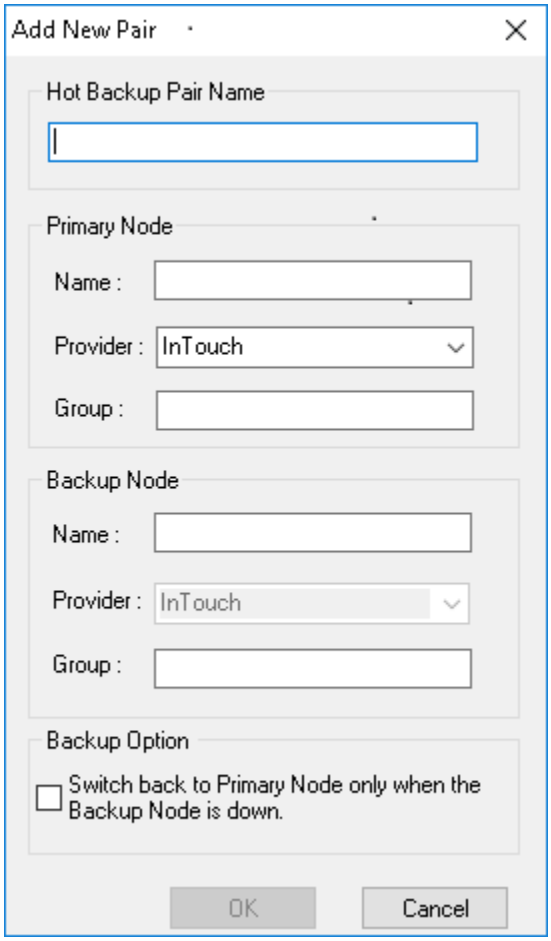
1. Open the Alarm Hot Backup Manager. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm Hot Backup Manager**.



2. On the **File** menu, select **Open**. Highlight the Provacc.ini file and then select **OK**.

   By default, the Alarm Hot Backup Manager checks for the Provacc.ini file in the last opened InTouch application folder. You should use the Provacc.ini file located in the InTouch application's folder. Otherwise, you can create a copy of the Provacc.ini file in another specified folder location and then select it for use with Hot Backup Manager.

3. Select **New Pair**. The **Add New Pair** dialog box appears.



4. In the **Hot Backup Pair Name** box, type a unique name for the new backup pair.

   A pair name can be 32 alphanumeric characters or less. You can use the dollar sign ($), pound sign (#), and underscore (_) character in a pair name.

5. In the **Primary Node** area, configure the primary node. Do the following:
   a. In the **Name** box, type the node name of the computer running the primary provider application. The node name must be unique to Hot Backup Manager. An error message appears if you enter a non-existent node name or the node name is used in another hot backup pair.
   b. In the **Provider** box, select the alarm provider from the drop-down list. The default is InTouch.
   c. In the **Group** box, type the name of the alarm group that queries alarms from the primary provider.

6. In the **Backup Node** area, configure the backup node. Do the following:
   a. In the **Name** box, type the node name of the computer running the backup provider application. This can be the same node that is running the Hot Backup Manager.
   b. In the **Provider** box, select the backup provider, if you specified a Galaxy or Galaxy_<GalaxyName> alarm

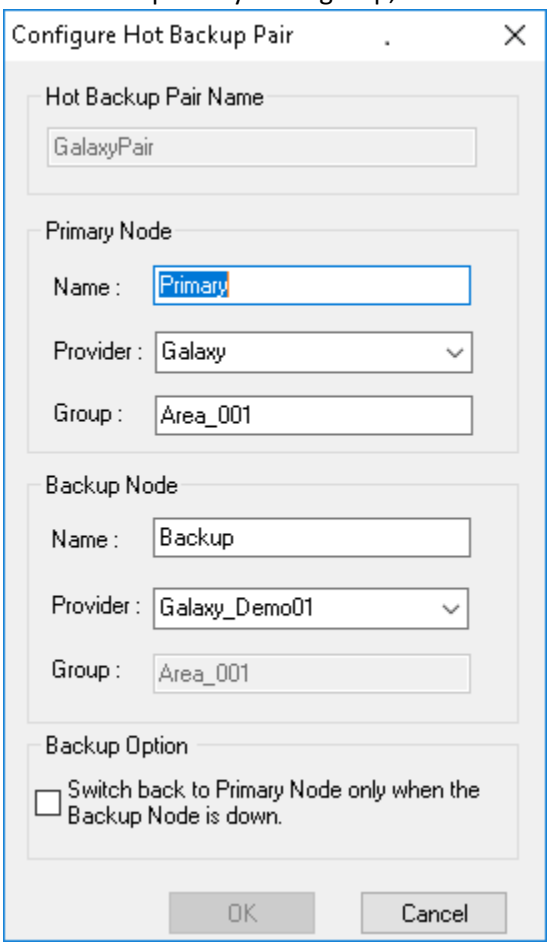
provider.

If you specified Galaxy or Galaxy_<GalaxyName> as the primary node alarm provider, the backup node provider must be Galaxy or Galaxy_<GalaxyName>.

If you specified InTouch as the primary node alarm provider, the backup node provider must be InTouch.

c. In the **Group** box, type the name of the alarm group that queries alarms from the backup provider.

If you specified Galaxy or Galaxy_<GalaxyName> alarm providers, the backup node group must be the same as the primary node group, and cannot be edited.



7. To continue using the backup node even when the primary node is available, select **Switch back to Primary Node only when the Backup Node is down**. (This option has not been tested on a non-English language Operating System.)

   By default this option is not selected.

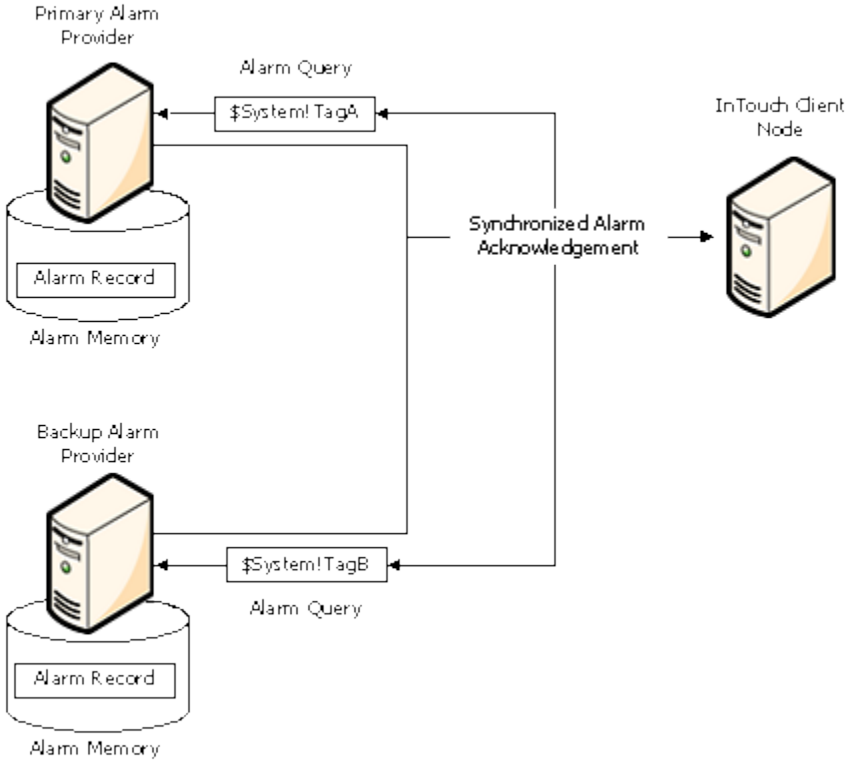
8. Select **OK**.
9. On the **File** menu, select **Save**.
10. Restart WindowMaker.

Set alarm key fields for a hot backup pair

To synchronize alarm acknowledgements between the primary and backup providers, you must identify a combination of tag alarm record fields. This combination of fields generates a unique mapping key to the paired alarm records stored in each provider's current alarm repository.

You can set alarm key fields and map alarms only for InTouch alarm providers.

The figure below shows a synchronized alarm acknowledgement request based on alarm record fields in a standard query.



A mapping key can be a combination of design-time and run-time alarm records. Design-time alarm records are based upon alarm properties of a tag when it is defined from the Tagname Dictionary.

For example, an alarm name field is known at design time because it takes on the name of the tag defined in the primary and backup node applications. QuickScripts or operators actions define or modify alarm properties stored as records while an application is running.

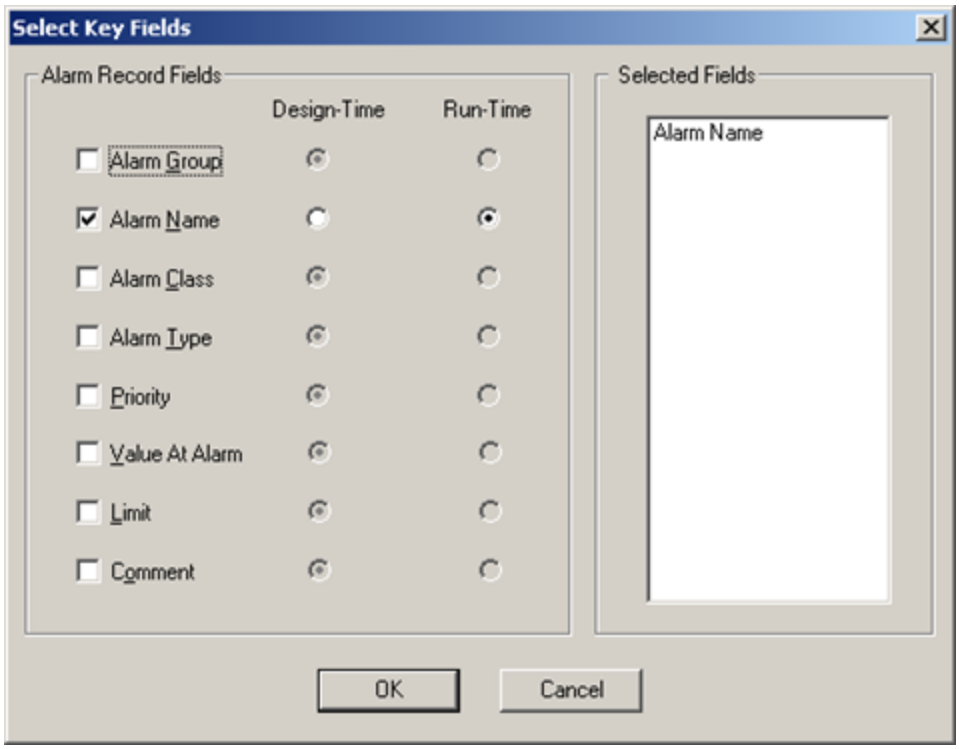
You can create a map key from any combination of design-time or run-time alarm record fields. The map key must select only a single record from each provider's current alarm repository. The key field must create a unique query.

You use the Hot Backup Manager to create a mapping key list from alarm record fields.

Note: When you specify a Galaxy or Galaxy_<GalaxyName> alarm provider, both the **Set KeyFields** and the **Map Alarms** options are disabled.

Create an alarm field mapping list for a hot backup pair

1. Open the Alarm Hot Backup Manager. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm Hot Backup Manager**.
2. Select a hot backup pair from the list.
3. Select **Set Key Fields**. The **Select Key Fields** dialog box appears.



4. In the **Alarm Record Fields** area, select the alarm record fields that you want to include in the mapping key list.

   The selected alarm record fields appear in the **Selected Fields** list box.
5. Select **Design-Time** or **Run-Time** for the alarm record fields you selected.
6. Select **OK**.
7. Restart WindowMaker.

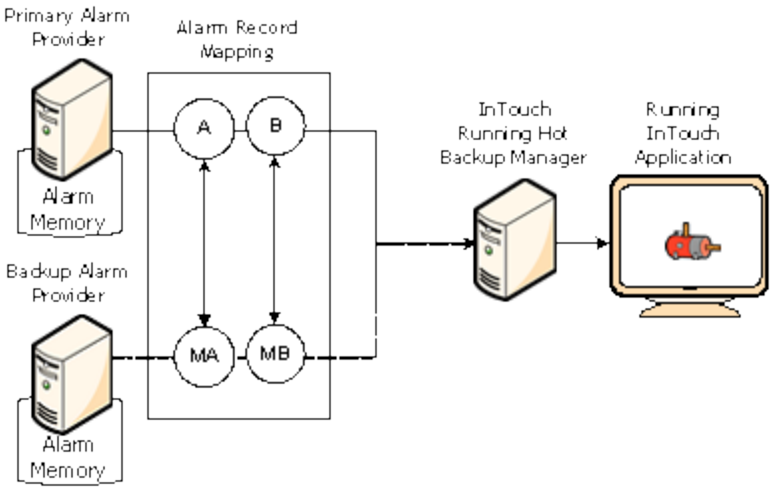
Create an alarm record mapping file

You must map the alarm records of a hot backup pair whenever the primary and backup alarm providers are running different applications. Alarm record mapping establishes a correspondence between dissimilar alarm records of the primary and backup providers. For example, you can map alarm records based upon their assigned InTouch tag names. Although their names may be different, the alarm records are logically consistent between the two provider alarm repositories.

Note: Creating an alarm record mapping file is unnecessary if both the primary and backup providers are running the same application. If no mapping file is provided, the Distributed Alarm system assumes the primary and the backup providers are running the same applications with the same alarm records.

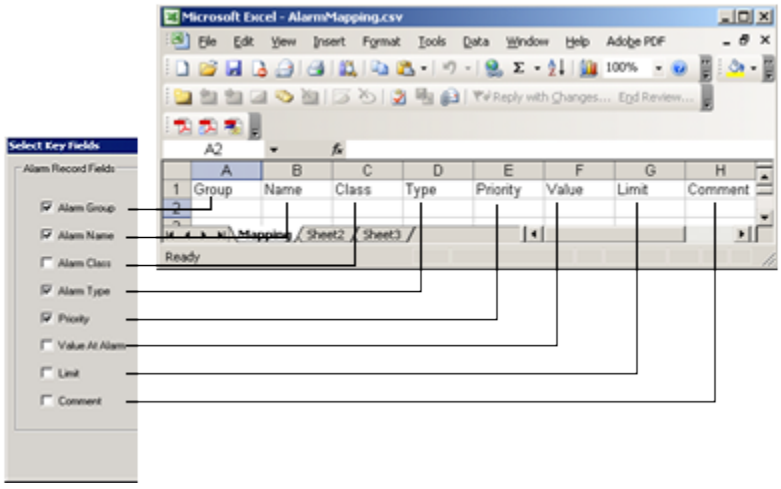
Mapping enables alarm acknowledgements to be synchronized between providers running different applications. When the Distributed Alarm system acknowledges an alarm on a provider, it also knows which alarm to acknowledge on the other provider.

The following figure shows alarm record mapping between the two providers of a hot backup pair. In this example, the A and B alarm records of the primary provider are mapped to the corresponding alarm records of the backup provider, MA and MB.



The Hot Backup Manager imports the alarm record map from a comma separated values (CSV) file that you create with Microsoft Excel or a text editor like Notepad. The mapping file includes an ordered list of alarm record fields that associate the corresponding alarm records of the primary and backup providers.

You must specify tag alarm record fields as the headers of the mapping file. The order of the headers within the file must match the alarm record fields shown from the **Select Key Fields** dialog box. The figure below shows the column headers of an Excel file that match the order of alarm record fields of the **Select Key Fields** dialog box.



You can create a mapping file that only includes the selected headers of alarm field records used to generate mapping keys. The figure below shows an Excel file that includes only the Name, Class, and Type headers. When you add headers, their order must always match the order of alarm record fields of the **Select Key Fields** dialog box.

Specify the alarm field records of the primary provider in the left set of columns. Likewise, specify the same records of the backup provider in the right set of columns.

Mapping File Column Header	Values Assigned to Alarm Field Records
Group	Name of the alarm group in which the tag has been assigned. An alarm group name cannot contain a blank space.
Name	Name of the tag whose alarm records are mapped. A tag name cannot contain a blank space.
Class	Class of alarm assigned to the tag. Possible Class values are: • VALUE for a value alarm. • DEV for a deviation alarm. • ROC for a rate of change alarm. • DSC for a discrete alarm.
Type	Type of alarm condition associated with an alarm class. • LOLO, LO, HI, and HIHI for a value alarm • MinDev and MajDev for a deviation alarm • ROC for a rate of change alarm • DSC for a discrete alarm
Priority	Priority assigned to an alarm condition. Priority must be a number from 1 to 999.
Value	See the following notes.
Limit	See the following notes.
Comment	See the following notes.

Value, Limit, and Comment columns:

- The "Value" and "Limit" column values can be anything other than Null, when the "Class" or "Type" values for that particular record in that particular node are not known.
- The "Value" and "Limit" column values can accept only 1234567890.-+eE characters when the Class value for that particular record in that particular node is known as Value, Dev or ROC.
- The "Value" and "Limit" column values can accept only 1234567890.-+eE characters when the Type value for that particular record in that particular node is known as LOLO, LO, HI, HIHI, MinDev, MajDev or ROC.
- The "Value" and "Limit" column values can be anything other than Null, when any one of the "Class" or "Type" values for that particular record in that particular node is known as DSC.
- The Comment column values have no limitations.

- All records of the mapping file should be unique. The Hot Backup Manager skips any duplicate records during the import process. You can see details after the import process is completed.

You can combine the field values of alarm records - such as Group, name, and Priority - to generate a "composite mapping key" that uniquely identifies alarm records.

An InTouch Alarm Provider equates the "Name" field to the name of the tag that generated the alarm. Therefore, when given hot backup pair, a mapping key can be generated using the combination of the alarm group name and the tag name.

For example:

Provider Node	Backup Node
$System!TagA	$System!TagB

If a provider has a name field and comment field together as a unique field then the mapping key can be a combination of name and comment.

Provider Node	Backup Node
tagA!CommentA	tagB!CommentB

This could be true for any other field combination for a third provider.

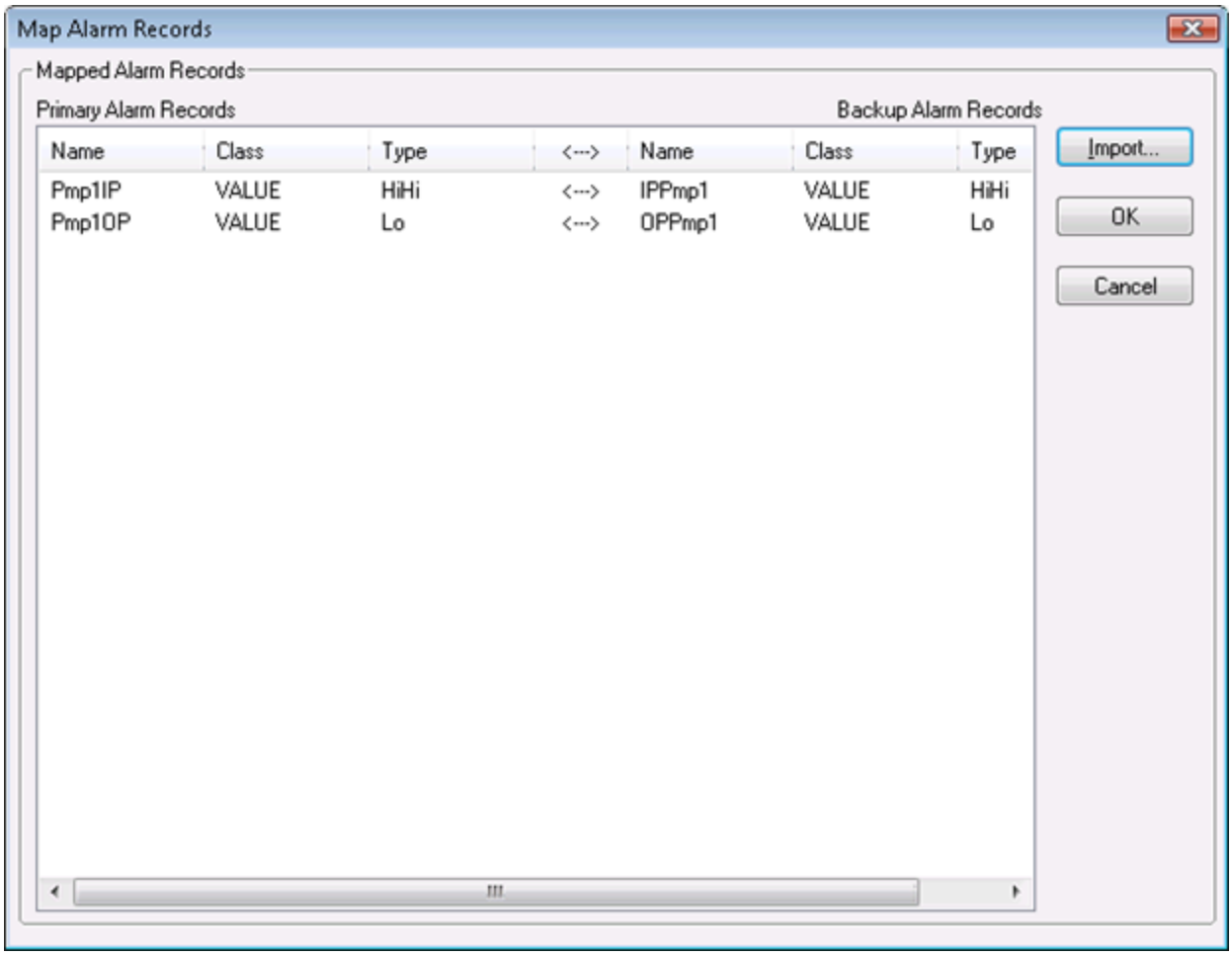
Import an alarm record mapping file

You can import the contents of the alarm record mapping file to the Hot Backup Manager.

No cross-validation between the Alarm Class and Alarm Type fields is done when you import the mapping file.

Import alarm records from a mapping file

1. Open the Alarm Hot Backup Manager. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm Hot Backup Manager**.
2. Select the hot backup pair from the list.
3. Select **Map Alarms**. The **Map Alarm Records** dialog box appears.



4. Select **Import**. The **Open** dialog box appears. Highlight the mapping file and select **Open**.

   The Hot Backup Manager begins importing records from the file.
5. Select **OK** after all mapping records have been imported.
6. On the **File** menu, select **Save**.
7. Restart WindowMaker.

Troubleshoot alarm mapping file import problems

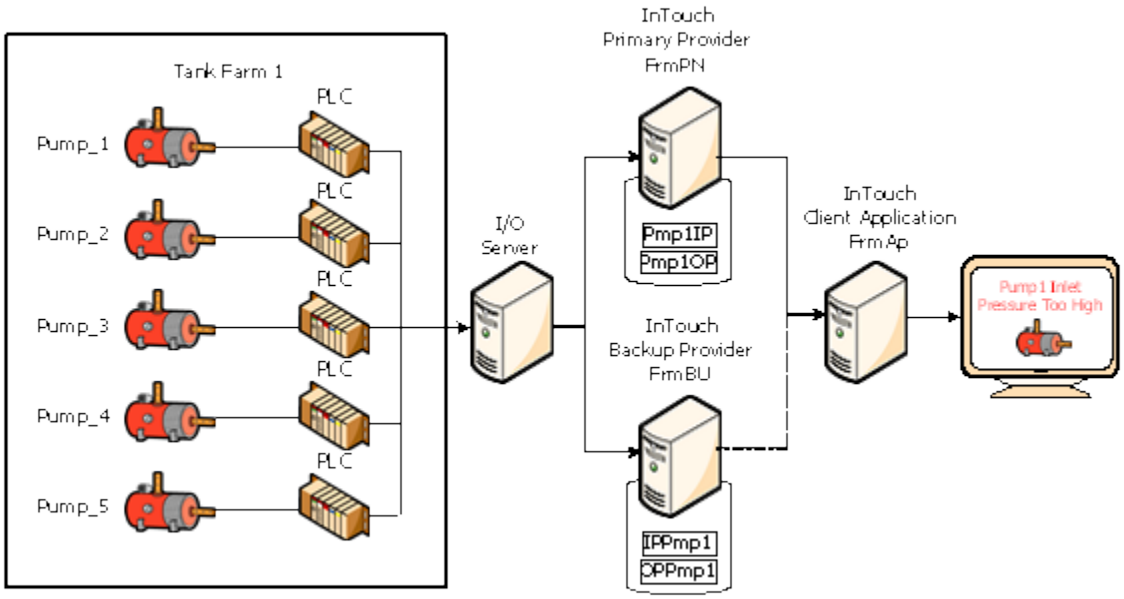
The following situations prevent a file from importing:

- The required number of columns should be filled with values for all the records at the import file. There should never be fewer values or more values for any record.
- The headings at the import file should be the same as that of the headings at the **Select Key Fields** dialog box and should be in the same order.

If a record that is imported has a wrong entry, you are prompted to skip that particular record number or to abort the importing process itself.

Example of a hot backup pair

This section describes a typical working scenario to set up an alarm backup pair. The figure below shows the configuration of a hot backup pair for an InTouch tank farm application. In this example, the InTouch application monitors pump pressure as an alarm condition.



All three computers are running the InTouch HMI. The hot backup pair includes FrmPN as the primary provider and FrmBU as the backup. These two nodes serve as alarm providers within an InTouch Distributed Alarm system.

The Hot Backup Manager run on the FrmAp node. The InTouch client application runs on FrmAp and consumes alarms from the two providers of the hot backup pair.

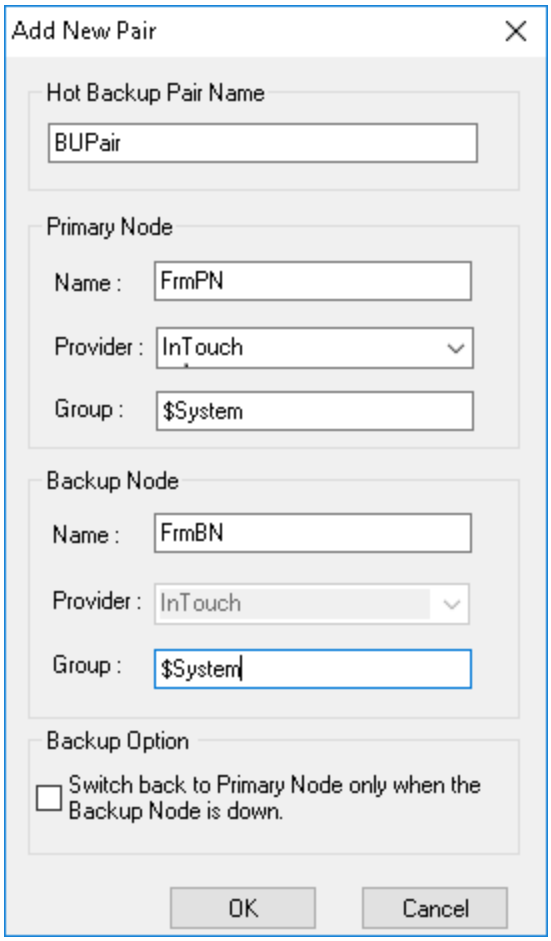
The InTouch application running on FrmPN generates two summary alarms for pump inlet and outlet pressure. The two alarms belong to the TnkFrm1 alarm group. The InTouch application running on FrmBU generates two logically equivalent alarms for pump pressure.

When you set up a redundant hot backup pair, you:

- Create a hot backup pair.
- Set the alarm record key fields.
- Create an alarm record mapping file.
- Import the alarm record mapping file.

Create a hot backup pair

1. Open the InTouch application in WindowMaker. In this example, the application runs on the FrmAp node.
2. Open the Alarm Hot Backup Manager. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm Hot Backup Manager**.
3. Select **New Pair**. The **Add New Pair** dialog box appears.
4. Complete the options of the **Add New Pair** dialog box, as shown in the following figure.



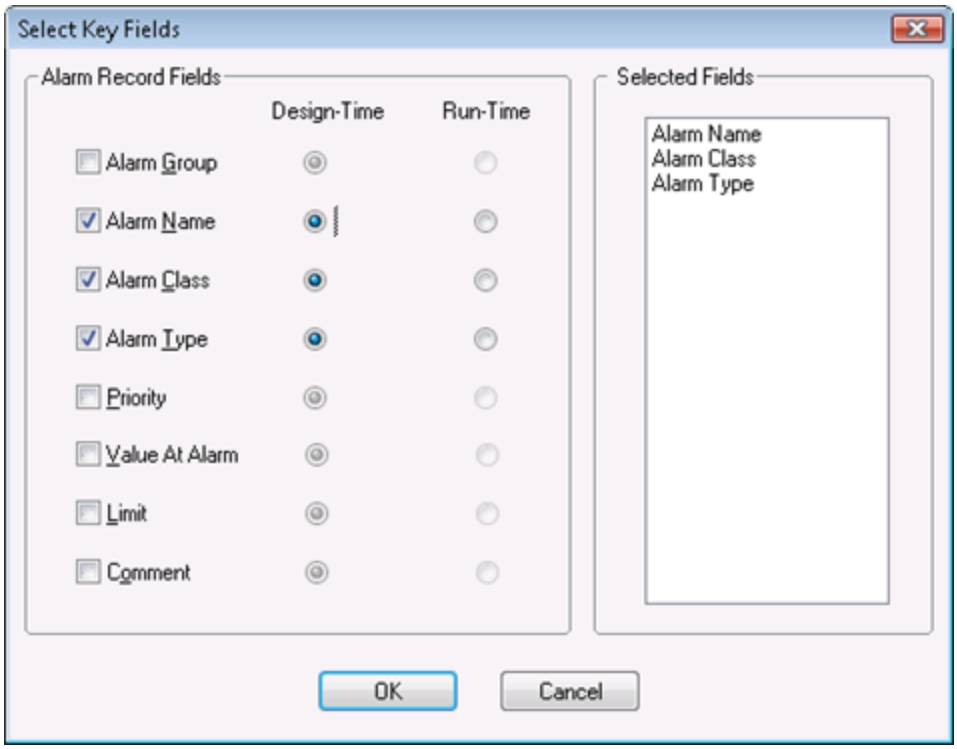
5. Select **OK** to return to the **Alarm Hot Backup Manager** dialog box.
6. Keep the **Alarm Hot Backup Manager** dialog box open within WindowMaker.

You are done with the first step to create a hot backup pair. Next, complete the following procedure to generate a unique mapping key to the paired alarm records stored in each provider's alarm repository.

Map alarm record key fields

1. Select the hot backup pair in the list.
2. Select **Set Key Fields**. The **Select Key Fields** dialog box appears.

   Complete the options of the **Select Key Fields** dialog box, as shown in the following figure. The alarms between the primary and backup provider are logically consistent but have been assigned different names and belong to different alarm groups. A unique mapping key to records stored in each provider's alarm repository can be generated by selecting **Alarm Group**, **Alarm Name**, **Alarm Class,** and **Alarm Type** as **Design-Time** options.



3. Select **OK**. When a message appears, select **Yes**.

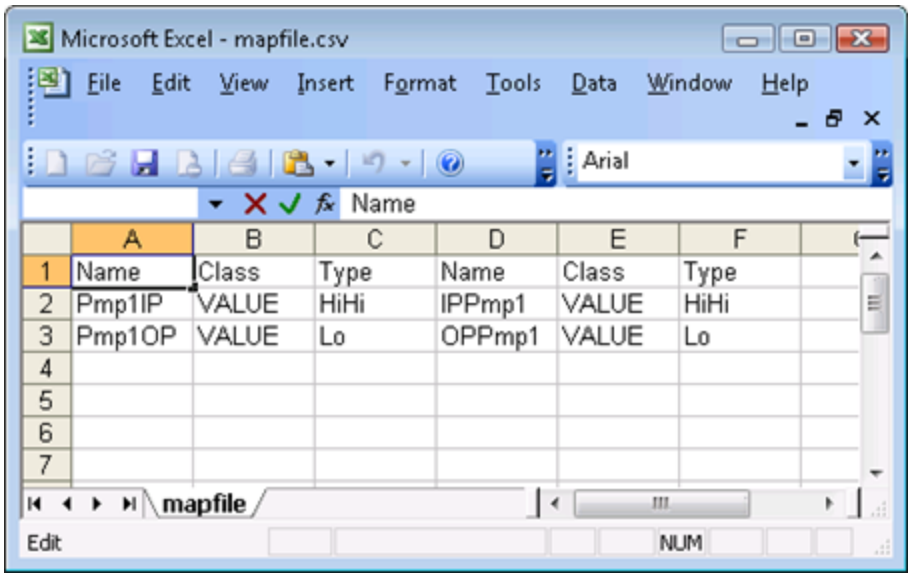
You are done with the second step to create an alarm record mapping key.

In this scenario all three nodes are running InTouch applications. The two provider nodes generate equivalent alarms, but are using different tagnames. The primary provider generates two summary alarms when a pump's inlet and outlet pressures are too high. The backup provider generates two logically identical alarms for the same pump pressure alarm conditions. Next, complete the following procedure to create a mapping file that associates equivalent records stored in each provider's alarm repository.

Create an alarm mapping file

1. Create a .csv file with Excel or a text editor like Notepad.
2. Enter the names of the file headers in the same order as the alarm record field options of the **Select Key Fields** dialog box.

   In this example, the file headers should be ordered by alarm group, alarm name, the class of alarm, and the type of alarm condition.
3. Map the alarms between the two providers on each row of the file.
4. The example of the Excel file below shows how the headers and alarm conditions should be specified for the two providers of the hot backup pair. Save the mapping file to a location accessible to the Hot Backup Manager running on the client node.



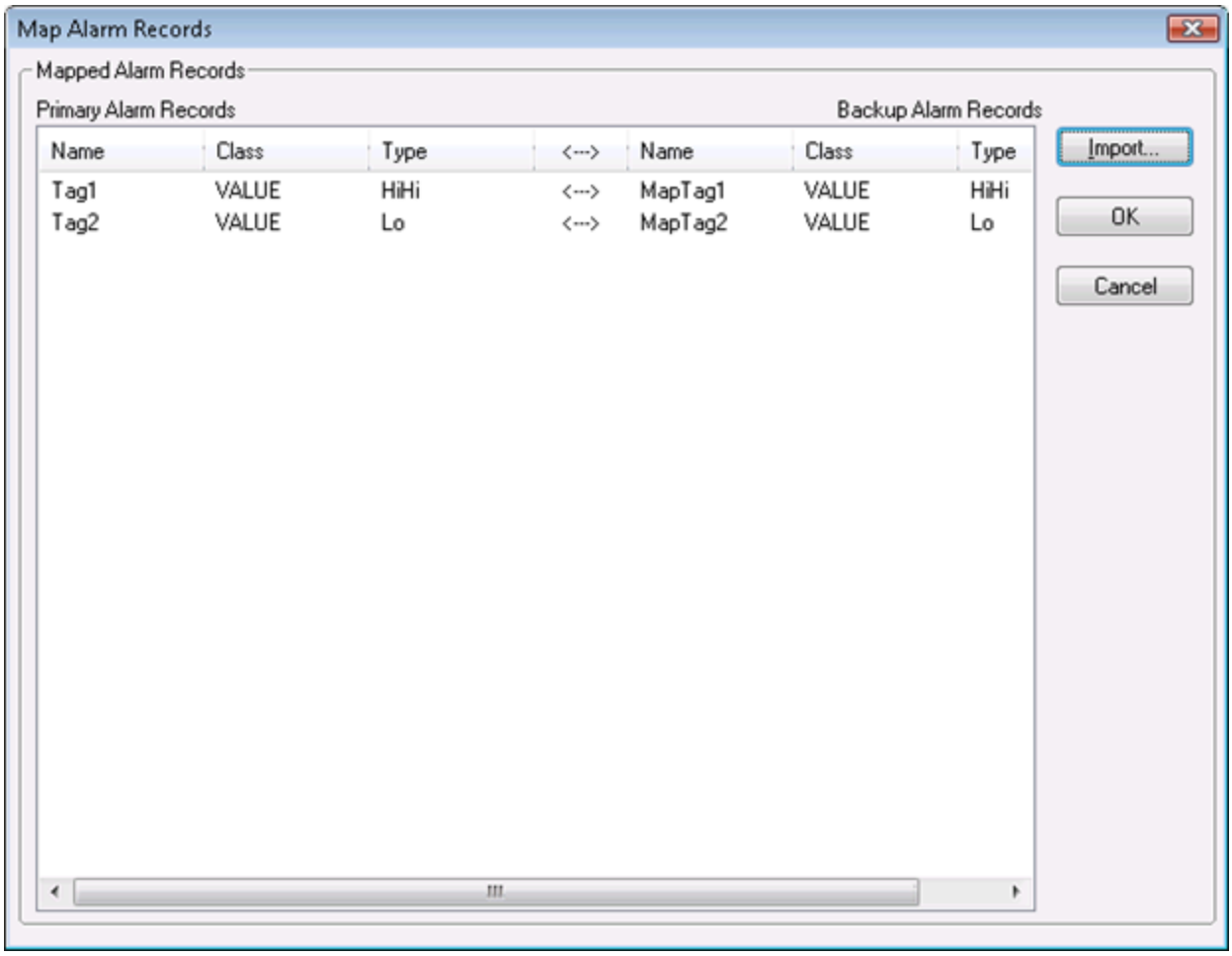
You are done with the third step to create an alarm record mapping file.

In the last step, you import the contents of the alarm mapping file to the Hot Backup Manager. In this example, the client application knows which pump pressure alarm records to acknowledge between the two alarm providers.

Import an alarm record mapping .csv file

1. Open the **Alarm Hot Backup Manager**.

   The alarm backup pair you created earlier should be listed.
2. Select **Map Alarms**. The **Map Alarm Records** dialog box appears.
3. Select **Import**. The **Open** dialog box appears.
4. Highlight your mapping file and select **Open**. The **Map Alarm Records** dialog box lists the alarm mapping records from the file.



5. Select **OK**. The Hot Backup Manager begins importing records from the file.
6. Select **OK** after all mapping records have been imported.

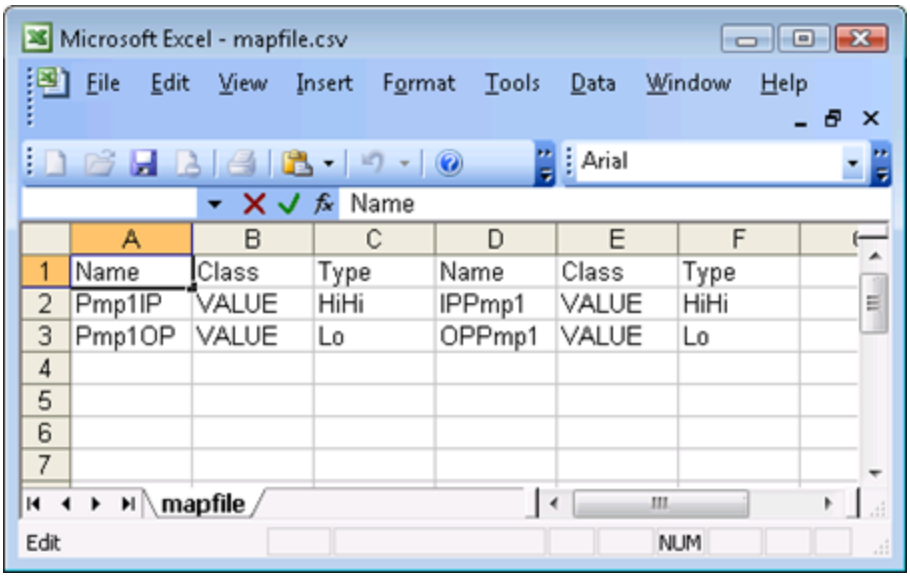
You can now run the hot backup application.

Acknowledgement synchronization example

For this example:

- Alarm Name, Alarm Class, and Alarm Type are selected as design-time key fields.
- Alarm Group is selected as a run-time field.

The following alarm record map file was created using Microsoft Excel.



The Pmp1IP alarm is mapped to the IPPmp1 alarm. Both have a Class of VALUE and a Type of HIHI.

The Pmp1OP alarm is mapped to the OPPmp1 alarm. Both have a Class of VALUE and a Type of Lo.

- When the HiHi alarm for Pmp1IP is acknowledged at the primary alarm node, the acknowledgment also appears on the HiHi alarm for IPPmp1 at the secondary provider node, provided that the alarm group names on both nodes remain the same.
- When the Low alarm for Pmp1OP is acknowledged at the primary alarm node, the acknowledgment also appears on the Low alarm for OPPmp1 at the secondary provider node, provided that the alarm group names on both nodes remain the same.

Acknowledgement synchronization occurs only if the design-time and run-time mapping match.

You can select any combination of design-time and run-time alarm record fields for mapping. However, make sure that the mappings do not result in multiple references.

For example, if two alarm record fields such as Class and Priority are selected, it is very possible that more than one alarm matches the criteria. In such a case, the Hot Backup synchronization is not guaranteed to work. In the process of propagating the acknowledgment, a random alarm that matches the criteria may also be picked up, while other matching alarms may be left unacknowledged.

Notes regarding hot backup pairs

- You can use a hot backup only for InTouch 7.11 and later clients.
- If the Distributed Alarm Display object queries a hot backup pair and then queries the primary provider again separately, the Distributed Alarm Display object shows duplicate records.
- Do not configure a provider as a primary or secondary of more than one hot backup pair.
- If a record at the primary provider is acknowledged and the secondary provider (which was down when the acknowledgment took place) starts up at a later time, the time stamp of the acknowledged record is identical with the record in the primary provider.
- The alarm consumer querying a hot backup pair shows the individual node name as the provider.
- You can choose any combination of design-time and run-time alarm record fields for mapping. However, be sure that the mappings do not result in multiple references.

- When mapping the Value and Limit key fields, the values are rounded off to the fourth decimal place and then mapped.
- An alarm record that does not have the specific combination of design-time and run-time mapping uses the default runtime mapping.

Create an alarm audit trail

When you configure an InTouch alarm provider to use either operating system or ArchestrA authentication and an alarm occurs, the alarm display contains the full name of the operator in the Operator Full Name column, assuming the operator is logged on.

For example, if a user is registered in the PLANT_FLOOR domain with a user ID of JohnS and a full name of John Smith, the Operator Full Name column contains John Smith. If the alarm is subsequently acknowledged, and the node performing the acknowledgement is set to use operating system or ArchestrA security, the Operator Full Name column is updated to show the full name of the acknowledgement operator. Otherwise, the alarm display shows a computer name concatenated with whatever is in the $Operator tag.

InTouch security can include an operator’s full name with alarm acknowledgements. This is also possible on records pertaining to alarm detection. In most organizations, a logon ID is not a person's full name, but rather an abbreviation or role classification.

When you configure operating system authentication for the provider and consumer InTouch nodes:

- The alarm display shows full names when alarms are generated and when acknowledgements are performed.
- The alarm printer prints full names when alarms are generated and when acknowledgements are performed.
- The Alarm DB Logger records domain name, log on user ID, and full user name with each alarm record for both Operator and AckOperator fields. This allows for unique identification even if an organization has two employees with identical full names.
- The Operator field shows the user account in the DomainName\UserName format.

Acknowledge alarms that require authentication

Some industries require that an explicit "signature" or user authentication be performed when acknowledging certain types of alarms. InTouch takes advantage of the Alarm Client control or script functionality to perform this type of authentication operation. The alarm priority is the key factor used to determine if the alarm or alarms to be acknowledged require a specific user signature.

You can configure alarms to require authentication so that users can acknowledge them. Alarms configured with authentication required means that the run time user must enter his or her credentials in order to acknowledge the alarm or group of alarms. If the system is configured with a smart card reader, the user can enter smart card credentials.

This section describes how to acknowledge alarms in WindowViewer that require authentication.

For information about configuring the Alarm Client to require authentication for acknowledging alarms, see Alarm Client Control documentation

For information about using the SignedAlarmAck() function to configure authentication behavior for acknowledging alarms, see "Add and maintain graphic scripts" in *Industrial Graphic Editor User* documentation.

About acknowledging alarms in WindowViewer that require authentication

Acknowledging alarms configured for authentication requires the following:

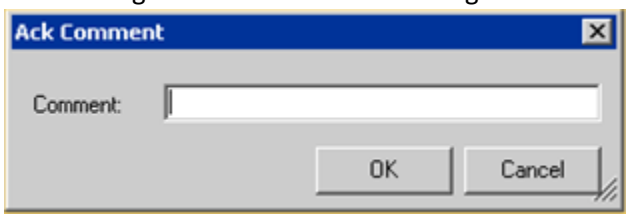
- Security must be enabled for the Galaxy.
- Security must be enabled for a deployed InTouchViewApp.
- The InTouchViewApp must include at least one Alarm Client control configured to display alarms and events and to require a signature (authentication) for alarm acknowledgement, and with minimum and maximum values for the alarm priority range.
- Run-time operators must have the appropriate user name, password, and operational permissions configured within the Galaxy: An operator must have sufficient privileges to run WindowViewer.

Regardless of who is currently logged on as the run-time user for the InTouch application, alarms configured to require a signature for alarm acknowledgement always require user authentication. This does not affect the session of the logged-on user.

Alarm acknowledgement basic run-time behavior

When the operator selects one or more alarms and attempts to acknowledge them:

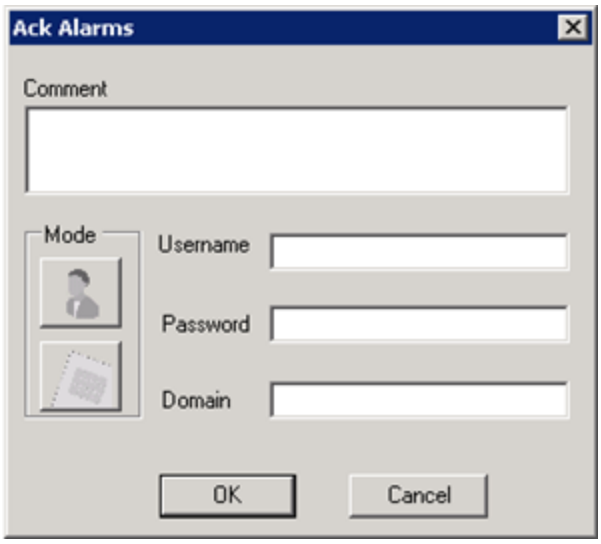
- The Alarm Client will check whether it has been configured to require acknowledgement signatures. If so, it checks whether any selected alarm falls within the configured priority range. If so, it requires a signature to acknowledge all the selected alarms.
- If no signature is required, a pop-up acknowledgement dialog displays. It may have a text box for an acknowledgement comment if so configured.



- If the operator credentials are valid, the Alarm Client attempts to acknowledge all of the selected alarms.
- In the Alarm Client grid, the signing user is identified as the person who acknowledged all of the selected alarms.
- The Alarm Client prefixes the text "Signed ACK" at the beginning of the acknowledgement comment to indicate that this was a signed acknowledgement.
- If none of the selected alarms had priority in the required range, the Alarm Client prefixes the text "Std ACK" at the beginning of the acknowledgement comment to indicate that this was a standard acknowledgement.

Acknowledge alarms that require authentication

1. In WindowViewer, select one or more unacknowledged alarms. Right-click and select the ack operation you wish to perform. If any of the alarms are configured to require a signature, the **Ack Alarms** dialog box appears.



If Smart Cards are not enabled, the **Mode** buttons for selecting either Smart Card or Password authentication are disabled.

2. Add a comment in the **Comment** box if configured for commenting.
3. If you are authenticating using a network user account, the user account options are shown.

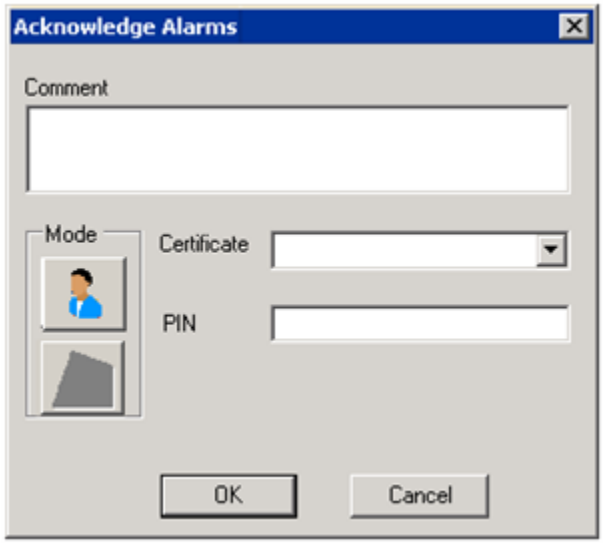
   Do the following.

   a. In the **Username** box, type your user name. The name of the currently logged-on user is shown by default. If no user is currently logged on, the box is blank.

   b. In the **Password** box, type the password associated with the user name.

   c. In the **Domain** box, type the domain name. If the security mode is ArchestrA Galaxy, then the domain displayed is ArchestrA and you cannot modify it.

   d. Select **OK**.
4. If you are authenticating using a Smart Card, the Smart Card **Ack Alarms** dialog box appears.



Do the following to authenticate using a Smart Card.

a. In the **Certificate** list, select your Smart Card certificate. The certificate list appears as domain_name/user_name. For a certificate to appear in the list, Smart Card must be currently inserted into a reader

attached to the computer. The certificate of the currently logged-on user is shown by default. If you insert or remove a card while the **Secured Write** dialog box is open, the certificate list automatically updates.

b. In the **PIN** box, enter the PIN for the Smart Card being used.

c. Select **OK**.

To authenticate using your name, password, and domain instead, select the Mode button. Go to Step 3.

Work with the Distributed Alarm Display object

The Distributed Alarm Display object is included in this version of the InTouch HMI to support applications developed with InTouch version 7 and earlier. As the Distributed Alarm Display object is a legacy object, you should use the Alarm Viewer Control instead to create alarm displays with more recent versions of the InTouch HMI.

The Distributed Alarm Display object is included in this version of the InTouch HMI to support applications developed with InTouch version 7 and earlier. As the Distributed Alarm Display object is a legacy object, you should use the Alarm Viewer Control instead to create alarm displays with more recent versions of the InTouch HMI.

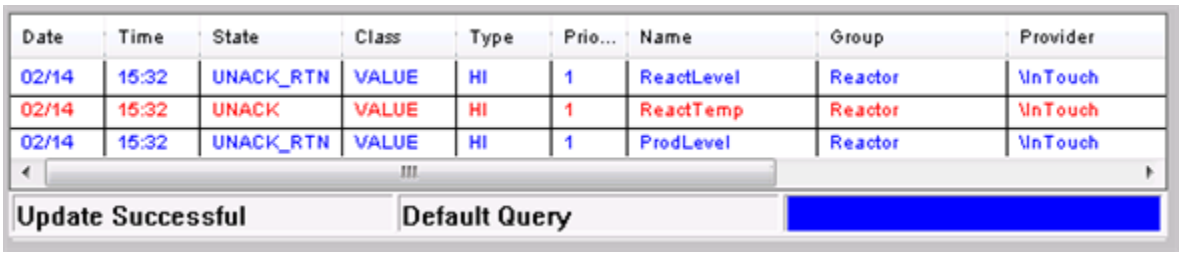
About working with the Distributed AlarmDisplay object

The Distributed Alarm Display object is included in this version of the InTouch HMI to support applications developed with InTouch version 7 and earlier. As the Distributed Alarm Display object is a legacy object, you should use the Alarm Viewer Control instead to create alarm displays with more recent versions of the InTouch HMI.

The Distributed Alarm Display object is included in this version of the InTouch HMI to support applications developed with InTouch version 7 and earlier. As the Distributed Alarm Display object is a legacy object, you should use the Alarm Viewer Control instead to create alarm displays with more recent versions of the InTouch HMI.

About the Distributed Alarm Display object

The Distributed Alarm Display object provides a single display to show both local and remote alarms.



This display object's features include built in scroll bars, sizable columns, multiple selection of alarms, a status bar, a shortcut menu, and colors based on alarm priority and state.

The Distributed Alarm Display object includes properties that can set the appearance of the alarm display (including the information that is shown), the colors used for various alarm conditions, and which alarm group and alarm priority levels are shown.

For more information about the display object, see Use a Distributed Alarm Display object at run time.

Distributed Alarm Display object guidelines

Observe the following guidelines when using the Distributed Alarm Display object:

- Each display must have an identifier so that any associated QuickScript functions knows which display to modify. This identifier, entered in the **Display Name** box in the **Alarm Configuration** dialog box, must be unique for each display.
- Displays should not overlap other InTouch objects, such as window controls or graphic objects. You can easily verify this by selecting the Distributed Alarm Display object in WindowMaker, and checking the display's "handles." The handles should not touch other objects on the screen.
- Displays should be used sparingly. Placing numerous displays on one screen can result in reduced system performance. When possible, limit the number of displays on your window and call further windows with additional displays.

Configure a Distributed Alarm Display object at design time

You can configure:

- General features, such as a status bar, scroll bars, and so on.
- The columns and sorting order.
- The alarm query to use to retrieve alarm records.
- The time format the alarm records are shown in.
- The font and colors for the shown alarm records.
- What run-time users can do in the display, such as resize columns, select alarms, access a shortcut menu, and so on.

Create a Distributed Alarm Display object

You create a Distributed Alarm Display object just as you would a wizard.

Create a Distributed Alarm Display object

1. On the **Draw** menu, in the **Insert** group, select **Wizards**.

   The **Wizard Selection** dialog box appears.
2. Select **Alarm Displays** in the list of wizards.
3. Double-click the **Dist. Alarm Display** wizard. The dialog box closes and your window reappears with the cursor in the "paste" mode.
4. Select the window to paste the Distributed Alarm Display object. To size the wizard, drag the selection handles.

You are now ready to configure the display.

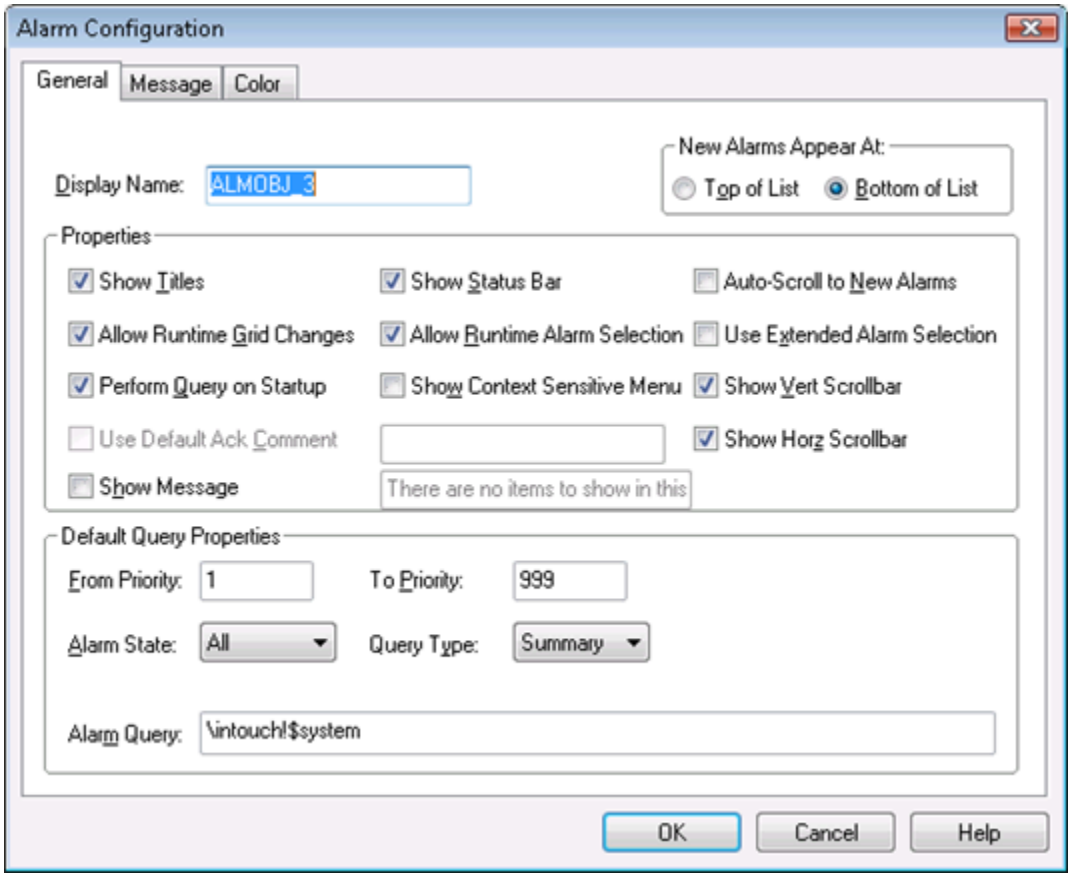
Configure the display properties of the grid

For the display grid, you can configure:

- A title bar, status bar, and scroll bars.
- Where new alarms appear in the grid and whether to auto-scroll to them.
- A default message to be shown if there are no alarm records to show.

Configure the appearance of the grid

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.



2. In the **Display Name** box, type the name for the alarm display. This name must be unique for each alarm display used. This name is used throughout the system for referring to this object for execution of tasks such as alarm acknowledgment and queries.
3. In the **New Alarms Appear At** area, configure where you want new alarms to appear in the object:
   - Select **Top of List** to show the most recent alarm at the top of the list.
   - Select **Bottom of List** to show most recent alarm at the bottom of the list.
4. In the **Properties** area, configure title bar, status bar, and scroll bars. Do any of the following:
   - Select the **Show Titles** checkbox to show the alarm message title bar.
   - Select the **Show Status Bar** checkbox to show the status bar.

- Select the **Show Vert Scrollbar** checkbox to show the vertical scroll bar.
- Select the **Show Horz Scrollbar** checkbox to show the horizontal scroll bar.

5. Select the **Show Message** checkbox to show a default message if there are no alarm records to show. Type the message in the box.
6. Select the **Auto-Scroll to New Alarms** checkbox to have the selection automatically jumps to the new alarm. New alarms are defined as those that are not currently shown within the display object.
7. Select **OK**.

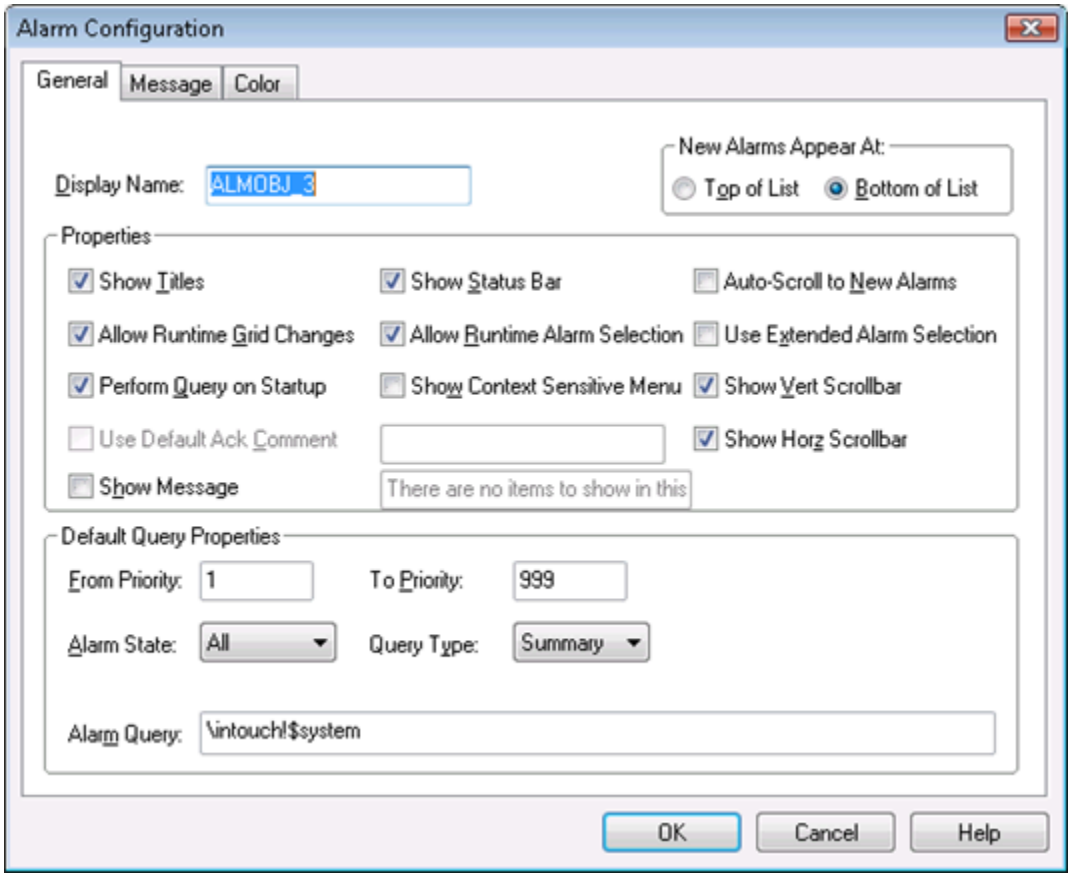
Control which features users can access at run time

You can allow the run-time user to change the column settings, select alarms, and open the shortcut menu.

Note: You can use script functions to run the commands that appear on the shortcut menu.

Configure the run-time features

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.



2. Configure the which options are available at run time. Do any of the following:
   - Select the **Allow Runtime Grid Changes** checkbox to allow the user to change column settings.
   - Select the **Show Context Sensitive Menu** checkbox to enable the shortcut menu.
   - Select the **Allow Runtime Alarm Selection** checkbox to allow the user to select alarms.
   - Select the **Use Extended Alarm Selection** checkbox to allow the user to select multiple alarms by holding

down Ctrl or Shift in conjunction with mouse selection. The default is to toggle selection of alarms by simply selecting them.

3. Select **OK**.

Configure which alarms to show

You can configure the Distributed Alarm Display object to show alarms based on:

- Priority.
- State, such as acknowledged or unacknowledged.
- Type, either summary or historical.

  When you configure the alarm query, you use text only. You cannot use tags. Example queries are as follows.

  Full path to Alarm Group:

  \\Node\InTouch!Group

Full path to local Alarm Group

\InTouch!Group

Another Group List:

GroupList

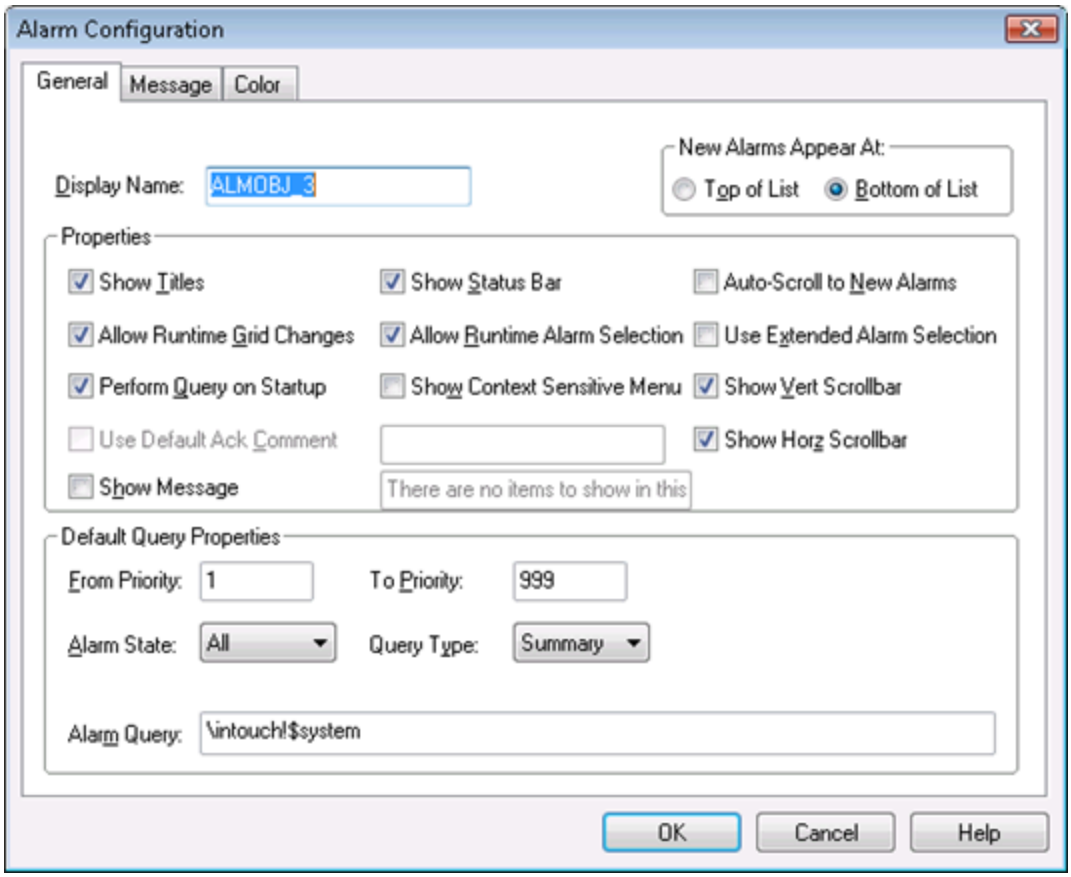
To perform multiple queries, separate each query with a space. For example:

\InTouch!Group GroupList

The default query properties are only used if you select the **Perform Query on Startup** checkbox or if the almDefQuery() function is executed.

Configure which alarms to show

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.



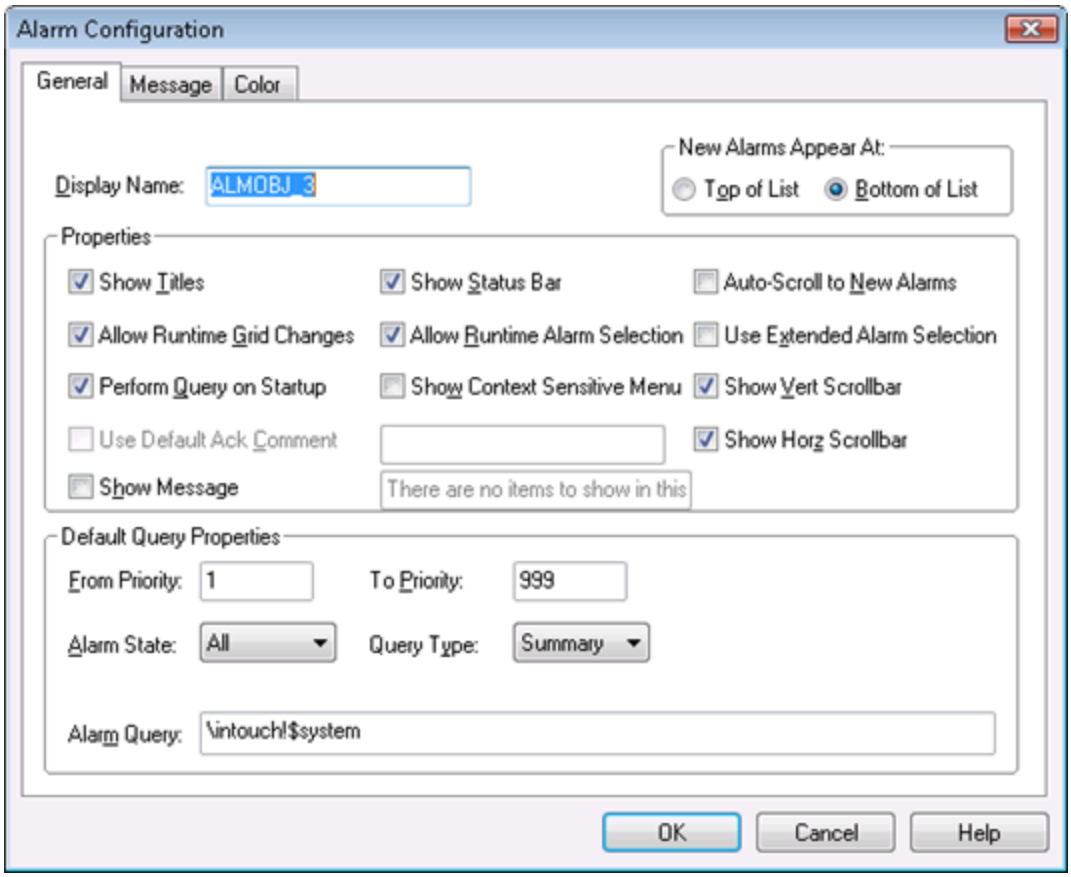
2. Select the **Perform Query on Startup** checkbox.
3. In the **Default Query Properties** area, configure the default query for the object.
   - In the **From Priority** box, type the default minimum alarm priority.
   - In the **To Priority** box, type the default maximum alarm priority.
   - In the **Alarm State** list, select the default alarm state to query (All, Unack, Ack).
   - In the **Query Type** list, select the alarm type, either Summary or Historical.
   - In the **Alarm Query** box, type the initial alarm query.
4. Select **OK**.

Configure a default alarm comment

You can configure a default comment to be used when an operator acknowledges an alarm. If you do not configure a default comment, when the operator acknowledges an alarm, a dialog box appears to let the operator enter a comment. The dialog box can be filled in or left blank.

Configure a default comment

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.



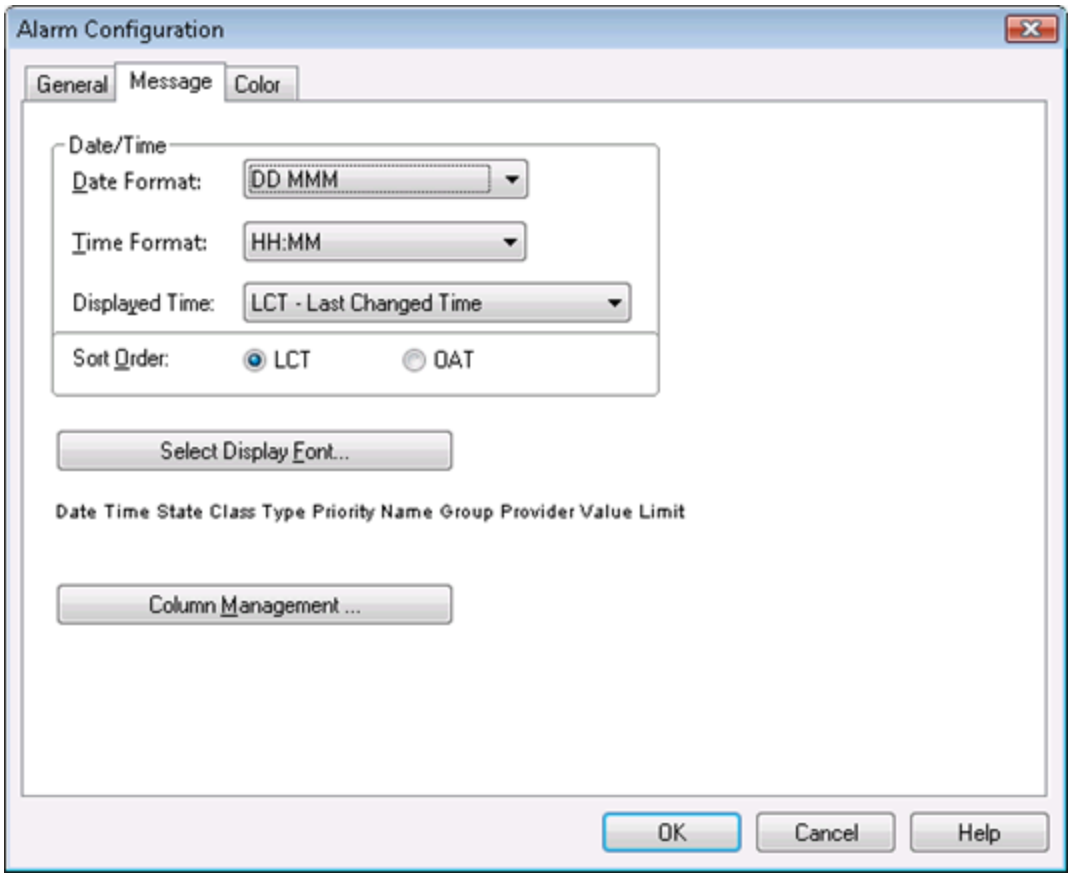
2. Select the **Show Context Sensitive Menu** checkbox.
3. Select the **Use Default Ack Comment** checkbox and then type the comment text in the box.
4. Select **OK**.

Configure the time format for alarm records

The original alarm time is the date/time stamp of the onset of the alarm. If tag is an I/O tag, then it is the time stamp from the I/O Server if that server is capable of passing time stamps.

Configure the alarm display time format

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.
2. Select the **Message** tab.



3. In the **Date Format** list, select the format for the date. Available formats are:

Selection	Shows	Selection	Shows
DD MMM	28 Feb	**MM/DD**	02/28
DD MM YYYY	28 Feb 2007	**MM/DD/YY**	02/28/07
DD/MM	28/07	**MMM DD**	Feb 28
DD/MM/YY	28/02/07	**MMM DD YYYY**	Feb 28 2007
YY/MM/DD	07/02/28	**YYYY/MM/DD**	2007/02/28

4. In the **Time Format** list, select the format for the time. Use the values in this list as a template to specify the format of the time. For example, to specify the time as 10:24:30 AM, use HH:MM:SS AP. The template characters are as follows:

Character	Description
AP	Selects the AM/PM format. For example, three o'clock in the afternoon is shown as 3:00 PM. A time without this designation defaults to 24 hour military time format. For example, three o'clock in the afternoon is shown as 15:00.
HH	Shows the hour the alarm/event occurred.
MM	Shows the minute the alarm/event occurred.
SS	Shows the second the alarm/event occurred.
SSS	Shows the millisecond the alarm/event occurred.

5. In the **Sort Order** area, configure the order in which you want the alarms to be sorted in the object:
   - Select **OAT** to use the original alarm time, which is the date/time stamp of the onset of the alarm.
   - Select **LCT** to use the last alarm change time, which is the date/time stamp of the most recent change of status for the instance of the alarm: onset of the alarm, change of sub-state, return to normal, or acknowledgment.
6. Select **OK**.

Configure the font for alarm records

You can set the font for the records and the heading in the control.

Configure the font properties

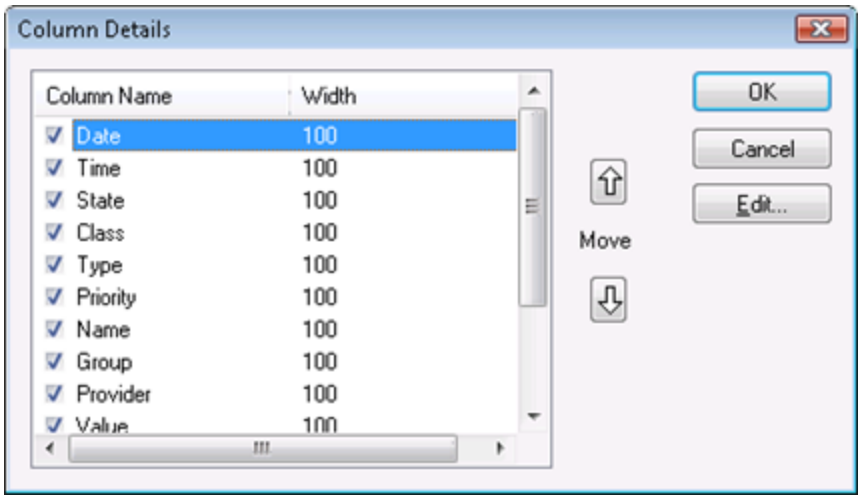
1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.
2. Select the **Message** tab.
3. Choose **Select Display Font**. The standard Windows **Font** dialog box appears.
4. Configure the font and then select **OK**.

Configure the columns for alarm records

You can select the columns to show, specify the column order, and set the column names and widths.

Configure the display column details

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.
2. Select the **Message** tab.
3. Select **Column Management**. The **Column Details** dialog box appears.



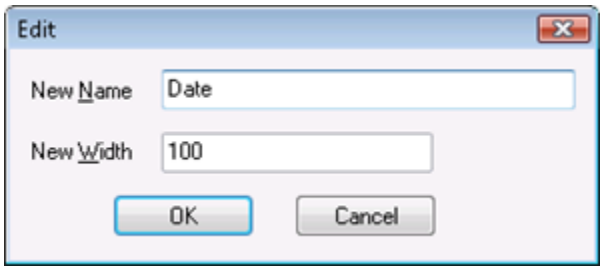
4. Select the checkbox in the **Column Name** list to show that column in the Distributed Alarm Display object. You must select at least one column. The following table describes the columns:

Column	Description
Date	Shows the date in the format selected from the **Message** tab.
Time	Shows the time in the format selected from the **Message** tab.
State	Shows the state of the alarm.
Class	Shows the category of the alarm.
Type	Shows the alarm type.
Priority	Shows the alarm priority.
Name	Shows the alarm/tagname.
Group	Shows the Alarm Group name.
Provider	Shows the name of the alarm provider.
Value	Shows the value of the tagname when the alarm occurred.
Limit	Shows the alarm limit value of the tagname.
Operator	Shows the logged-on operator's ID associated with the alarm condition.
Comment	Shows the tagname comment. This comment is typed in the **Alarm Comment** box when the tagname's alarm was defined in the database.

5. To rearrange the columns, select the column name and use the Move Up and Down arrow buttons. The

column name appearing at the top of the **Column Details** dialog box is the column shown to the furthest left of the alarm display.

6. To edit the column name and width, select a column name and then select **Edit**. The **Edit** dialog box appears for that column.



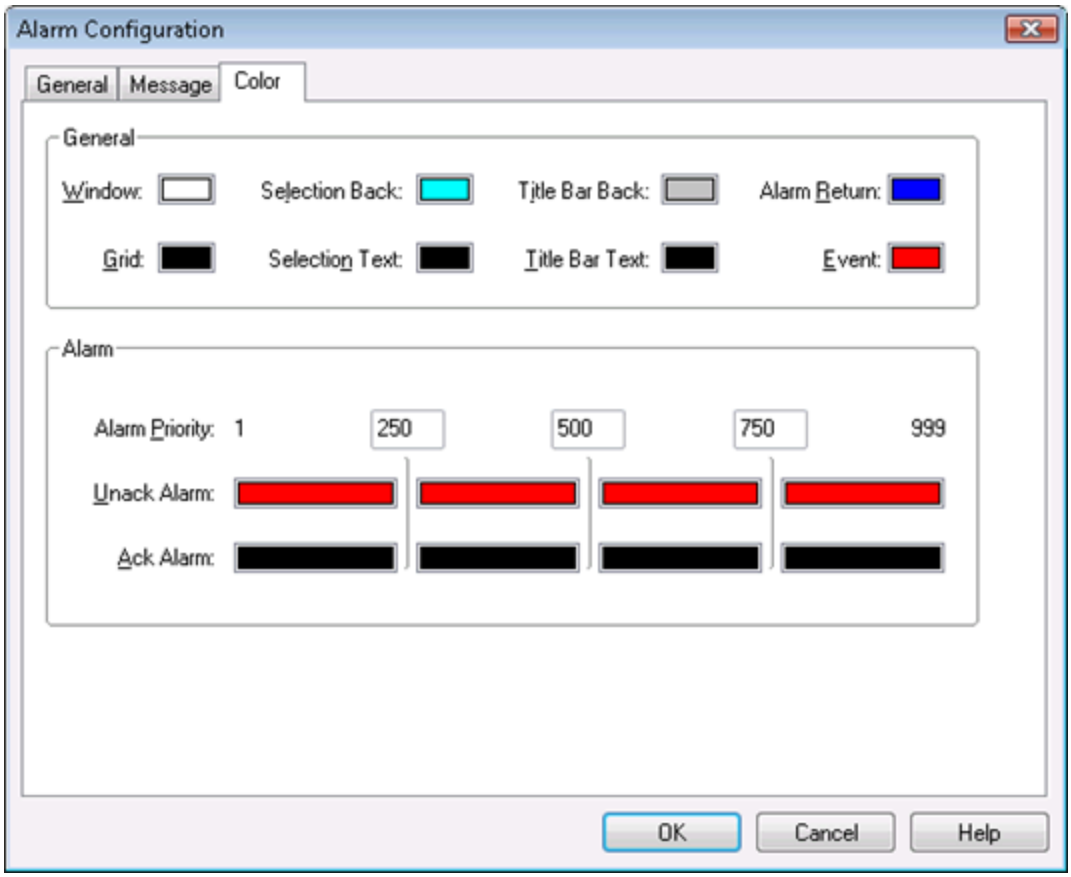
   a. In the **New Name** box, type a new name if you want to show a column name other than the default column name.
   b. In the **New Width** box, type in a default column width. The column width can range from 1 to 999 pixels. The default column width is 100 pixels.
   c. Select **OK** in the **Edit** dialog box.
7. Select **OK** in the **Column Details** dialog box.
8. Select **Apply**.

Configure colors for alarm records

You can configure colors for various parts of the Distributed Alarm Display object, such as the background color and the color of selected records. You can also configure different colors for alarm records of different ranges.

Configure the alarm display colors

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.
2. Select the **Color** tab.



3. In the **General** area, select each color box to open the InTouch color palette. Select the color that you want to use for each of the following:

Option	Description
Window	Sets the display background color.
Grid	Sets the grid color.
Selection Back	Sets the highlighted text background color.
Selection Text	Sets the highlighted text color.
Title Bar Back	Sets the title bar background color (visible only if the **Show Titles** option is on).
Title Bar Text	Sets the title bar text color (visible only if the **Show Titles** option is on).
Alarm Return	Sets the color of returned alarms (alarms that have returned to normal without being acknowledged).
Event	Sets the color of event alarms.

4. In the **Alarm Priority** boxes, type the breakpoint values for the alarm display.

5. Select the **UnAck Alarm** and **Ack Alarm** color boxes to open the InTouch palette. Select the color in the palette that you want to use.
6. Select **OK**.

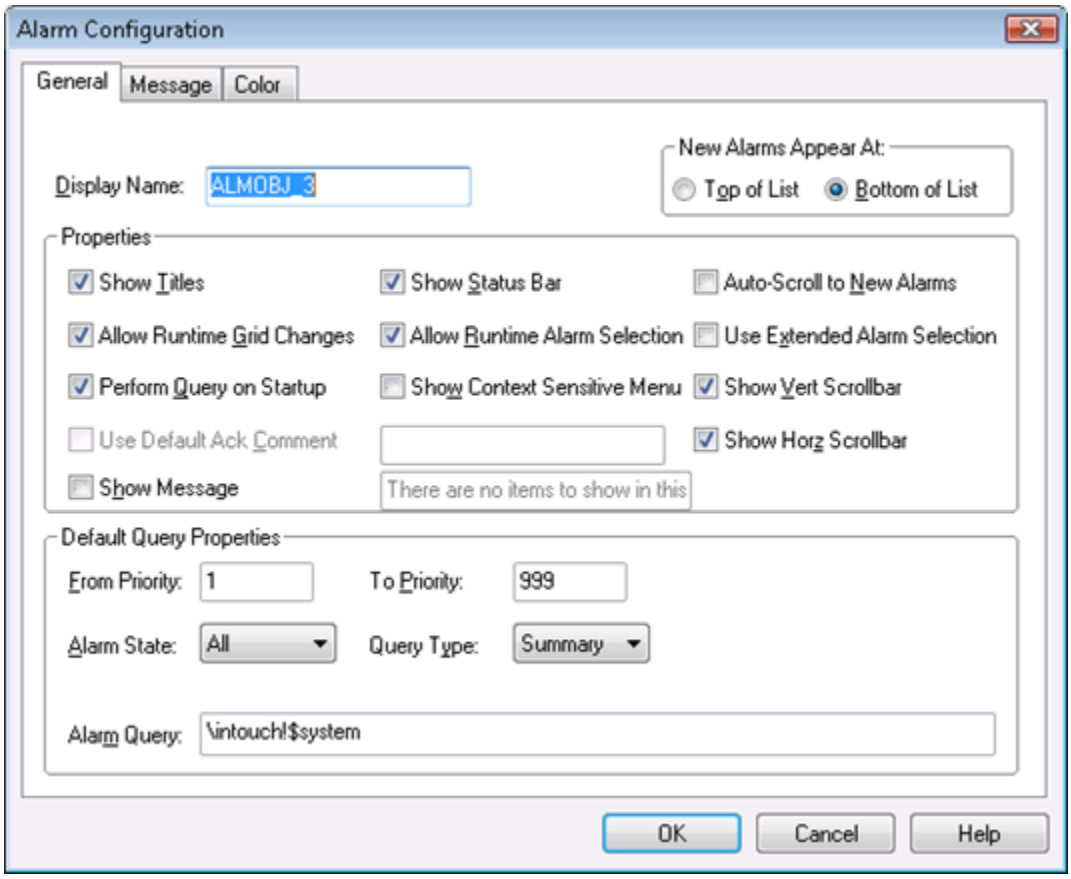
Configure the display type

The Distributed Alarm Display object can show summaries of active alarms or listings of historical alarms.

You can also change the display type dynamically at run time. This can be done, for example, by running the almQuery() script function. The almQuery() script function uses parameters that you can use to set the specified Distributed Alarm Display object (for example: "AlmObj_1") to a specified display type (for example: "Summary"). For more information, see almQuery() Function.

Configure the query default

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.



2. In the **Query Type** list, select the type of alarm display that you want to use for the run-time default.
3. Select **OK**.

Use the Distributed Display to monitor local alarms

You can use the Distributed Alarm Display object to show and acknowledge both local and remote alarms.

To set up a display to monitor just local alarms

1. Right-click the Distributed Alarm Display object and then select **Properties**. The **Alarm Configuration** dialog box appears.
2. In the **Alarm Query** box, type: `\InTouch!$System`

   You can substitute any valid alarm group for $System. You can also define an alarm group list containing just \InTouch!$System, and then use this group list instead of a direct reference.
3. Configure the other parameters for the type of display and any filtering your application requires.

Use a Distributed Alarm Display object at run time

The Distributed Alarm Display object uses a grid to show the alarm messages. This grid allows you to dynamically size column widths simply by selecting a column handle and dragging it to the desired width. This functionality is available only during run time.

Grid column changes are not saved; therefore, if you make grid column changes and close the window containing the alarm display, the grid columns will again be at their default width upon re-opening that window. You can adjust the default column width in WindowMaker.

The grid allows you to select single or multiple alarms in a list box. The selected alarms can be acknowledged by using the almAckSelect() QuickScript function. When you configure the Distributed Alarm Display object, you can also define the selection behavior to allow either toggle selection (item by item) or multiple selection (holding down CTRL or SHIFT keys in conjunction with a mouse click to select multiple alarms). You can turn off run-time selection.

You can also configure up to eight different colors for each shown alarm message based on the priority of the alarm and whether or not it is acknowledged.

The scroll bars are available if they were turned on at configuration time.

The alarm display object may have controls to page through alarm records, depending on how the control is configured.

Sizable display columns

The Distributed Alarm Display object uses a grid to hold the alarm messages. At run-time, you can dynamically size the column widths simply by selecting a column and dragging it. You can also double-click on the vertical grid line to automatically size the column. This functionality is available only during run time. Column resizing must be turned on at configuration time.

Grid column changes are not saved when you close the window containing the alarm display.

Multiple selection

You can select a single or multiple alarms in a list box, depending on how the alarm display is configured.

Alarm message colors

Up to eight different colors can be used for each shown alarm message, based on the priority of the alarm and whether it is acknowledged or not.

Status bar

Depending on how the alarm object is configured, the status bar shows a status message, the current alarm query, and a progress bar.

The left portion of the status bar shows the current status of the control.

These indicators provide an overview of the current state of the display query and provide details about the suppression available in the Distributed Alarm Display object. The right pane of the status bar is red when freeze is in effect and the left pane of the status bar is red when suppression is in effect.

The word "Suppression" shows in the left pane when suppression is in effect.

Feature	Description
Status Message	The status message at the left end of the status bar provides a more detailed description of the current query status.
Alarm Query	The Alarm Query provides a visual indication of the current alarm query.
Progress Bar	The update progress bar at the right end of the status bar provides a visual indication of the current query progress.

The status messages are as follows:

Status Message	State/Indicator	Progress Bar
None	No Query	None
Update Incomplete	Query Incomplete	Blue/Green
Update Successful	Query Complete	Red
Suppression	Query name	Solid Blue
Freeze	Query name	Red

Shortcut menu

Depending on how the alarm object is configured, you can right-click on the object to open a shortcut menu for common commands:

Select this command	To do this
Ack Selected	Acknowledge the selected alarm.
Ack Others	Acknowledge all alarms in the display, or only visible alarms, selected groups, selected tagnames, and priorities.
Suppress Selected	Suppress the selected alarm.
Suppress Others	Suppress all alarms in the display, or only visible alarms, selected groups, selected tagname, and selected priorities.
Query Favorites	Opens the **Alarm Query** dialog box, where you can select the query to use.
Stats	Opens the **Alarm Statistics** dialog box.
Suppression	Opens the **Alarm Suppression** dialog box.
Freeze	Freezes the current display.
Dismiss Selected	Dismiss the selected alarm.
Dismiss Others	Dismiss all alarms in the display, or only visible alarms, selected groups, selected tagname, and selected priorities.

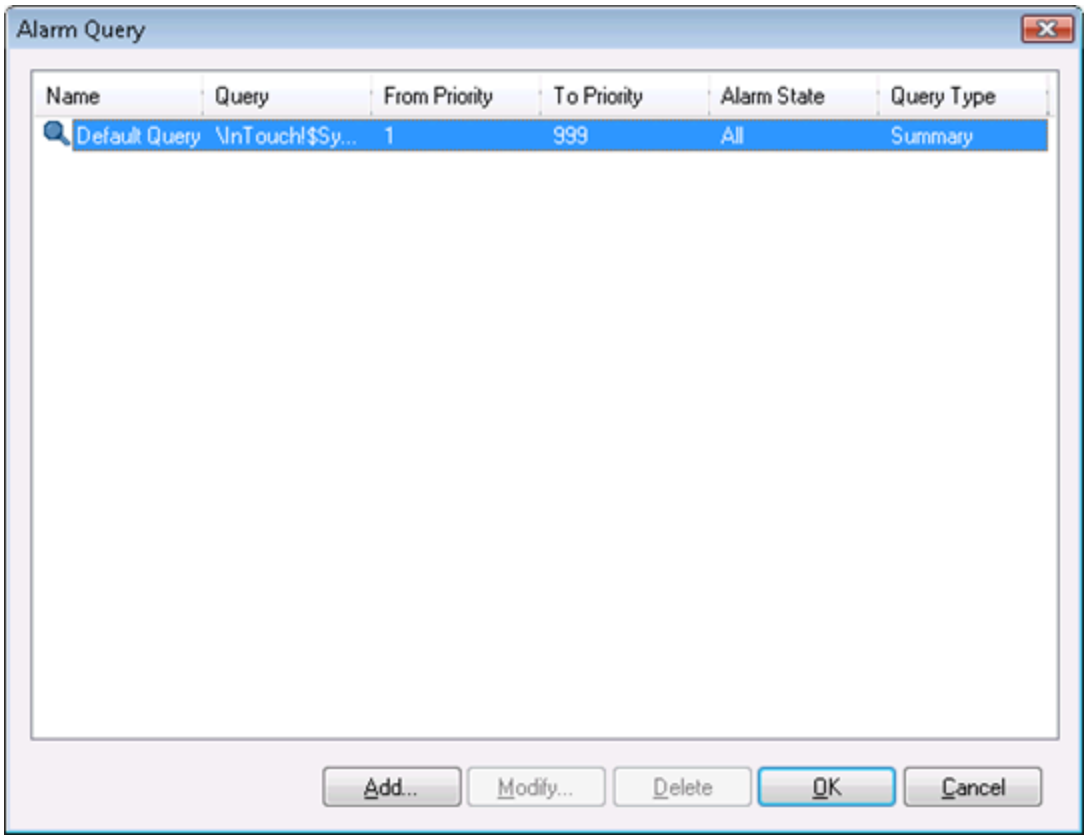
Select and configure alarm query favorites

Use the **Query Favorites** command on the shortcut menu to quickly select an alarm query from a list of previously defined alarm queries. You can also create new named queries, edit an existing query, or delete an existing query.

Note: For multi-line alarm queries appearing in the Distributed Alarm Display, line separations appear as "garbage" characters. This does not affect the function.

Select an alarm query for display

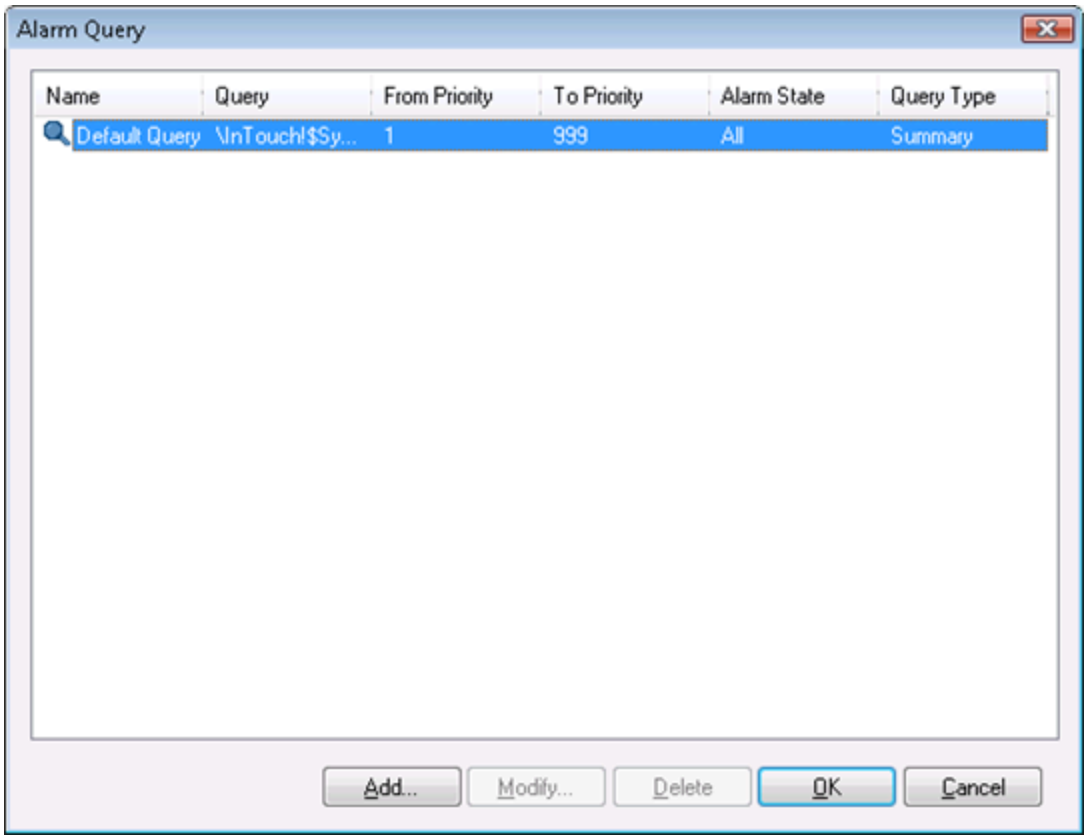
1. At run time, right-click the Distributed Alarm Display and then click **Query Favorites**. The **Alarm Query** dialog box appears.



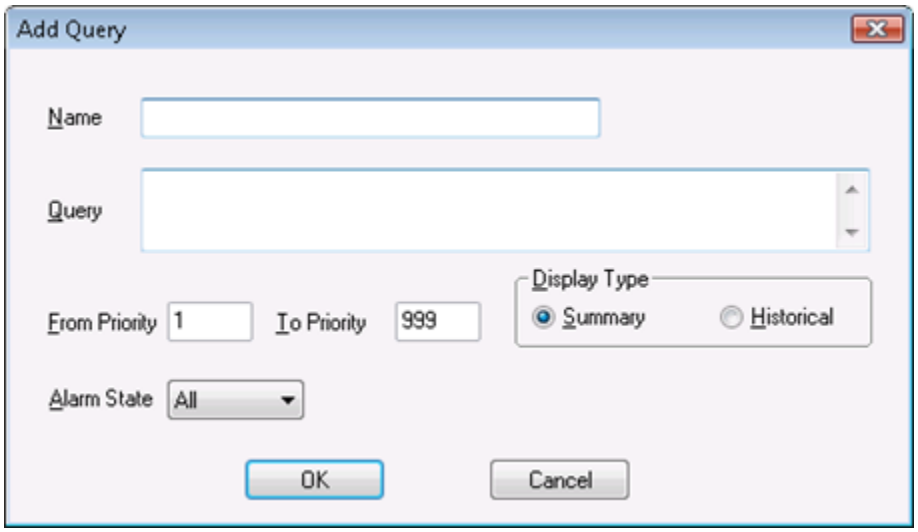
2. In the list of currently defined queries, select the named query to use.
3. Select **OK**. The Distributed Alarm Display object shows the alarm information for the selected query.

To add a new named query

1. At run time, right-click the Distributed Alarm Display and then select **Query Favorites**. The **Alarm Query** dialog box appears.



2. Select **Add**. The **Add Query** dialog box appears.



3. Configure the query. Do the following:
   a. In the **Name** box, type the name for the query.
   b. In the **Query** box, type the sets of InTouch alarm queries that you want to perform. You can specify one or more alarm providers and groups.
   c. In the **From Priority** box, type the minimum alarm priority value (1 to 999). In the **To Priority** box, type the maximum alarm priority value (1 to 999).
   d. In the **Alarm State** list, select the alarm state that you want to use in the alarm query.
   e. In the **Display Type** area, select the type of alarms to show. For more information, see Summary alarms

versus historical alarms.

4. Select **OK** to close the **Add Query** dialog box.
5. Select **OK** in the **Alarm Query** dialog box.

To modify an existing named query

1. At run time, right-click the Distributed Alarm Display and then select **Query Favorites**. The **Alarm Query** dialog box appears.
2. In the list of currently defined queries, select the named query to modify.
3. Select **Modify**. The **Modify Query** dialog box appears.
4. Make the necessary modifications and then select **OK** to close the **Modify Query** dialog box.
5. Select **OK** on the **Alarm Query** dialog box.

Note: Modifications are not automatically applied to other Distributed Alarm Display objects that are using the alarm query being modified.

To delete an existing named query

1. At run time, right-click the Distributed Alarm Display and then select **Query Favorites**. The **Alarm Query** dialog box appears.
2. In the list of currently defined queries, select the named query to modify.
3. Select **Delete**. When a message appears, select **Yes**.
4. Select **OK** on the **Alarm Query** dialog box.

Note: Deletion are not automatically applied to other Distribute Alarm Display objects that are using the alarm query being deleted.

Control the Distributed Alarm Display object using functions and dotfields

You can control the Distributed Alarm Display object at run time using functions and .dotfields.

For a list of returned error numbers for functions, see Error descriptions.

Get or set properties

Properties are accessible through the GetProperty*X*() function, where X is the data type (D for Discrete, I for Integer, and M for Message). For example:

```
GetPropertyM(ControlName.Property, MsgTag)
```

For more information on the GetPropertyX() functions, see Built-In functions.

When you run the script that includes the GetPropertyX() function, the property value is saved to the MsgTag. If multiple rows are selected, the property assigned to MsgTag is the tag value in the first row of the multiple selection.

Acknowledge alarms

The Distributed Alarm Display object is capable of acknowledging any alarms that it can query (summary display

only). The Distributed Alarm Display object includes alarm acknowledgment functions. These functions supplement the .**Ack** dotfield used to acknowledge local alarms and alarm groups. Use these functions to acknowledge all alarms, shown alarms, and selected alarms.

You can also acknowledge alarms by their characteristics. such as group membership, priority, application name, and tag name.

- almAckAll() Function
- Control the Distributed Alarm Display object using functions and dotfields
- almAckGroup() Function
- almAckPriority() Function
- almAckRecent() Function
- almAckTag() Function
- almAckSelect() Function
- almAckSelectedGroup() Function
- almAckSelectedPriority() Function
- almAckSelectedTag() Function

almAckAll() Function

Acknowledges all alarms in a current query, including those not currently shown in the Distributed Alarm Display object in summary mode.

Category

Alarms

Syntax

```
[Result=]almAckAll(ObjectName,Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Comment

Alarm acknowledgment comment.

Examples

```
MessageTag = "Acknowledge All by " + $Operator;
almAckAll("AlmObj_1",MessageTag);
```

See Also

Ack(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckDisplay() Function

Acknowledges only those alarms currently visible in the Distributed Alarm Display object in summary mode.

Category

Alarms

Syntax

```
[Result=]almAckDisplay(ObjectName,Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Comment

Alarm acknowledgment comment.

Example

```
almAckDisplay("AlmObj_1","Display Acknowledgement");
```

See Also

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckGroup() Function

Acknowledges all alarms shown in the named Distributed Alarm object that match the specified provider and group name.

Category

Alarms

Syntax

```
[Result=]almAckGroup( “ObjectName”, ApplicationName, GroupName, Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the InTouch alarm group, such as $System.

Comment

Alarm acknowledgment comment.

Example

```
MessageTag = “Acknowledge group, Turbines, by ” + $Operator;
almAckGroup(“AlmObj_1”, “\Intouch”, “Turbine”, MessageTag);
```

almAckPriority() Function

Acknowledges all alarms shown in the named Distributed Alarm object as a result of the last query that match the alarm’s application name, alarm group, and priority range.

Category

Alarms

Syntax

```
[Result=]almAckPriority(ObjectName, ApplicationName, GroupName, FromPri, ToPri, Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the Group for example, $System

FromPri

Starting number of the alarm priority range. For example, 100.

ToPri

Ending number of the alarm priority range. For example, 900.

Comment

Alarm acknowledgment comment.

Example

```
almAckPriority(“AlmObj_1”, “\\node1\Intouch”, “Turbines”, 10, 100, “Range 10 to 100 acknowledged”);
```

almAckRecent() Function

Acknowledges the most recent alarms that have occurred.

Syntax

```
[Result=]almAckRecent(ObjectName, Comment)
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Comment

Alarm acknowledgment comment.

Example

```
almAckRecent("AlmObj_1",$DateString);
```

almAckTag() Function

Acknowledges all alarms shown in the named Distributed Alarm Display object as a result of the last query. The alarm must match the application name, group name, tag name, and priority range specified by the query.

Category

Alarms

Syntax

```
[Result=]almAckTag("ObjectName", ApplicationName, GroupName, TagName, FromPri, ToPri, Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the application. For example, \\node1\Intouch.

GroupName

The name of the alarm group. For example, $System.

TagName

The name of the tag whose value is in an alarm state.

FromPri

Starting number of the alarm priority range. For example, 100.

ToPri

Ending number of the priority range. For example, 900.

Comment

Alarm acknowledgment comment.

Example

```
almAckTag("AlmObj_1", "\\node1\Intouch", "Turbines", "Valve1", 10, 100, "Acknowledged for Valve1");
```

See Also

Ack(), almAckAll(), almAckGroup(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckSelect() Function

Acknowledges only those alarms selected in the Distributed Alarm Display object in summary mode.

Category

Alarms

Syntax

```
[Result=]almAckSelect(ObjectName,Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Comment

Alarm acknowledgment comment.

Example

This example acknowledges only those alarms that occurred during the day shift or the night shift.

```
IF ($Hour >= 0 and $Hour < 8) THEN
   AckTag = "Night Shift";
ELSE
   AckTag = "Day Shift";
ENDIF;
almAckSelect ("AlmObj_1",AckTag);
```

See Also

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

almAckSelectedGroup() Function

Acknowledges all alarms with same provider and group names that have the same group name as one or more of the alarms that are selected within the named Distributed Alarm Display object.

Category

Alarms

Syntax

```
[Result=]almAckSelectedGroup(ObjectName,Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Comment

Alarm acknowledgment comment.

Example

```
MessageTag = "Acknowledge selected groups by " + $Operator;
almAckSelectedGroup ("AlmObj_1", MessageTag);
```

See Also

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedPriority(), almAckSelectedTag()

almAckSelectedPriority() Function

Acknowledges all alarms with same provider and group names that have the same priority value as one or more of the alarms that are selected within the named Distributed Alarm Display object. The priorities are calculated from the minimum and maximum priorities of the selected alarm records.

Category

Alarms

Syntax

```
[Result=]almAckSelectedPriority(ObjectName, Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Comment

Alarm acknowledgment comment.

Example

```
MessageTag = "Acknowledge selected priorities by " + $Operator;
almAckSelectedPriority ("AlmObj_1", MessageTag);
```

See Also

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedTag()

almAckSelectedTag() Function

Acknowledges all alarms that have the same Tagname from the same provider and group name and having the same priority as one or more of the selected alarms within the named Distributed Alarm Display object. This function works only if InTouch is the alarm provider.

Category

Alarms

Syntax

```
[Result=]almAckSelectedTag(ObjectName,Comment);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Comment

Alarm acknowledgment comment.

Example

```
MessageTag = "Acknowledge selected tagnames by " + $Operator;
almAckSelectedTag ("AlmObj_1", MessageTag);
```

See Also

Ack(), almAckAll(), almAckGroup(), almAckTag(), almAckDisplay(), almAckRecent(), almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority()

Select alarms

You can create scripts to select alarms from a Distributed Alarm Display object. You can select all alarms, only selected alarms, or obtain a count of current alarms.

- almSelectAll() Function
- almUnselectAll() Function
- almSelectionCount() Function
- almSelectGroup() Function
- almSelectItem() Function
- almSelectPriority() Function
- almSelectTag() Function

You can also select specific alarms based upon the data source, alarm priority, and InTouch tags.

almSelectAll() Function

Toggles the selection of all the alarms in a named Distributed Alarm Display object.

Category

Alarms

Syntax

```
[Result=]almSelectAll(ObjectName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

Example

```
If $AccessLevel > 8000 THEN
   almSelectAll("AlmObj_1");
   almAckSelect("AlmObj_1", "Ack Selected by a Manager");
ENDIF;
```

See Also

almSelectItem(), almSelectGroup(), almSelectPriority(), almSelectTag(), almUnSelectAll()

almUnselectAll() Function

Unselects all selected alarms in a named Distributed Alarm Display object.

Category

Alarms

Syntax

```
[Result=]almUnselectAll(ObjectName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

Example

```
If $AccessLevel == 9999 THEN
   almAckSelect("AlmObj_1", "Comment");{This alarm can be acknowledged by only
   Administrator}
   ELSE
   almUnselectAll("AlmObj_1");
ENDIF;
```

See Also

almSelectAll(), almSelectItem(), almSelectGroup(), almSelectPriority(), almSelectTag()

almSelectionCount() Function

Returns the number of alarms selected by the operator in the Distributed Alarm Display object.

Category

Alarms

Syntax

```
[Result=]almSelectionCount(ObjectName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

Example

The AlarmCount tag is assigned the number of alarms selected by the operator from the Distributed Alarm Display object.

```
AlarmCount = almSelectionCount("AlmObj_1");
```

almSelectGroup() Function

Toggles the selection of all alarms that are contained by a named Distributed Alarm Display object as a result of the display's last query and where the resultant alarm contains the same alarm group name.

Category

Alarms

Syntax

```
[Result=]almSelectGroup(ObjectName, ApplicationName,GroupName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the Application. For example, \\node1\Intouch.

GroupName

The name of the Group. For example, $System.

Example

```
almSelectGroup("AlmObj_1","\InTouch","Turbine");
```

See Also

almSelectAll(), almSelectItem(), almSelectPriority(), almSelectTag(), almUnSelectAll()

almSelectItem() Function

Toggles the selection of the last selected or unselected item in an alarm display object.

Syntax

```
[Result=]almSelectItem(ObjectName);
```

Argument

ObjectName
The name of the alarm object. For example, AlmObj_1.

Example

```
almSelectItem("AlmObj_1");
```

See Also

almSelectAll(), almSelectGroup(), almSelectPriority(), almSelectTag(), almUnSelectAll()

almSelectPriority() Function

Toggles the selection of all alarms in a named Distributed Alarm Display object as a result of the display's last query and where the resultant alarms are within the specified priority range.

Category

Alarms

Syntax

```
[Result=]almSelectPriority( "objectName", ApplicationName, GroupName, FromPri, ToPri );
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the Application. For example, \\node1\Intouch.

GroupName

The name of the Group. For example, $System.

FromPri

Starting priority of alarms. For example, 100 or integer tag.

ToPri

Ending priority of alarms. For example, 900 or integer tag.

Example

```
almSelectPriority("AlmObj_1","\\node1\Intouch", "Turbines",10,100);
```

See Also

almSelectAll(), almSelectItem(), almSelectGroup(), almSelectTag(), almUnSelectAll()

almSelectTag() Function

Toggles the selection of all alarms in a named Distributed Alarm Display object as a result of the display's last query and given tag name.

Category

Alarms

Syntax

```
[Result=]almSelectTag (ObjectName, ApplicationName, GroupName, TagName, FromPri, ToPri);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the Application. For example, \\node1\Intouch.

GroupName

The name of the Group. For example, $System.

TagName

The name of the alarm tag.

FromPri

Starting priority of alarms. For example, 100 or integer tag.

ToPri

Ending priority of alarms. For example, 900 or integer tag.

Example

```
almSelectTag("AlmObj_1","\\node1\Intouch","Turbines","Valve1",10,100);
```

See Also

almSelectAll(), almSelectItem(), almSelectGroup(), almSelectPriority(), almUnSelectAll()

Retrieve information about a selected alarm record

You can create scripts that return information about selected alarms. Use the following dotfields in your script:

- .AlarmTime Dotfield
- .AlarmDate Dotfield
- .AlarmName Dotfield
- .AlarmValue Dotfield
- .AlarmClass Dotfield
- .AlarmType Dotfield
- .AlarmState Dotfield
- .AlarmLimit Dotfield
- .AlarmPri Dotfield
- .AlarmGroupSel Dotfield
- .AlarmAccess Dotfield
- .AlarmProv Dotfield
- .AlarmOprName Dotfield
- .AlarmOprNode Dotfield
- .AlarmComment Dotfield

.AlarmTime Dotfield

Returns the time when an alarm occurred. The alarm must be selected in Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM( "ObjectName.AlarmTime",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

In this example, AlmObj_1 is the name of the Distributed Alarm Display object and **almTime** is a memory message tag.

```
GetPropertyM("AlmObj_1.AlarmTime",almTime);
```

If used in a Touch Pushbutton QuickScript, this statement returns the time when the alarm occurred to the **almTime** tag.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmType, .AlarmValue

.AlarmDate Dotfield

Returns the date associated with a selected alarm. The alarm has to be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmDate",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

If used in a Touch Pushbutton QuickScript, this statement returns the date to the **almDate** tag.

```
GetPropertyM("AlmObj_1.AlarmDate",almDate);
```

AlmObj_1 is the name of the Distributed Alarm Display object and **almDate** is a memory message tag that retrieves the date for the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmName Dotfield

Returns the name of the tag associated with a selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmName",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

If used in a Touch Pushbutton QuickScript, this statement returns the name of the alarm to **almName**.

```
GetPropertyM("AlmObj_1.AlarmName",almName);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almName** is a memory message tag that retrieves the name of the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmValue Dotfield

Returns the value of the alarm for the tag associated with the selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmValue,TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Remarks

This function uses a message tag to retrieve the numeric value. This is because the GetProperty functions do not support real numbers. You can use the StringToReal() function to assign the result to a real tag.

Example

If used in a Touch Pushbutton QuickScript, this statement returns the value to **almValue**.

```
GetPropertyM("AlmObj_1.AlarmValue", almValue);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almValue** is a memory message tag containing the alarm value for the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmClass Dotfield

Returns the class of alarm for the tag associated with a selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmClass",Tagname);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

The following statement returns the alarm class associated with the selected alarm.

```
GetPropertyM("AlmObj_1.AlarmClass",almClass);
```

AlmObj_1 is the name of the Distributed Alarm Display object and **almClass** is a memory message tag containing the class of alarm for the tag associated with the selected alarm.

If used in a Touch Pushbutton QuickScript, this statement returns the alarm class of the alarm to the **almClass** tag.

See Also

GetPropertyM(), .AlarmAccess, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmType Dotfield

Returns the alarm type for the tag associated with a selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmType",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

If used in a Touch Pushbutton QuickScript, this statement returns the type of the selected alarm to the **almTyp**e tag when the operator acknowledges the alarm.

```
GetPropertyM("AlmObj_1.AlarmType",almType);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almType** is a memory message tag containing the alarm type for the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmValue

.AlarmState Dotfield

Returns the state of the selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM( "ObjectName.AlarmState",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

If used in a Touch Pushbutton QuickScript, this statement returns the state of the selected alarm to the almState tag.

```
GetPropertyM("AlmObj_1.AlarmState",almState);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almState** is a memory message tag containing the alarm state for the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmTime, .AlarmType, .AlarmValue

.AlarmLimit Dotfield

Returns the limit for the tag associated with a selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM ("ObjectName.AlarmLimit",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Remarks

This function uses a message tag to retrieve the numeric value. This is because the GetProperty functions do not support real numbers. You can use the StringToReal() function to assign the result to a real tag.

Example

If used in a pushbutton QuickScript, this statement returns the limit of the selected alarm to the almLimit tag.

```
GetPropertyM("AlmObj_1.AlarmLimit",almLimit);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almLimit** is a memory message containing the alarm limit for the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmPri Dotfield

Returns the priority (1-999) for the tag associated with a selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmPri", TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

If used in a Touch Pushbutton QuickScript, this statement returns the alarm priority to the almPrilvl tag.

```
GetPropertyM("AlmObj_1.AlarmPri",almPrilvl);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almPrilvl** is a memory message tag containing the priority level of the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmGroupSel Dotfield

Returns the alarm group of the tag associated with a selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM( "ObjectName.AlarmGroupSel",TagName);
```

Parameter

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

If used in a Touch Pushbutton QuickScript, this statement returns the name of the alarm group to the almGroup tag.

```
GetPropertyM("AlmObj_1.AlarmGroupSel",almGroup);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almGroup** is a memory message tag containing the alarm group of the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmGroup, .AlarmName

.AlarmAccess Dotfield

Returns the Access Name of the tag associated with a selected alarm. The alarm record must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
GetPropertyM("Objectname.AlarmAccess",TagName);
```

Parameter

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

If used in a Touch Pushbutton QuickScript, this statement returns the Access Name of the tag associated with the alarm to the almAccess tag.

```
GetPropertyM("AlmObj_1.AlarmAccess",almAccess);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almAccess** is a memory message tag containing the Access Name of the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType

.AlarmProv Dotfield

Returns the alarm provider for the tag associated with a selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM( "ObjectName.AlarmProv",TagName);
```

Parameter

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

If used in a Touch Pushbutton QuickScript, this statement returns the provider name to the almProv tag.

```
GetPropertyM("AlmObj_1.AlarmProv", almProv);
```

AlmObj_1 is the name of the Distributed Alarm Display object and **almProv** is a memory message tag containing the name of the provider for the tag associated with the selected alarm.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmOprName Dotfield

Returns the name of the logged on operator who acknowledged the selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM( "ObjectName.AlarmOprName",TagName);
```

Parameter

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

```
GetPropertyM("AlmObj_1.AlarmOprName",almOprName);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almOprName** is a memory message tag containing the name of the operator responding to the alarm associated with the tag.

If used in a Touch Pushbutton QuickScript, this statement returns the name of the operator to the almOprName tag.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmOprNode Dotfield

Returns the operator node for the tag associated with a selected alarm. The alarm must be selected from the Distributed Alarm Display object in summary mode.

When an alarm is acknowledged in a Terminal Services environment, the Operator Node is the name of the client machine that the respective operator established the Terminal Services session from. If the node name cannot be retrieved, the node's IP address is used instead.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM( "ObjectName.AlarmOprNode",TagName);
```

Parameter

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

```
GetPropertyM("AlmObj_1.AlarmOprNode",almOprNode);
```

Where AlmObj_1 is the name of the Distributed Alarm Display object and **almOprNode** is a memory message tag containing the name of the operator's node for the tag associated with the selected alarm.

If used in a Touch Pushbutton QuickScript, this statement returns the operator’s node to the almOprNode tag.

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmComment, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

.AlarmComment Dotfield

Returns the alarm comment, which is a read/write text string that describes the alarm, not the tag. By default, the comment is empty in a new application.

However, when an old InTouch application is converted to InTouch version 7.11 or later, the tag comment is

copied to the .AlarmComment dotfield for backward compatibility.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM( "ObjectName.AlarmComment",TagName);
```

Parameter

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any message tag.

Data Type

String (read-only)

Example

The following example returns the alarm comment for a tag selected in the AlmObj_1 Distributed Alarm Display object and places it in the almComment tag :

```
GetPropertyM("AlmObj_1.AlarmComment", almComment);
```

See Also

GetPropertyM(), .AlarmAccess, .AlarmClass, .AlarmDate, .AlarmLimit, .AlarmName, .AlarmOprName, .AlarmOprNode, .AlarmPri, .AlarmProv, .AlarmState, .AlarmTime, .AlarmType, .AlarmValue

Set the alarm query

Use the following query functions to retrieve records from the alarm memory.

- almDefQuery() Function
- almQuery() Function
- almSetQueryByName() Function

almDefQuery() Function

Performs a query using default properties to update a named Distributed Alarm Display object.

Category

Alarms

Syntax

```
[Result=]almDefQuery(ObjectName);
```

Argument

ObjectName

The name of the Distributed Alarm Display object. For example, AlmObj_1.

Remarks

The default query properties are specified while developing the Distributed Alarm Display object in WindowMaker.

Example

```
almDefQuery("AlmObj_1");
```

See Also

almQuery(), almSetQueryByName()

almQuery() Function

Performs a query to update a named Distributed Alarm Display object and uses the specified parameters.

Category

Alarms

Syntax

```
[Result=]almQuery(ObjectName,AlarmList,FromPri,ToPri,State,Type);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

AlarmList

Sets the Alarm Query/Name Manager alias to perform the query against, for example, "\intouch!$System" or a

Message tag.

FromPri

Starting priority of alarms to show. For example, 100 or integer tag.

ToPri

Ending priority of alarms to show. For example, 900 or integer tag.

State

Specifies type of alarms to show. For example, "UnAck" or Message tag. Valid states are All, UnAck or Ack.

Type

The type of alarm records that appear in the updated display:
"Hist" = Historical alarms
"Summ" = Summary alarms

Example

This statement retrieves all historical alarms specified in MyAlarmListGroup with a priority of 500 to 600. The alarms appear in the AlmObj_1 alarm display.

```
almQuery("AlmObj_1","MyAlarmListGroup",500,600,"All","Hist");
```

In this example, MyAlarmListGroup is an alarm list configured from the Name Manager setup.

See Also

almDefQuery(), almSetQueryByName()

almSetQueryByName() Function

Starts a new alarm query for the named instance of the Distributed Alarm Display object using the parameters from a user-defined query favorite file.

Category

Alarms

Syntax

```
[Result=]almSetQueryByName(ObjectName, QueryName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

QueryName

The name of the query created by using Query Favorites.

Remarks

This is a query for the particular instance of the Distributed Alarm Display object. There may be several such displays on the screen, each with its own query.

Example

This example starts a new query using parameters from the query named "Turbine Queries."

```
almSetQueryByName("AlmObj_1","Turbine Queries");
```

See Also

almQuery(), almDefQuery()

Check the current query properties

Use the following dotfields to return the status of alarm memory queries.

- .AlarmGroup Dotfield
- .QueryType Dotfield
- .QueryStateDotfield
- .Successful Dotfield
- .PriFrom Dotfield
- .PriTo Dotfield

.AlarmGroup Dotfield

Contains the current query used to populate a Distributed Alarm Display object.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyM( "ObjectName.AlarmGroup",TagName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

TagName

Any message tag.

Remarks

This read-only dotfield contains the current alarm query used by the named Distributed Alarm Display object. This query can be a list of alarm groups or direct alarm provider references.

Data Type

String (read-only)

Example

This statement returns the current alarm query used by the AlmObj_1 Distributed Alarm Display object to the **CurrentQuery** tag:

```
GetPropertyM("AlmObj_1.AlarmGroup",CurrentQuery);
```

See Also

GetPropertyM(), .AlarmGroupSel, .AlarmName

.QueryType Dotfield

Shows the current type of alarm query.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyI( "ObjectName.QueryType",TagName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

TagName

Any integer tag

Remarks

This read-only dotfield contains the current query type used by a named Distributed Alarm Display object.

Data Type

Integer (read-only)

Valid Values

1 = Historical

2 = Summary

Example

The following statement returns the current query type of the AlmObj_1 Distributed Alarm Display object to the AlmQueryType tag:

```
GetPropertyI("AlmObj_1.QueryType",AlmQueryType);
```

See Also

GetPropertyI(), .QueryState

.QueryStateDotfield

Shows the current alarm state query filter.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyI( "ObjectName.QueryState",TagName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

TagName

Any integer tag

Remarks

This read-only dotfield contains the current query filter used by a named Distributed Alarm Display object.

Data Type

Integer (read-only)

Valid Values

0 = All

1 = Unacknowledged

2 = Acknowledged

Example

The following statement returns the current query filter of the AlmObj_1 Distributed Alarm Display object to the AlmQueryState tag:

```
GetPropertyI("AlmObj_1.QueryState", AlmQueryState);
```

See Also

GetPropertyI(), .QueryType

.Successful Dotfield

Indicates whether the current query is successful or not.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyD( "ObjectName.Successful",TagName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

TagName

A discrete tag that holds the property value when the function is processed.

Remarks

This read-only dotfield contains the state of the last query used by a named Distributed Alarm Display object.

Data Type

Discrete (read-only)

Valid Values

0 = Error in query

1 = Successful query

Example

The following statement returns the status of the last query of the AlmObj_1 Distributed Alarm Display object to the AlmFlag tag:

```
GetPropertyD("AlmObj_1.Successful",AlmFlag);
```

See Also

GetPropertyD()

.PriFrom Dotfield

Returns the minimum value of an alarm priority range used by the current query.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyI("ObjectName.PriFrom", Tagname);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

Any integer tag.

Data Type

Integer (read-only)

Example

The following statement returns the minimum priority value of the query used by Distributed Alarm Display object AlmObj_1to the **MinPri** integer tag:

```
GetPropertyI("AlmObj_1.PriFrom",MinPri);
```

See Also

GetPropertyI(), .PriTo, .AlarmPri

.PriTo Dotfield

Contains the maximum value of the alarm priority range used by the current query.

Usage

```
[ErrorNumber=]GetPropertyI("ObjectName.PriTo", Tagname);
```

Parameter

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

An integer tag that holds the property value when the function is processed.

Data Type

Integer (read-only)

Example

The following statement returns the maximum priority value of the query used by AlmObj_1 Distributed Alarm Display object to the MaxPri integer tag:

```
GetPropertyI("AlmObj_1.PriTo",MaxPri);
```

See Also

GetPropertyI(), .PriFrom, .AlarmPri

Check for updates to the Distributed Alarm Display object

Use the following dotfields to find out whether the Distributed Alarm Display object contains all current alarms or if there are pending updates.

- .ListChanged Dotfield
- .PendingUpdates Dotfield

.ListChanged Dotfield

Indicates whether there are any new alarms or updates for the Distributed Alarm Display object.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyD("ObjectName.ListChanged",TagName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

TagName

A discrete tag that holds the property value when the function is processed.

Remarks

This read-only dotfield contains the status about whether there have been any changes that need to be updated in the Distributed Alarm Display object. This property is automatically reset on reading the property.

Data Type

Discrete (read-only)

Valid Values

0 = No new alarms or updates for the display object

1 = New updates for the display object

Example

```
The following statement returns the status of any new alarms or updates for the AlmObj_1
Distributed Alarm Display object to the AlmDispStat tag:
GetPropertyD("AlmObj_1.ListChanged",AlmDispStat);
```

See Also

GetPropertyD()

.PendingUpdates Dotfield

Indicates the number of pending updates to the Distributed Alarm Display object. There are pending updates usually when the display is frozen and new alarm records are created. These do not show, but the pending updates count is increased.

Category

Alarms

Usage

```
[ErrorMessage=]GetPropertyI( "ObjectName.PendingUpdates", TagName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

TagName

An integer tag that holds the property value when the function is processed.

Remarks

This read-only dotfield contains the number of pending updates for a named Distributed Alarm Display object. Any value greater than zero indicates the Distributed Alarm Display object has new alarm data. This value is reset each time the object is refreshed.

Data Type

Integer (read-only)

Example

The following statement returns an integer 1 or greater if there are any pending updates for the AlmObj_1 Distributed Alarm Display object to the AlarmPendingUpdates tag:

```
GetPropertyI( "AlmObj_1.PendingUpdates",AlarmPendingUpdates);
```

See Also

GetPropertyI()

Suppress alarms

The Distributed Alarm Display object can suppress one or more alarms at an alarm consumer that match exclusion criteria. If an alarm matches the exclusion criteria, it does not appear in the instance of the display.

You can use QuickScript functions to suppress alarms.

- almSuppressAll() Function
- almUnsuppressAll() Function
- almSuppressDisplay() Function
- almSuppressGroup() Function
- almSuppressPriority() Function
- almSuppressTag() Function
- almSuppressSelected() Function
- almSuppressSelectedGroup() Function
- almSuppressSelectedPriority() Function
- almSuppressSelectedTag() Function
- almSuppressRetain() Function
- .SuppressRetain Dotfield

almSuppressAll() Function

Suppresses the showing of all current and future instances of the alarms in the current query, including those not currently shown in the Distributed Alarm Display object in summary mode.

Syntax

```
[Result=] almSuppressAll(ObjectName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

Remarks

This function works like almAckAll(), identifying which alarms to suppress by identifying all alarms that you would see if you looked at the display and scrolled up and down to look at them all.

Example

```
almSuppressAll("AlmObj_1");
```

See Also

almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almUnsuppressAll() Function

Clears all suppressed alarms.

Syntax

```
[Result=] almUnSuppressAll(ObjectName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

Example

```
almUnSuppressAll("AlmObj_1");
```

See Also

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag()

almSuppressDisplay() Function

Suppresses the showing of current and future occurrences of alarms visible in the Distributed Alarm Display object in summary mode.

Syntax

```
[Result=]almSuppressDisplay(ObjectName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

Remarks

This function works like the corresponding almAckDisplay() function, identifying which alarms to suppress by identifying all alarms that are currently shown.

Example

```
almSuppressDisplay("AlmObj_1");
```

almSuppressGroup() Function

Suppresses the showing of current and future occurrences of any alarm with the specified provider and group name.

Syntax

```
[Result=]almSuppressGroup(ObjectName, ApplicationName,GroupName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the application. For example, \\node1\InTouch

GroupName

The name of the alarm group. For example, $System

Example

```
almSuppressGroup( "AlmObj_1","\InTouch","Turbines");
```

See Also

almSuppressAll(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressPriority() Function

Suppresses the showing of current and future occurrences of any alarm of the specified priority range having the same provider name and Group name.

Syntax

```
[Result=]almSuppressPriority(ObjectName, ApplicationName, GroupName, FromPri, ToPri);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the application. For example, \\node1\InTouch

GroupName

The name of the Group. For example, $System

FromPri

Starting priority of alarms. For example, 100 or Integer tag.

ToPri

Ending priority of alarms. For example, 900 or Integer tag.

Example

```
almSuppressPriority("AlmObj_1","\\node1\Intouch", "Turbines",10,100);
```

See Also

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressTag() Function

Suppresses the showing of current and future occurrences of any alarm that belongs to the specified tagname having the same provider name, group name, and priority range.

Category

Alarms

Syntax

```
[Result=]almSuppressTag(ObjectName, ApplicationName, GroupName, TagName, FromPri, ToPri,
AlarmClass, AlarmType);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

ApplicationName

The name of the application. For example, \\node1\InTouch

GroupName

The name of the Group. For example, $System

TagName

The name of the alarm tag.

FromPri

Starting priority of alarms. For example, 100 or Integer tag.

ToPri

Ending priority of alarms. For example, 900 or Integer tag.

AlarmClass

The class of the alarm. For example, "Value."

AlarmType

The alarm type of the alarm. For example, "HiHi."

Example

```
almSuppressTag("AlmObj_1","\\node1\Intouch", "Turbines","Valve1",10,100,"Value","LoLo");
```

See Also

almSuppressAll(), almSuppressGroup(), almSuppressDisplay(), almSuppressPriority(), almSuppressRetain(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressSelected() Function

Suppresses the showing of current and future occurrences of the alarms selected in the Distributed Alarm Display object in summary mode.

Category

Alarms

Syntax

```
[Result=]almSuppressSelected(ObjectName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Remarks

This functions works like the **almAckSelect()** function, identifying the alarms by the ones selected in the display object.

Example

```
almSuppressSelected("AlmObj_1");
```

See Also

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

almSuppressSelectedGroup() Function

Suppresses the showing of current and future occurrences of the alarms that belong to the same groups of one or more selected alarms having the same provider name within the named Distributed Alarm Display object.

Category

Alarms

Syntax

```
[Result=]almSuppressSelectedGroup(ObjectName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Remarks

This functions works like the almAckSelectedGroup() function, identifying the alarms that are selected, then identifying the groups to which they belong, and suppressing future occurrences of the alarms from those groups.

Example

```
almSuppressSelectedGroup("AlmObj_1");
```

See Also

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressSelected(),

almSuppressSelectedPriority(), almSuppressSelectedTag()

almSuppressSelectedPriority() Function

Suppresses the showing of current and future occurrences of the alarms that belong to the same priority of one or more selected alarms having the same provider name and Group tag within the named Distributed Alarm Display object.

Category

Alarms

Syntax

```
[Result=]almSuppressSelectedPriority(ObjectName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Remarks

The priorities are calculated from the minimum and maximum of the selected alarm records.

This function works like the almAckSelectedPriority() function, identifying the alarms selected in the display, then identifying the corresponding priorities of those alarms, and suppressing future occurrences of alarms with the same priorities.

Example

```
almSuppressSelectedPriority("AlmObj_1");
```

See Also

almSuppressAll(), almSuppressGroup(), almSuppressTagName(), almSuppressDisplay(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedTag(), almAckSelectedPriority()

almSuppressSelectedTag() Function

Suppresses the showing of current and future occurrences of any alarm that belongs to the same Tagname name of one or more selected alarms having the same provider name, group name, and priority range.

Category

Alarms

Syntax

```
[Result=]almSuppressSelectedTag(ObjectName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Example

```
almSuppressSelectedTag("AlmObj_1");
```

See Also

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressSelectedAlarm(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almAckSelectedTag(), almUnSuppressAll()

almSuppressRetain() Function

Suppresses all alarms raised by subsequent queries.

Category

Alarms

Syntax

```
[Result=]almSuppressRetain(ObjectName,SuppressionRetainFlag);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

SuppressionRetainFlag

Any discrete or analog tag, 0, or non-zero value. TRUE if suppression information is retained for following queries, FALSE otherwise.

Remarks

If the flag is 0 when the alarm query is changed, the suppression filters are removed.

Example

```
almSuppressRetain("AlmObj_1", 0);
```

See Also

almSuppressAll(), almSuppressGroup(), almSuppressTag(), almSuppressDisplay(), almSuppressPriority(), almSuppressSelected(), almSuppressSelectedGroup(), almSuppressSelectedPriority(), almSuppressSelectedTag(), almUnSuppressAll()

.SuppressRetain Dotfield

Reads/writes the status of the feature that retains the suppression for the Distributed Alarm Display object.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyD( "ObjectName.SuppressRetain",TagName);
[ErrorNumber=]SetPropertyD( "ObjectName.SuppressRetain",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

Tagname

A discrete tag that holds the property value when the script is processed.

Data Type

Discrete (read-write)

Valid Values

0 = Retain Off

1 = Retain On

Example(s)

The following statement sets the status of suppression retainer for the "AlmObj_1" from the SupRtn discrete tag:

```
SetPropertyD("AlmObj_1.SuppressRetain", SupRtn);
```

See Also

GetPropertyD(), SetProperty()

Scroll the alarm display

Use the following function and dotfields to scroll the alarm list within the Distributed Alarm Display object vertically or horizontally. You can also freeze the display.

- almMoveWindow() Function
- .Freeze Dotfield
- .PrevPage Dotfield
- .NextPage Dotfield

almMoveWindow() Function

Scrolls the alarm list of the Distributed Alarm Display object vertically or horizontally.

Category

Alarms

Syntax

```
[Result=]almMoveWindow(ObjectName,Option,Repeat);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Option

The type of scrolling action to perform:

Type	Description
LineDn	One line down.
LineUp	One line up.
PageDn	One page down.
PageUp	One page up.
Top	To the top of the list.
Bottom	To the bottom of the list.

PageRt	One page to the right.
PageLf	One page to the left.
Right	To the end of the list (right side).
Left	To the beginning of the list (left side).

Repeat

The number of times this operation should be repeated.

Example

```
almMoveWindow("AlmObj_1", "Bottom", 0);
almMoveWindow("AlmObj_1", "LineDn", 3);
almMoveWindow("AlmObj_1", "PageUp", 0);
```

.Freeze Dotfield

The **.Freeze** dotfield reads the freeze status or freezes/unfreezes the Distributed Alarm Display object.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyD("ObjectName.Freeze", TagName);
[ErrorNumber=]SetPropertyD("ObjectName.Freeze", TagName);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

TagName

A discrete tag that holds the property value when the function is processed.

Remarks

A read-write dotfield that contains or changes the freeze status of a Distributed Alarm Display object. When the alarm display object is frozen, the shown alarms cannot be updated nor can new alarms be added. Freeze has no effect on whether the alarms flash or not.

Data Type

Discrete (read-write)

Valid Values

0 = Freeze OFF

1 = Freeze ON

Example

The following statement sets the Freeze property for the "AlmObj_1" from the AlmFreeze discrete tag.

```
SetPropertyD("AlmObj_1.Freeze",AlmFreeze);
```

See Also

GetPropertyD(), SetPropertyD()

.PrevPage Dotfield

Scrolls the Distributed Alarm Display object one page (one screen full of alarms) up.

Category

Alarms

Usage

```
[ErrorNumber=]SetPropertyD("ObjectName.PrevPage",0);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Remarks

When this property is set, the Distributed Alarm Display object shows the previous page. After the previous page is shown, the variable is automatically set to 1, unless the top of the list has been reached. In this case, the value remains 0.

Data Type

Discrete (read/write)

See Also

GetPropertyD(), SetPropertyD(), .NextPage, .PageNum, .TotalPages

.NextPage Dotfield

Scrolls the Distributed Alarm Display object one page (one screen full of alarms) down.

Category

Alarms

Usage

```
[ErrorNumber=]SetPropertyD("ObjectName.NextPage",0);
```

Arguments

ObjectName

The name of the alarm object. For example, AlmObj_1.

Remarks

When this property is set, the Distributed Alarm Display object shows the next page. After the next page is shown, the variable is automatically set to 1, unless the bottom of the list has been reached. In this case, the value remains 0.

Data Type

Discrete (read/write)

See Also

GetPropertyD(), SetPropertyD(), .PrevPage, .PageNum, .TotalPages

Show alarm statistics and counts

Use the following functions and dotfields to show statistical information about the current Distributed Alarm Display object.

- almShowStats() Function
- .PageNum Dotfield
- .TotalPages Dotfield
- .NumAlarms Dotfield
- .ProvidersReq Dotfield
- .ProvidersRet Dotfield

almShowStats() Function

Shows the **Alarm Statistics** dialog box of the specified Distributed Alarm Display object.

Category

Alarms

Syntax

```
[Result=]almShowStats(ObjectName);
```

Argument

ObjectName

The name of the alarm object. For example, AlmObj_1.

Example

```
almShowStats("AlmObj_1");
```

.PageNum Dotfield

Contains the current page number shown in the alarm object.

Category

Alarms

Syntax

```
[ErrorNumber=]GetPropertyI("ObjectName.PageNum",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

An integer tag that holds the number of the page currently shown from the Distributed Alarm Display object.

Remarks

This read-only dotfield returns the number of the currently shown page in a named Distributed Alarm Display object.

Data Type

Integer (read-only)

Example

The following statement returns the number of the page currently shown from the AlmObj_1 Distributed Alarm Display object to the **AlarmPage** integer tag:

```
GetPropertyI("AlmObj_1.PageNum",AlarmPage);
```

See Also

GetPropertyI(), .NextPage, .PrevPage, .TotalPages

.TotalPages Dotfield

Contains the total number of pages in the Distributed Alarm Display object.

Category

Alarms

Syntax

```
[ErrorNum=]GetPropertyI("ObjectName.TotalPages", TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

An integer tag that retrieves the total number of alarm pages contained in the named Distributed Alarm Display object.

Remarks

This dotfield returns the total number of alarm pages contained in a named Distributed Alarm Display object. One page equals to all alarms shown in the object on the screen at any given time.

Data Type

Integer (read-only)

Example

The following statement returns the total number of pages contained in the AlmObj_1 Distributed Alarm Display object to the Alm**TotalPages** integer tag:

```
GetPropertyI( "AlmObj_1.TotalPages",AlmTotalPages);
```

See Also

GetPropertyI(), NextPage, PrevPage, PageNum

.NumAlarms Dotfield

Contains the number of alarms within a Distributed Alarm Display object.

Category

Alarms

Syntax

```
[ErrorNum=]GetPropertyI("ObjectName.NumAlarms", Tagname);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

An integer tag that holds the current number of alarms registered in a named Distributed Alarm Display object. This includes not only those alarms shown, but all alarms registered.

Remarks

This read-only dotfield returns the total number of alarms within a Distributed Alarm Display object.

Data Type

Integer (read-only)

Example

The following statement returns the current number of alarms used by the AlmObj_1 Distributed Alarm Display object to the **AlarmCount** integer tag:

```
GetPropertyI("AlmObj_1.NumAlarms",AlarmCount);
```

See Also

GetPropertyI()

.ProvidersReq Dotfield

Contains the number of alarm providers required by the current query used by a named Distributed Alarm Display object.

Category

Alarms

Syntax

```
[ErrorNumber=]GetPropertyI( "ObjectName.ProvidersReq",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

An integer tag that holds the current number of alarm providers registered in a named Distributed Alarm Display object. This includes not only those alarms shown, but all alarms registered.

Data Type

Integer (read-only)

Example

The following statement returns the number of alarm providers required by the current query used by Distributed Alarm Display object "AlmObj_1". This value is written to the TotalProv integer tag:

```
GetPropertyI("AlmObj_1.ProvidersReq",TotalProv);
```

See Also

GetPropertyI(), .ProvidersRet

.ProvidersRet Dotfield

Contains the number of alarm providers returned by the current query used by a named Distributed Alarm Display object.

Category

Alarms

Usage

```
[ErrorNumber=]GetPropertyI ("ObjectName.ProvidersRet",TagName);
```

Parameters

ObjectName

Name of the Distributed Alarm Display object. For example, AlmObj_1.

TagName

An integer tag that holds the number of alarm providers that have successfully returned their alarms to the named Distributed Alarm Display object.

Remarks

This read-only dotfield contains the number of alarm providers returned by the current query used by a named Distributed Alarm Display object.

Data Type

Integer (read-only)

Example

The following statement returns the number of alarm providers that have successfully returned their alarms to the AlmObj_1 Distributed Alarm Display object . This value is written to the **RetProv** integer tag:

```
GetPropertyI("AlmObj_1.ProvidersRet",RetProv);
```

See Also

GetPropertyI(), .ProvidersReq

Error descriptions

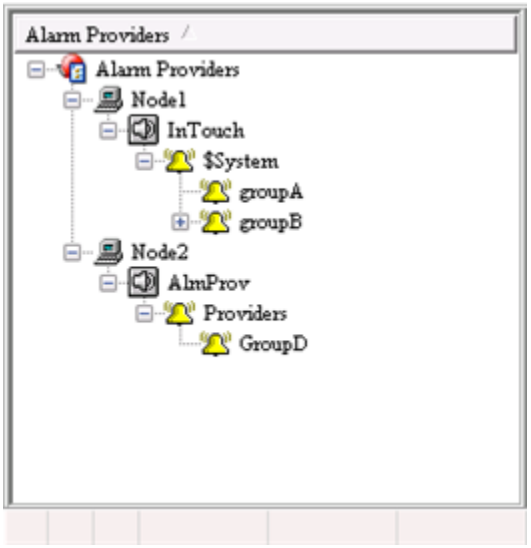
The following table describes the error numbers. If a number is returned that is not in this table, the error is an unknown error.

Error Number	Description
0	Success
-1	General failure

Error Number	Description
-2	Insufficient memory available
-3	Property is read-only
-4	Specified item already present
-5	Object name unknown
-6	Property name unknown

View alarm hierarchies

The Alarm Tree Viewer ActiveX control shows the alarm group hierarchy of alarm providers selected by an alarm query. Items that appear in the Alarm Tree Viewer control include alarm providers, nodes, and groups.



You can enhance the usability of the Alarm Viewer control by using an Alarm Tree Viewer control. You can create a script so that when the operator selects an alarm provider in the Alarm Tree Viewer control, the Alarm Viewer control queries the new alarm provider.

You can configure how the Alarm Tree Viewer control appears and what data is shown. For more information, see Configure an Alarm Tree Viewer control.

When you finish configuring the Alarm Tree Viewer control, you can modify the data you are viewing by:

- Sorting the data by name.
- Updating the tree.
- Performing another query.

For more information about ActiveX controls, see ActiveX controls.

Configure an Alarm Tree Viewer control

You can configure the following for the Alarm Tree Viewer control:

- General control appearance, including colors
- Text font
- Automatic refresh
- Which features users can access at run time
- Which providers and groups to show
- Custom saved queries
- Sort order for alarm groups

You can configure these options from within WindowMaker and while the Alarm Tree Viewer control is running.

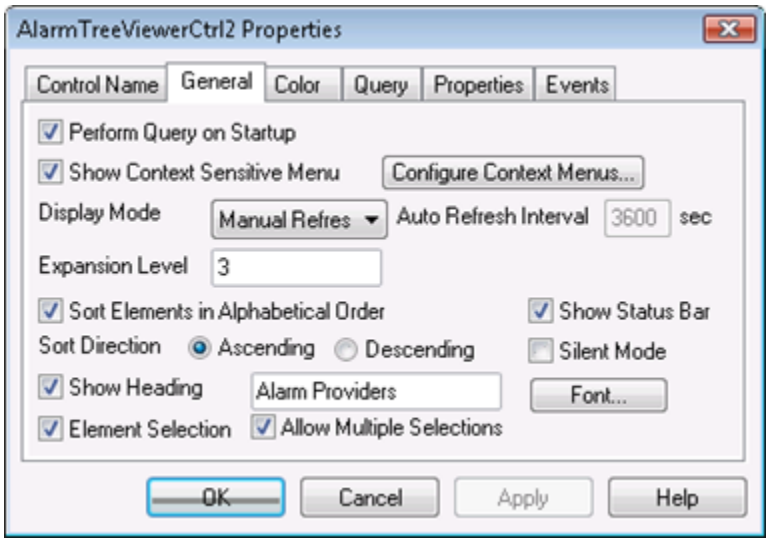
Configure the appearance and colors

When you configure the visual appearance of the Alarm Tree Viewer control, you can:

- Include a status bar.
- Include a column header.
- Set the colors of visual elements.

Configure the appearance

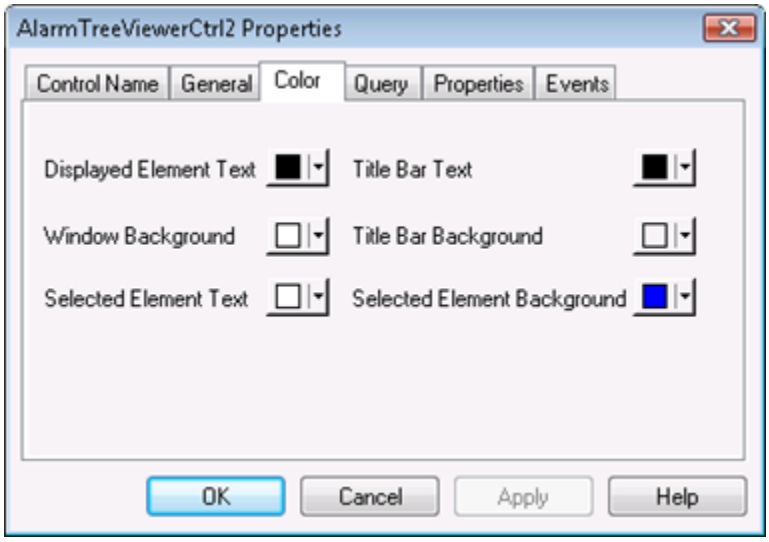
1. Right-click the Alarm Tree Viewer control, and then select **Properties**. The **AlarmTreeViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.



3. Configure how the Alarm Tree Viewer control appears to run-time users. Do any of the following:
   - Select the **Perform Query on Startup** checkbox for the tree to automatically update using default query properties. Otherwise, users must run the Refresh command to update the tree.

- Select the **Show Context Sensitive Menu** checkbox to activate the shortcut menu. Select Configure Context Menus to configure what commands appear on the menu. For more information, see Control which features users can access at run time.
- In the **Display Mode** list, select how you want the tree to refresh. For an automatic refresh, type the refresh interval the **Auto Refresh Interval** box. The range is 5 to 32767 seconds.
- In the **Expansion Level box**, type the number of expansion levels for the tree. This determines to which alarm group branch level the alarm tree is opened when you manually refresh the control. A value of 1 shows only the provider, a value of 2 shows the direct alarm groups of the provider, and so on.
- Select the **Sort Elements in Alphabetical Order** checkbox to sort the tree elements in alphabetical order. Select either **Ascending** or **Descending** for the sort direction.
- Select the **Show Heading** checkbox to show a header above the hierarchy. In the box, type the header bar text.
- Select the **Show Status Bar** checkbox to show a status bar at the bottom of the Alarm Tree Viewer control.
- Select Font to configure the font properties for the tree. The standard Windows **Font** dialog box appears.
- Select the **Element Selection** checkbox to enable users to select an element in the tree.
- Select the **Allow Multiple Selections** checkbox to enable users to select one or more elements using the CTRL and SHIFT keys.
- Select the Silent Mode checkbox prevent the Alarm Tree Viewer control from showing run-time error messages. Error messages are always sent to the Logger.

4. Select **Apply**.
5. Select the **Color** tab.



6. Select the palette button to assign colors to the visual elements of the Alarm Tree Viewer control.

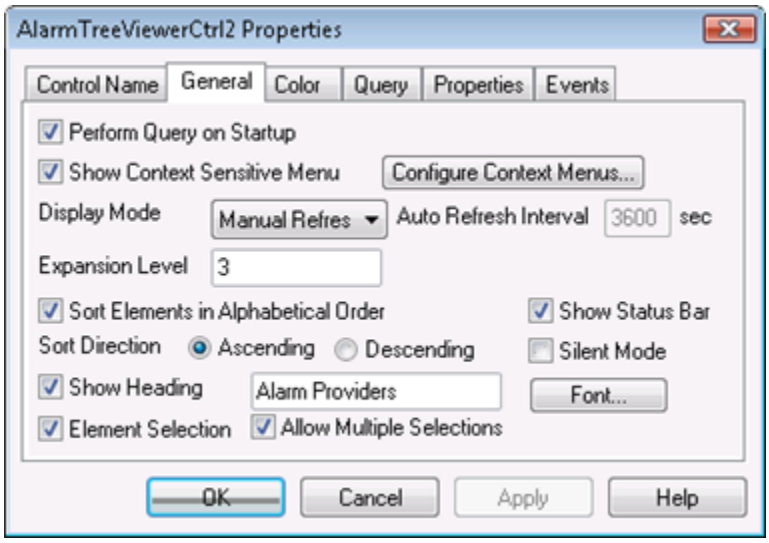
   You can set the colors of the title bar text, window background, selected element text, and selected element background.
7. Select **Apply**.

Configure fonts

You can configure how the text appears for the Alarm Tree Viewer control.

Configure the font

1. Right-click the Alarm Tree Viewer control, and then select **Properties**. The **AlarmTreeViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.



3. Select **Font**. The standard Windows **Font** dialog box appears. Configure the font and then select **OK**.
4. Select **OK**.

Configure automatic refresh

You can configure the Alarm Tree Viewer control to refresh automatically at run time. Otherwise, the operator must refresh the Alarm Tree Viewer control manually.

Configure automatic refresh

1. Right-click the Alarm Tree Viewer control, and then select **Properties**.

   The **AlarmTreeViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.
3. In the **Display Mode** list, select how you want the tree to refresh, either **Manual Refresh** or **Auto Refresh**. For an automatic refresh, type the tree refresh interval the **Auto Refresh Interval** box. The range is 5 to 32767 seconds.
4. Select **Apply**.

Tune the Alarm Tree View refresh

When the Alarm Tree View is being refreshed, it is possible that the alarm providers listed in the tree do not match the alarm providers that were included in the query. This can occur if remote alarm providers have very large hierarchies of alarm groups or if the network connection with the alarm providers is slow.

To ensure that the Alarm Tree View refreshes properly, there are three settings that can be entered and tuned in the [InTouch] section of the InTouch.ini file. These settings specify how long to wait for a complete response to the query and how frequently to retry the alarm query during that period.

These settings are not included in the InTouch.ini file by default and must be manually entered to tune them. When the settings are not explicitly entered in the InTouch.ini file, their default values are used.

AlarmTreeFastRetryMax

This setting determines how long immediately after a query was submitted that fast retries (1 per second) of the tree data retrieval will be performed. Values are in seconds.

Allowed values: 1 to 32767

Default value: 10

Example:

```
[InTouch]
AlarmTreeFastRetryMax=5
```

AlarmTreeSlowRetryInterval

Once the fast retries have been completed, this setting determines the frequency, in seconds, with which additional retries of the tree data retrieval will be performed.

Allowed values: 1 to 32767

Default value: 5

Example:

```
[InTouch]
AlarmTreeSlowRetryInterval=10
```

AlarmTreeTotalRetryMax

This setting specifies the maximum duration, in seconds, for which both types of retries will be performed.

Allowed values: 1 to 32767

Default value: 30

Example:

```
[InTouch]
AlarmTreeTotalRetryMax=60
```

Retry behavior with the default settings

For example, using the default retry settings and the default maximum retry duration of 30 seconds:

- During the fast retry interval, the retrieval will be retried for 10 seconds, once every second.
- During the slow retry interval, the retrieval will be retried every 5 seconds for another 20 seconds.

When the refresh Is considered complete

The display will continue retrying until:

- Connections are completed to all of the providers specified in the alarm query, and
- The hierarchy trees are shown for all the providers requested

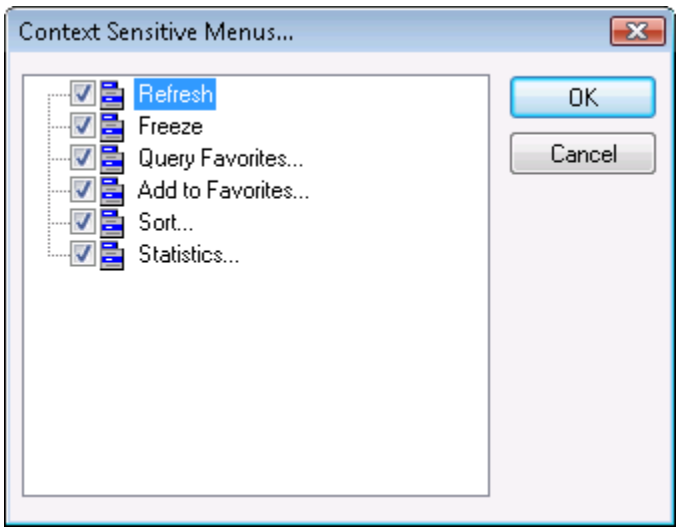
The display refresh timers are reset to their modified or default values each time a query is submitted.

Manage access to features at runtime

The Alarm Tree Viewer control includes a shortcut menu that operators can open by right-clicking on the control during run time. You can configure what commands appear on the menu.

Configure the run-time shortcut menu

1. Right-click the Alarm Tree Viewer control, and then select **Properties**.
   The **AlarmTreeViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.
3. Select the **Show Context Sensitive Menu** checkbox to activate the shortcut menu.
4. Select **Configure Context Menus**.
   The **Context Sensitive Menus** dialog box appears.



5. Select the checkbox for each command that you want to appear in the shortcut menu. You must select least one shortcut command.

Command	Description
Refresh	Refreshes the data shown in the Alarm Tree Viewer control.
Freeze	Allows you to toggle the freeze/unfreeze mode of the tree.

Query Favorites	Shows the **Alarm Query** dialog box to select a query favorite from an available list.
Add to Favorites	Allows you to add new queries from the **Add Query** dialog box.
Sort	Shows the **Sort** dialog box to sort Alarm Tree Viewer control data in ascending or descending order
Statistics	Shows the **Alarm Statistics** dialog box with the percentage of current retrieved alarm providers shown in the Alarm Tree Viewer control.

6. Select **OK** to close the **Context Sensitive Menus** dialog box.
7. Select **Apply**.

Configure which providers and groups to show

You configure alarm queries for alarm providers and groups that belong to the Alarm Tree Viewer control. The alarm query is a list of one or more alarm providers separated by spaces. The valid syntax for the alarm provider is as follows.

Full path to alarm provider:

\\Node\ProviderName

Path to local alarm provider:

\ProviderName

For multiple queries, separate each query with a space. For example:

\InTouch \\Node17\InTouch \\MyNode\InTouch

The default alarm query is \InTouch. You cannot use a tag for the alarm query.

If you query multiple alarm groups and later you undeploy one or more groups, the Alarm Tree Viewer control does not automatically update to remove these groups from the view. You must stop and re-start the alarm provider to un-register it.

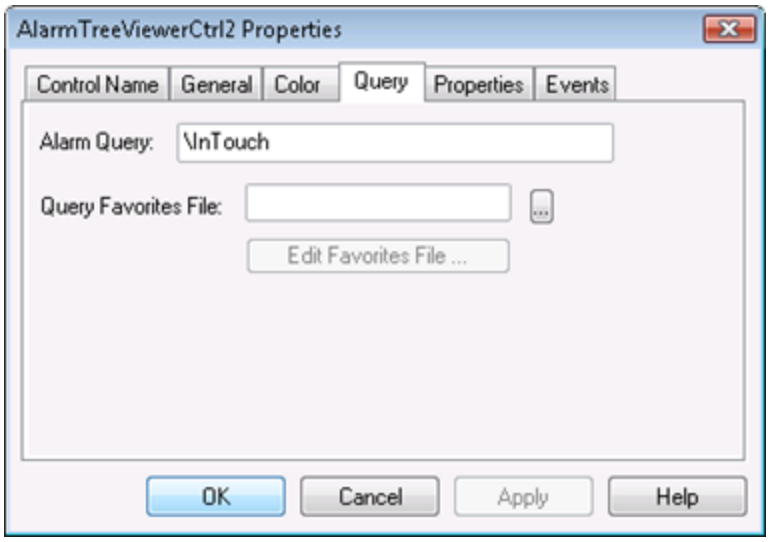
If you query an Galaxy by specifying \Galaxy in your alarm query, then all InTouch alarm providers deployed within the Galaxy are shown. For example:

\\Node\Galaxy!Area[name]

If you query information from a node with multiple alarm providers that contain groups with the same names, records are shown for the last alarm provider in the tree.

Configure the alarm query

1. Right-click the Alarm Tree Viewer control and then select **Properties**. The **AlarmTreeViewerCtrl Properties** dialog box appears.
2. Select the **Query** tab.



3. In the **Alarm Query** box, type the path to the initial alarm query.
4. Select **Apply**.

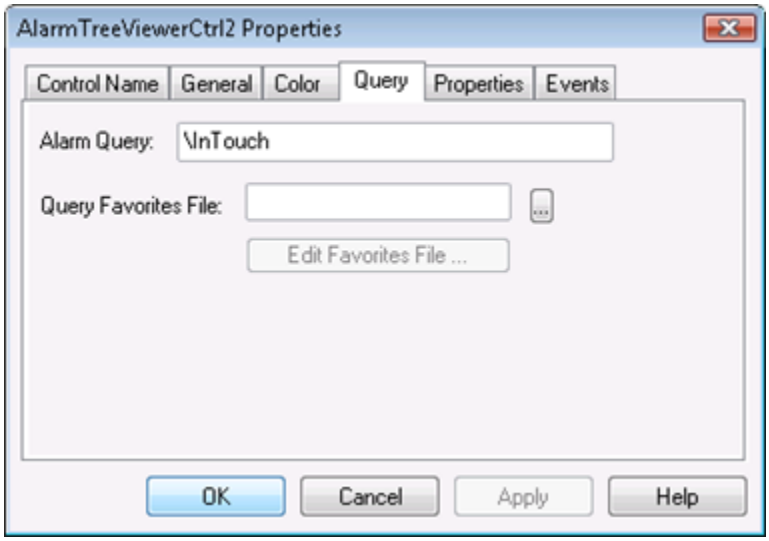
Create custom saved queries using query favorites

You can configure a list of query favorites for operators to select from a shortcut menu.

The query file can be saved to any folder and does not need to be in the InTouch application folder. The alarm query file is an .xml file.

Configure the query favorites file

1. Right-click the Alarm Tree Viewer control and then select **Properties**. The **AlarmTreeViewerCtrl Properties** dialog box appears.
2. Select the **Query** tab.



3. Configure the query favorites file.
   a. In the **Query Favorites File** box, type the network path and file name or select the ellipse button to browse for the file.

b. To edit the **Query Favorites** file, select the **Edit Favorites File** button. The **Alarm Query** window opens, allowing you to add, modify, or delete filters from your favorites file. When you are done, select **OK** to save your changes and close the window.

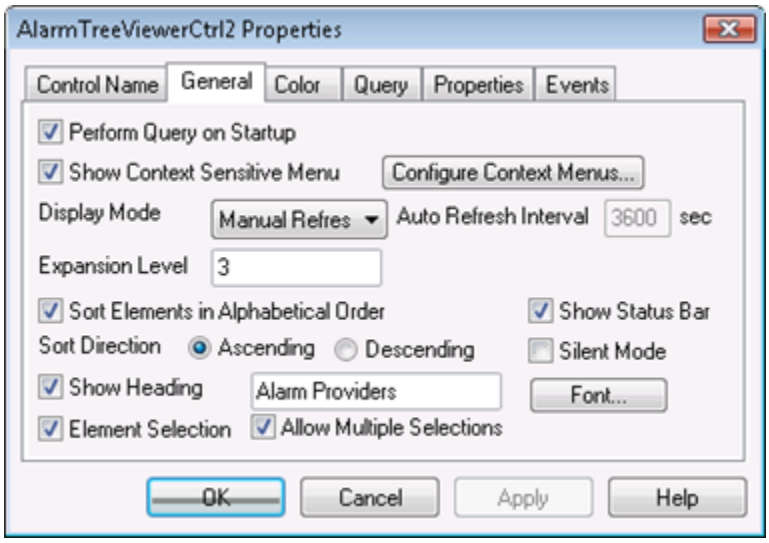
4. Select **OK**.

Configure the sort order for alarm groups

In the Alarm Tree Viewer control, nodes and alarm groups can be shown in alphabetical order, either ascending or descending.

Configure the sort order of alarm groups

1. Right-click the Alarm Tree Viewer control and then select Properties. The **AlarmTreeViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.



3. Select the **Sort Elements in Alphabetical Order** checkbox to list alarm groups in alphabetical order.
4. Select **Ascending** or **Descending** to specify the sort direction.
5. Select **OK**.

Use an Alarm Tree Viewer control at run time

Use the Alarm Tree Viewer control to navigate the hierarchy of alarm providers and alarm groups.

The Alarm Tree Viewer control can show multiple nodes and alarm providers.

- Nodes are represented by a computer icon.
- Alarm providers are represented by a speaker icon.
- Alarm groups are represented by a bell icon.

With one or more alarm groups selected, you can generate queries for alarms that can be used in the Alarm Tree Viewer control and the Alarm DB View controls. To select multiple alarm groups, hold down the shift key while selecting a group. To un-select all groups, select an empty area.

One or more of the following commands appears on the run-time shortcut menu, depending on how the control is configured:

- **Refresh** – Forces a manual update of the alarms.
- **Freeze** – Stops the alarms from updating.
- **Query Favorites** – Opens the **Alarm Query** dialog box where you can select an alarm query from a list of previously defined alarm queries.
- **Add to Favorites** – Opens the **Add Query** dialog box with a query string entered based on the selected Groups (if any).
- **Sort** – Opens the **Sort** dialog box with options to alphabetically sort alarm groups in ascending or descending order.
- **Statistics** – Opens the **Alarm Statistics** dialog box to show the percentage of retrieved alarm providers.

Understand Alarm Tree Viewer control status bar information

The Alarm Tree Viewer control status bar appears at the bottom of the window and shows the following information:

- Name of the current query
- Percentage complete status of the current query

Use query favorites

Use the Query Favorites command on the Alarm Tree Viewer control's shortcut menu to quickly select and run and alarm query from a list of previously defined alarm queries. You can also create new named queries, edit an existing query, or delete an existing query.

Select and run an alarm query

1. Right-click the Alarm Tree Viewer control at run time.
2. Select **Query Favorites**. The Alarm Query dialog box appears.
3. Select the named query that you want to show in the list of currently defined queries.
4. Select **OK**. The Alarm Tree Viewer control shows alarm group information from the selected query.

Use Alarm Tree Viewer control ActiveX properties

You can set the value an Alarm Tree Viewer control property directly using a script or you can assign it to an InTouch tag or I/O reference. For more information about setting properties, see Scripting ActiveX Controls.

The following table lists all Alarm Tree Viewer control properties. For more information on setting color values, see Configure colors for ActiveX Controls.

Property Name	Purpose
AddtoFavoritesMenu	Enables or disables the **Add to Favorites** shortcut menu command.
AlarmQuery	Shows the initial alarm query and allows you to change the query. The valid syntax is `\\<node>\<provider>` or `\<provider>`.
ElementSelection	Controls whether an element in the tree can be selected or not by the operator during run time.
ExpansionLevel	Sets the branch level to which the alarm tree is opened when you manually refresh the control. A value of 1 shows only the provider, a value of 2 shows the direct alarm groups of the provider, and so on.
Font	Gets or sets the font of records and headings shown in the control.
FreezeMenu	Enables or disables the **Freeze** menu command.
HeaderText	Gets or sets the text that appears in the header of the Alarm Tree Viewer control.
MultiSelection	Allows you to select multiple elements in the Alarm Tree Viewer control.
QueryFavoritesFile	Gets or sets the query favorites file name.

Property Name	Purpose
QueryFavoritesMenu	Enables or disables the **Query Favorites** menu command.
QueryStartup	Automatically updates the Alarm Tree Viewer control using default query properties if selected. If not selected, you must requery to update the Alarm Tree Viewer control.
RefreshInterval	Gets the auto refresh interval of the control in seconds.
RefreshMenu	Gets or sets a value that determines whether the **Refresh** command appears in the shortcut menu.
SelTextBackColor	Gets or sets the background color for the selected element.
SelTextColor	Gets or sets the text color for the selected element.
ShowContextMenu	Enables or disables the shortcut menu.
ShowHeading	Shows or hides the title bar of the Alarm Tree Viewer control.
ShowStatusBar	Gets or sets a value that determines whether the status bar is shown.
SilentMode	Gets or sets a value that determines whether the control is in Silent mode.
SortElements	Enables or disables sorting in the Alarm Tree Viewer control.
SortMenu	Enables or disables the **Sort** menu command.
SortOrder	Gets or sets the sort direction. Possible values are "Ascending" and "Descending," represented as 0 and 1 respectively.
StatsMenu	Enables or disables the **Statistics** menu command.
TextColor	Gets or sets the text color the Alarm Tree Viewer control.
TitleBackColor	Gets or sets the title bar background color. Available only if the ShowHeading property is set.
TitleForeColor	Gets or sets the title bar foreground color. Available only if the ShowHeading property is set.

Property Name	Purpose
WindowColor	Gets or sets the window background color of the Alarm Tree Viewer control.

Use Alarm Tree Viewer control ActiveX Methods

You can use the Alarm Tree Viewer control methods in scripts to:

- Retrieve information about the control.
- Retrieve information about specific entries in the alarm hierarchy.
- Freeze the control.
- Create query strings.
- Run queries.

For more information about calling methods, see Scripting ActiveX Controls.

Retrieve information about the Alarm Tree Viewer control

You can use these methods to retrieve information about the Alarm Tree Viewer control.

- AboutBox() Method
- GetElementCount() Method

AboutBox() Method

Shows the Alarm Tree Viewer About dialog box.

GetElementCount() Method

Gets the total number of elements in the tree.

Syntax

```
Object.GetElementCount()
```

Example

The name of the control is AlarmTreeViewerCtrl1 and nTag1 is an integer or real tag.

```
nTag1 = #AlarmTreeViewerCtrl1.GetElementCount();
```

Retrieve information about specific Alarm Tree Viewer entries

You can use a set of methods to retrieve information about elements shown in the Alarm Tree Viewer control

window.

- CheckElementMembership() Method
- GetElementCount() Method
- GetElementName() Method
- GetElementPath() Method
- GetSelectedElementCount() Method
- GetSelectedElementName() Method
- GetSelectedElementPath() Method
- GetSubElementCount() Method
- GetSubElementName() Method
- GetSubElementPath() Method

CheckElementMembership() Method

Checks if the descendant tree element is part of the ancestor tree element.

Syntax

```
Object.CheckElementMembership(PathName, DescendantElementName, AncestorElementName)
```

Parameter

PathName

The name of the path. For example, \InTouch or \\NodeName.

DescendantElementName

The name of the descendant element name. For example, GroupA.

AncestorElementName

The name of the ancestor element name. For example, GroupB.

GetElementCount() Method

Gets the total number of elements in the tree.

Syntax

```
Object.GetElementCount()
```

Example

The name of the control is AlarmTreeViewerCtrl1 and nTag1 is an integer or real tag.

```
nTag1 = #AlarmTreeViewerCtrl1.GetElementCount();
```

GetElementName() Method

Gets the element name corresponding to the index.

Syntax

```
Object.GetElementName(ElementIndex)
```

Parameter

ElementIndex

The index of the element.

Example

The name of the control is AlarmTreeViewerCtrl1 and StrTag is a message tag.

```
StrTag = #AlarmTreeViewerCtrl1.GetElementName(3);
```

GetElementPath() Method

Gets the element path corresponding to the index, down to the indicated expansion level.

Syntax

```
Object.GetElementPath(ElementIndex, ExpansionLevel)
```

Parameter

ElementIndex

The index of the element.

ExpansionLevel

The level of expansion.

Example

The name of the control is AlarmTreeViewerCtrl1, StrTag is a message tag, and returns the path of the element at index 17 up to 4 levels.

```
StrTag = #AlarmTreeViewerCtrl1.GetElementPath(17, 4);
```

GetSelectedElementCount() Method

Gets the number of selected elements in the tree.

Syntax

```
Object.GetSelectedElementCount()
```

Example

The name of the control is AlarmTreeViewerCtrl1 and nTag1 is an integer or real tag.

```
nTag1 = #AlarmTreeViewerCtrl1.GetSelectedElementCount();
```

GetSelectedElementName() Method

Gets the name of the selected element on the Alarm Tree Viewer control.

Syntax

```
Object.GetSelectedElementName()
```

Example

The name of the control is AlarmTreeViewerCtrl1 and StrTag is a message tag.

```
StrTag = #AlarmTreeViewerCtrl1.GetSelectedElementName();
```

GetSelectedElementPath() Method

Gets the path of the selected element to the indicated expansion level.

Syntax

```
Object.GetSelectedElementPath(ExpansionLevel)
```

Parameter

ExpansionLevel
The level of expansion.

Example

The name of the control is AlarmTreeViewerCtrl1 and StrTag is a message tag.

```
StrTag = #AlarmTreeViewerCtrl1.GetSelectedElementPath(3);
```

GetSubElementCount() Method

Gets the total number of sub-elements from the indicated element.

Syntax

```
Object.GetSubElementCount(Path, ElementName)
```

Parameter

Path

The name of the path. For example:
\\NodeName\InTouch
If the path parameter is empty, the Alarm Tree Viewer control finds the first element of the tree that matches the indicated element name.

ElementName

The name of the element. For example, Group1.

Examples

The name of the control is AlarmTreeViewerCtrl1 and nTag1 is an integer or real tag.

```
nTag1 = #AlarmTreeViewerCtrl1.GetSubElementCount("", "Group1" );
nTag1 = #AlarmTreeViewerCtrl1.GetSubElementCount( "\\NodeName", "Group1" );
nTag1 = #AlarmTreeViewerCtrl1.GetSubElementCount( "\InTouch", "Group1" );
nTag1 = #AlarmTreeViewerCtrl1.GetSubElementCount( "\\NodeName\InTouch", "Group1" );
```

GetSubElementName() Method

For the indicated element, gets the name of the sub-element at the corresponding index.

Syntax

```
Object.GetSubElementName(Path, ElementName, ElementIndex)
```

Parameter

Path

The name of the path. For example:
\\NodeName\InTouch
If the path parameter is empty, the Alarm Tree Viewer control finds the first element of the tree that matches the indicated element name.

ElementName

The name of the element. For example, Group1.

ElementIndex

The index of the element.

Examples

The name of the control is AlarmTreeViewerCtrl1 and StrTag is a message tag.

```
StrTag = #AlarmTreeViewerCtrl1.GetSubElementName("", "Group1", 1);
StrTag = #AlarmTreeViewerCtrl1.GetSubElementName("\\NodeName", "Group1", 1);
StrTag = #AlarmTreeViewerCtrl1.GetSubElementName("\InTouch", "Group1", 1);
StrTag = #AlarmTreeViewerCtrl1.GetSubElementName("\\NodeName\InTouch", "Group1", 1);
```

GetSubElementPath() Method

Gets the path of the sub-element from the index of the element name to the indicated expansion level.

Syntax

```
Object.GetSubElementPath(Path, ElementName, ElementIndex, ExpansionLevel)
```

Parameter

Path

The name of the path. For example:
\\NodeName\InTouch
If the path parameter is empty, the Alarm Tree Viewer control finds the first element of the tree that matches the indicated element name.

ElementName

The name of the element. For example, Group1.

ElementIndex

The index of the element.

ExpansionLevel

The level of expansion.

Examples

The name of the control is AlarmTreeViewerCtrl1 and StrTag is a message tag.

```
StrTag = #AlarmTreeViewerCtrl1.GetSubElementPath("", "Group1", 1, 3);
StrTag = #AlarmTreeViewerCtrl1.GetSubElementPath("\\NodeName", "Group1", 1, 3);
StrTag = #AlarmTreeViewerCtrl1.GetSubElementPath("\InTouch", "Group1", 1, 3);
StrTag = #AlarmTreeViewerCtrl1.GetSubElementPath("\\NodeName\InTouch", "Group1", 1, 3);
```

Freeze the tree

You can use the Freeze() method to prevent the Alarm Tree Viewer control tree from being updated with any further changes.

Freeze() Method

Freezes the Alarm Tree Viewer control tree.

Syntax

```
Object.Freeze(Frozen)
```

Parameters

Frozen

Contols whether the tree can be updated.

1 = Freezes the tree.

0 = Unfreezes the tree.

Example

Tag1 is defined as memory discrete tag and the name of the control is AlarmTreeViewerCtrl1.

```
Tag1 = 1;
#AlarmTreeViewerCtrl1.Freeze(Tag1);
```

Create a query string from a selection

You can use the GetAlarmQueryFromSelection() method to retrieve a query string from a selected element in the Alarm Tree Viewer control. The query string can then be used dynamically in an Alarm Viewer control.

GetAlarmQueryFromSelection() Method

Returns an alarm query string from the selected element in the Alarm Tree Viewer control.

Syntax

```
Object.GetAlarmQueryFromSelection()
```

Example

The name of the control is AlarmTreeViewerCtrl1 and StrTag is a message tag. For example: StrTag is set to \\NodeName\InTouch\GroupA.

```
StrTag = #AlarmTreeViewerCtrl1.GetAlarmQueryFromSelection();
```

Run queries

You can run queries for the Alarm Tree Viewer control using methods that either retrieve an existing query saved in a query favorites file or set a string that specifies a new collection of alarm providers.

- SetQueryByName() method
- SetQueryByString() Method

SetQueryByName() method

Sets the current query as specified by the query name passed. The query must be in the query favorites file.

Syntax

```
Object.SetQueryByName(QueryName)
```

Parameter

QueryName

The name of the query as created by using query favorites. For example, Turbine Queries.

Example

The name of the control is AlarmTreeViewerCtrl1.

```
#AlarmTreeViewerCtrl1.SetQueryByName("Turbine Queries");
```

SetQueryByString() Method

Sets the current query as a new string specifying a new collection of Alarm Providers.

Syntax

```
Object.SetQueryByString(NewQuery)
```

Parameters

NewQuery

String containing an alarm query. For example:
\\MasterNode\InTouch

Example

The name of the control is AlarmTreeViewerCtrl1.

```
#AlarmTreeViewerCtrl1.SetQueryByString("\\MasterNode\InTouch");
```

Error handling while using methods and properties

All Alarm Tree Viewer control error messages are sent to the Logger. If you configure the Alarm Tree Viewer control to run in silent mode, no run-time errors are shown.

Use Alarm Tree Viewer control ActiveX events to trigger scripts

You can assign QuickScripts to Alarm Tree Viewer control events, such as a mouse click or double-click. When the event occurs, the QuickScript runs.

The Alarm Tree Viewer control supports the following events:

- Click
- DoubleClick
- ShutDown
- StartUp

The Click event has one parameter called ClicknElementID, which identifies the element in the tree that is clicked at run time.

The DoubleClick event has one parameter called DoubleClicknElementID, which identifies the element in the tree that is double-clicked at run time.

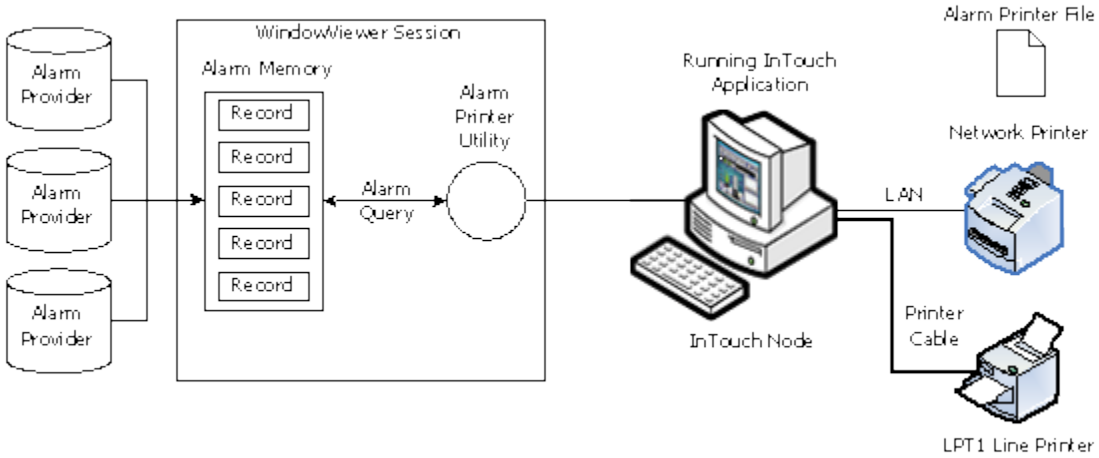
For the Click and DoubleClick events, an ElementID of -1 is returned for the "All Providers" node.

Note: The Alarm Tree Viewer control ignores the user interface methods when they are called from the StartUp event, because the control is not visible yet. These methods include: AboutBox(), CheckElementMembership(), Freeze(), GetAlarmQueryFromSelection(), GetElementCount(), GetElementName(), GetElementPath(), GetSelectedElementCount(), GetSelectedElementName(), GetSelectedElementPath(), GetSubElementCount(), GetSubElementName(), GetSubElementPath(), and Refresh().

For more information about scripting ActiveX events, see Scripting ActiveX Controls.

Print alarms

You use the InTouch Alarm Printer utility to print alarms from multiple nodes. You can print alarm records stored in the alarm memory on an event-by-event basis using a dedicated line or network printer. Also, you can use the Alarm Printer to save alarm records to a file.



You can configure the Distributed Alarm system to print certain events on a line printer as they occur. Typically, you print alarms immediately to record information in the event of a catastrophic failure. Generally, you use a dot matrix printer connected through a serial or parallel port directly to the computer running the InTouch application. Windows network printers and laser printers are usually inappropriate for gathering data for

catastrophic events because they hold entire pages in memory before actually printing a page.

Configure alarm printing and logging

You can run multiple instances of the Alarm Printer. Each instance of the Alarm Printer must be configured to print to a different printer and must be configured with a separate alarm query.

You can configure separate instances of Alarm Printer to print alarms in specific priority ranges. For example, one Alarm Printer instance can print only high priority alarms, while another instance prints only low priority alarms. Likewise, you can use one instance of the Alarm Printer to print alarms from one area of the factory while another instance prints alarms from a different area.

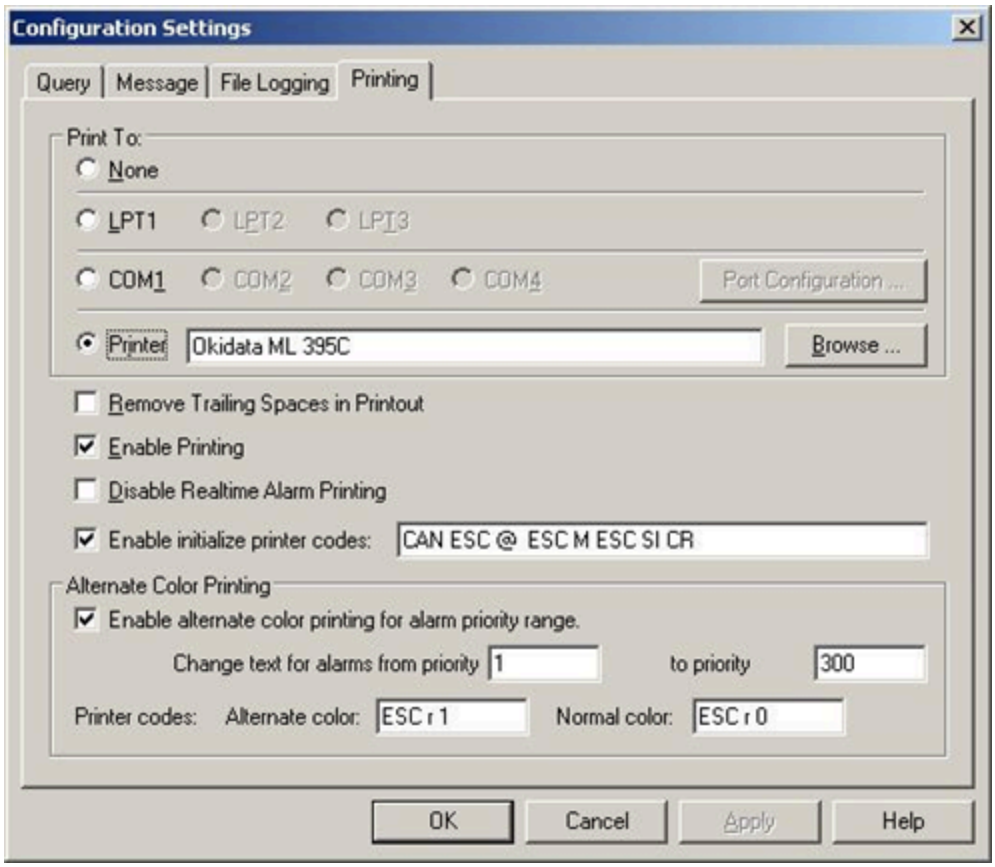
You can save an Alarm Printer configuration to a file, which has the .alc extension. You can create as many configuration files as you want. The Alarm Printer uses an individual configuration file for each instance of Alarm Printer that is running.

Configure printer settings

You must designate a printer to use with the Alarm Printer. Also, you must select if alarms are printed as they occur. Use a dedicated printer for printing alarms. The printer can be connected locally or to a network. Any Windows printer can be used as the output for the Alarm Printer.

Configure the printer settings

1. Open the Alarm Printer utility. Do the following:
   a. In the WindowMaker **Tools** view, expand **Applications**.
   b. Double-click **Alarm Printer**.
2. On the menu bar, select **Configure**. The **Configuration Settings** dialog box appears.
3. Select the **Printing** tab.



4. In the **Print To** area, select the connection to the alarm printer.
   - Select **None** to not use a printer.
   - Select **LPT1-3** to use a printer connected by a parallel port to the computer running the InTouch application.
   - Select **COM1-4** to use a printer connected by a serial port to the computer running the InTouch application. Select **Port Configuration** to show the **COM Properties** dialog box and change the default values assigned to the selected COM port.
   - Select **Printer** to use a printer connected through the network to the computer running the InTouch application. In the box, type the name of the printer or select **Browse** to select an available printer.

     Note: If the printer you want does not appear, add the printer using the Windows Add Printer wizard.
5. Select the **Remove Trailing Spaces in Printout** checkbox to prevent the printer from printing blank lines or empty pages.
6. Select the **Enable Printing** checkbox to print alarms.
7. Select the **Disable Realtime Alarm Printing** checkbox to prevent the Alarm Printer from printing alarms as they occur.
8. Optionally, configure print command sequences to initialize the printer with specific settings and to change the print characteristics for alarms in a specified priority range. For more information, see Configure alarm print commands.
9. Select **OK**.

Configure alarm print commands

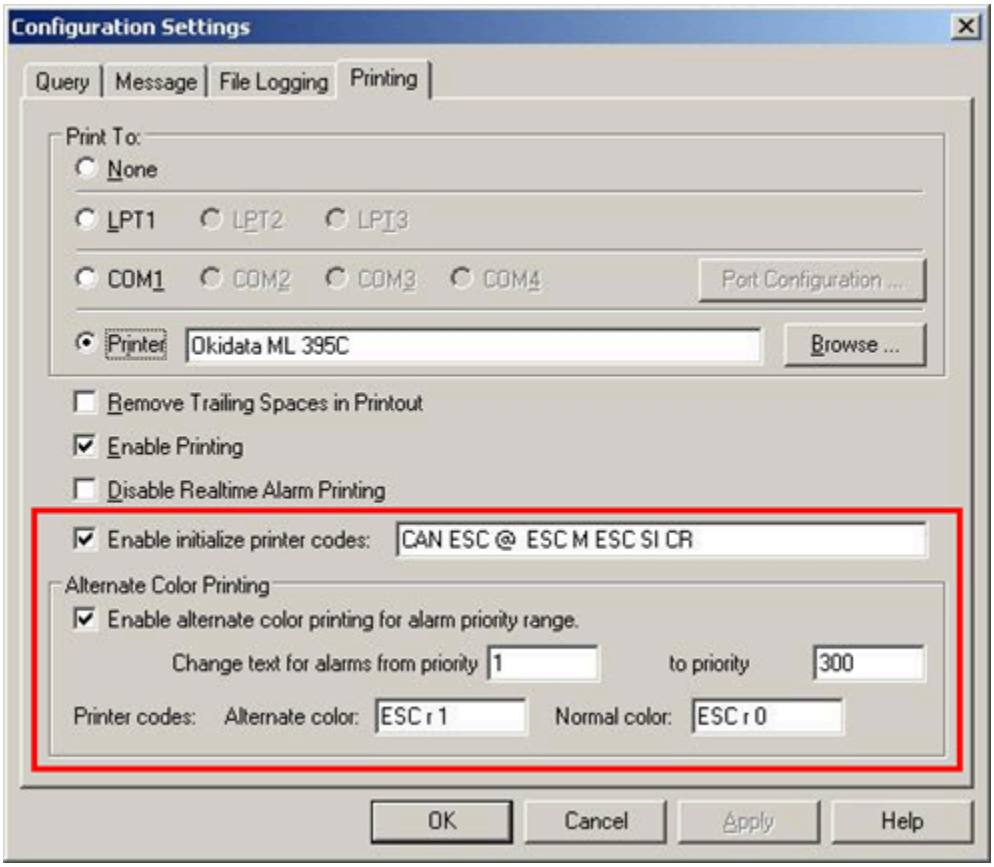
You can configure the Alarm Printer to send a printer command initialization sequence when the printer is first opened (that is, each time the query starts and stops). Printer command initialization sequences can be used to configure printer settings, such as the character pitch and the left and right print margins.

You can also configure alternate and normal print colors so that the print color indicates whether the alarm is in or outside of a specified range of alarm priorities. The alternate and normal color commands do not necessarily have to specify a color. They can also be used to enable/disable double-striking, emphasized printing, underlining, italic, or other differentiating print characteristics to make one set of alarm priorities appear visually distinct from the others.

Note: For monochrome printers, you might find that double-strike printing is much faster than emphasized printing.

Print command settings

The **Printing** tab in the **Configuration Settings** dialog box includes settings for specifying printer command sequences.



These settings are described in the following table.

Setting	Description
Enable initialize printer codes	Causes the command sequence that is entered in the accompanying box to be sent to the printer each time the query is started and stopped.
Enable alternate color printing for alarm priority range	Select this option to pass the alternate and normal color commands for use with the specified alarm priority range.
Change text for alarms from priority … to priority	Enter the alarm priority range from a low number (representing higher priority alarms) to a higher number (representing lower priority alarms).
Alternate color	The printer command sequence that will be sent each time a row of alarm text is sent to the printer when the alarm priority is in the specified range.
Normal color	The printer command sequence that will be sent each time a row of alarm text is sent to the printer when the alarm priority is not in the specified range.

Sample printer command sequences

As an example, the following printer command sequences are for an Okidata Microline 395C color dot matrix printer. Refer to the documentation that accompanied your printer for its command sequences.

Printer initialization command:

```
CAN ESC @ ESC M ESC SI CR
```

where the commands are as follows:

- CAN – Cancel any remaining data in the printer's internal buffer.
- ESC @ – Reset the printer to menu defaults.
- ESC M – Set the printing to 12 CPI (Elite).
- ESC SI – Set printing to compressed to obtain 20 CPI.
- CR – Move the print head back to the left margin.

Alternate color command that sets the print color to red:

```
ESC r 1
```

Normal color command that sets the print color to black:

```
ESC r 0
```

Printer command escape sequences syntax

Each printer command escape sequence consists of one or more bytes. Information about these bytes should be

available in the printer's documentation or by knowing that the printer can parse ESC/P or ESC/P2 syntax. Information about these syntaxes is publicly available on the Internet.

Generally, there is no limitation on the length of the printer command escape sequence text. The parser interprets the stream of text tokens and converts them to a sequence of contiguous byte values. The byte values are then streamed to the printer, unmodified, at the appropriate time, such as at printer initialization or when the printer setting for the alarm must be modified and restored.

When specifying the printer command escape sequence as text, each byte character must be separated by a space character. The space character indicates to the parser where a token begins and ends. For example, if the printer's draft mode can be activated by ESCx0, then the command sequence is entered as:

```
ESC x 0
```

If you need to send a space character to the printer, then use the abbreviation SP to represent that byte.

Note: The parser is case-sensitive, so all abbreviations must be entered in capital letters. For example, use ESC, not esc or Esc.

The following types of items can be included in the command sequence:

- The abbreviated name of the control characters
- The number, letter, or other printable character
- The decimal number for the ASCII character
- The hexadecimal number for the ASCII character

For a list of the hexadecimal or decimal numbers for the control and printable characters, refer to a publicly available ASCII table.

Enter names of control characters

You can enter the names of control characters, as these are not easily typed from the keyboard. For a list of control character names, refer to the Abbr column of a publicly available ASCII table. The only character that is not typically listed in an ASCII table is the space character, which must be entered as SP.

Note: The parser does not process common C language escape characters (for example, \r, \n, and \t).

Enter printable characters

Enter any printable character by typing it in the box. The only exception is the space character (hexadecimal 0x20, decimal 32), which must be entered as SP. The character's value is interpreted as the number that represents it in the ASCII table, and that is the value sent to the printer.

For a list of printable characters, refer to a publicly available ASCII table.

Enter a character's decimal number

Enter the character's decimal number by typing it in the box. The maximum decimal value that can be entered is 255 (hexadecimal 0xFF). There is no warning if you enter a decimal number that is too large, such as 497; it will cause 255 to be returned. Negative numbers are not supported.

Enter a character's hexadecimal number

Enter the character's hexadecimal number by typing it in the box. The allowable range of values is 0x00 to 0xFF.

Configure which alarms to print

The Alarm Printer queries the alarm memory to select records to print or save to a log file. The query selects records from the internal alarm memory based upon:

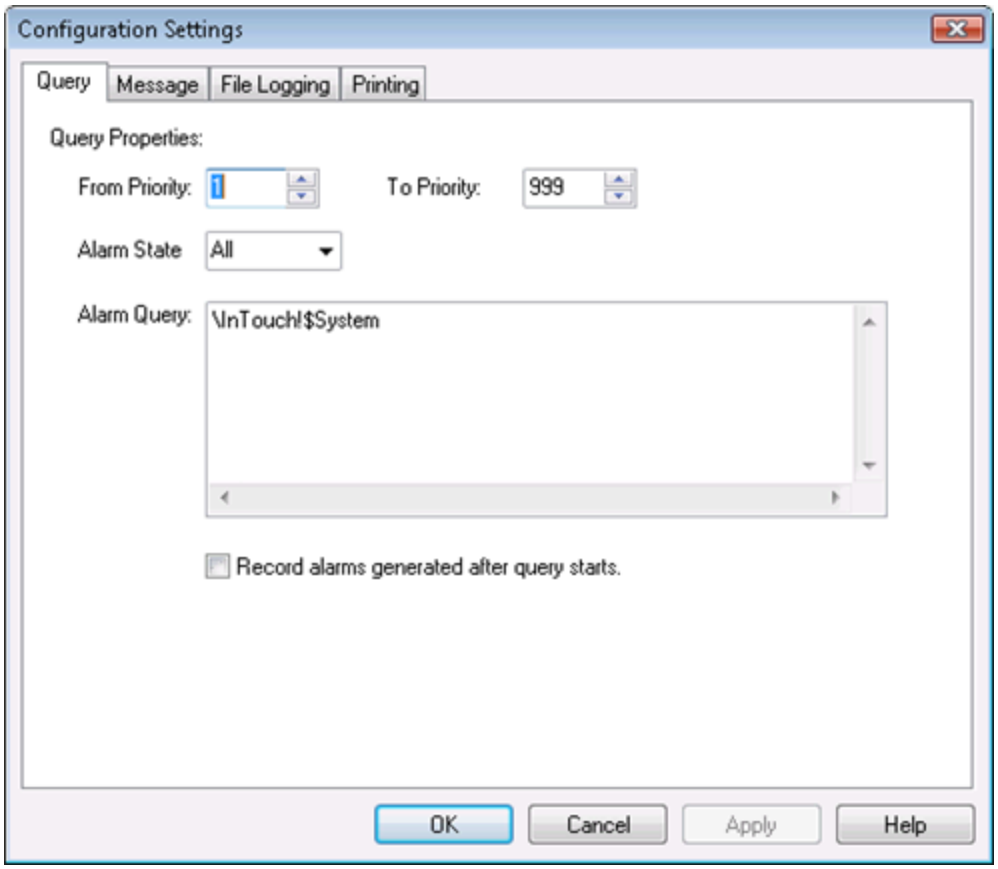
- Alarm priority
- Current alarm state (unacknowledged/acknowledged)
- Alarm group membership

Each alarm has an assigned priority number that represents the severity of the alarm. An alarm priority ranges from 1 to 999. The most severe alarm is assigned a priority of 1. The least severe alarm is assigned a priority of 999.

If a network or printer connection fails, the Alarm Printer does not reprint all alarms. The Alarm Printer only prints the alarms that have not been printed before the connection failure.

Configure which alarms to print

1. Open the Alarm Printer utility. Do the following:
   a. In the WindowMaker **Tools** view, expand **Applications**.
   b. Double-click **Alarm Printer**.
2. On the menu bar, select **Configure**. The **Configuration Settings** dialog box appears.
3. Select the **Query** tab.



4. In the **From Priority** box, enter the highest priority alarm value (1 to 999).
5. In the **To Priority** box, enter the lowest priority alarm value (1 to 999).
6. In the **Alarm State** list, select the alarm state.

Select	To show
All	All alarms.
Ack	Only acknowledged alarms.
Unack	Only unacknowledged alarms.

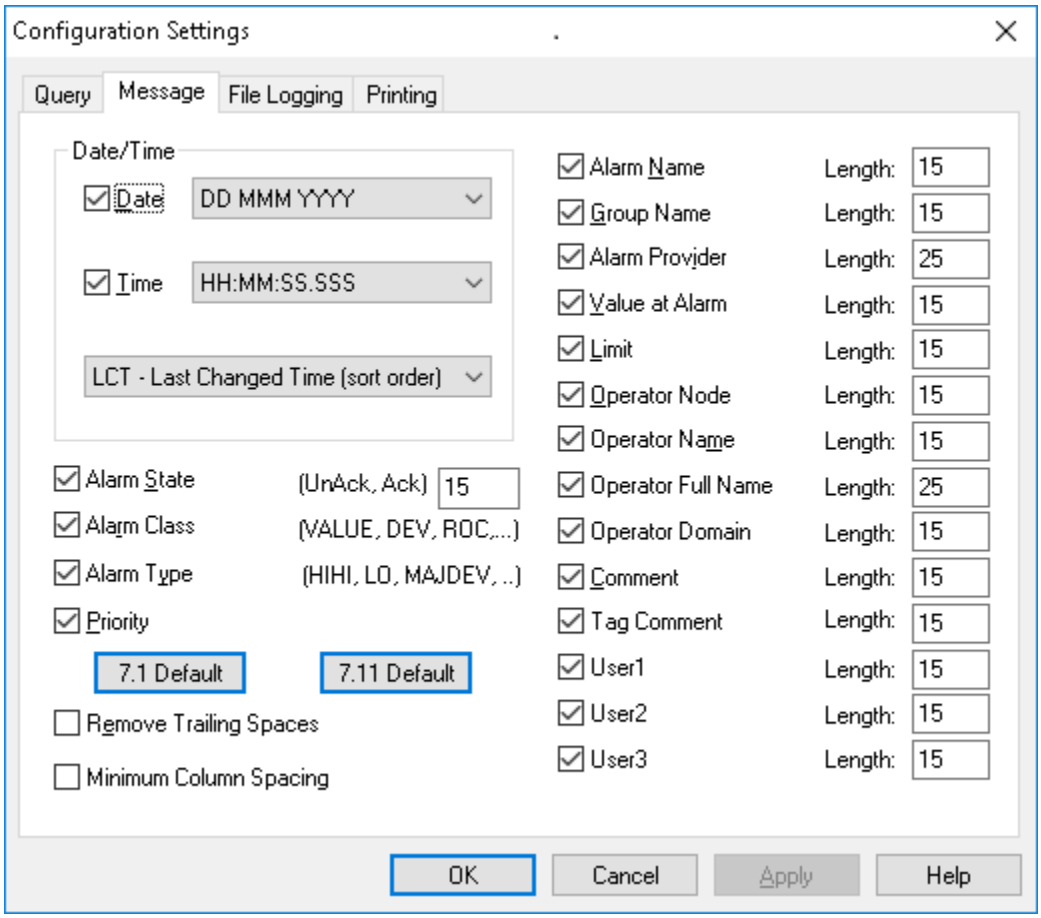
7. In the **Alarm Query** box, type one or more alarm queries. You can specify one or more alarm providers and groups. Use blank spaces to separate the queries.
8. Select the **Record alarms generated after query starts** checkbox to only include alarms that occur after the query starts. The Alarm Printer ignores alarms records that are in the alarm memory and were triggered before the Alarm Printer started querying.
9. Select **OK**.

Configure the format of print and file output

Each option you select appears as a separate field in the printed output. Fields containing data exceeding the specified maximum field length are truncated to those limits.

Configure the format of alarm print and file output

1. Open the Alarm Printer utility. Do the following:
   a. In the WindowMaker **Tools** view, expand **Applications**.
   b. Double-click **Alarm Printer**.
2. On the menu bar, select **Configure**. The **Configuration Settings** dialog box appears.
3. Select the **Message** tab.



4. In the **Date/Time** area, select the **Date** checkbox, and then select a date format from the list. The listed date formats include the following components:

Option	Description
DD	Two-digit day of the month (01-31)
MM	Two-digit month of the year (01-12)
YY	Last two digits of the year
YYYY	Four-digit year
MMM	Three-character abbreviation of the month

5. Select the **Time** checkbox and select a time format from the list. The listed formats included the following

time components:

Option	Description
AP	Selects the AM/PM time format. For example, 3:00 PM is shown as 3:00 PM. A time without this designation defaults to 24 hour military time format. For example, 3:00 PM is shown as 15:00.
HH	Two-digit hour of the day (00-23 or 01-12) when the alarm/event occurred.
MM	Two-digit minute of the hour (00-59) when the alarm/event occurred.
SS	Two-digit second of the minute (00-59) when the alarm/event occurred.
SSS	Three-digit millisecond of the second when the alarm/event occurred.

6. Select the order that alarms appear in the alarm record according to the onset time of the alarm:

Option	Description
OAT	(Original Alarm Time) The date/time stamp of the onset of the alarm.
LCT	(Last Changed Time) The date/time stamp of the most recent change of status for the instance of the alarm: onset of the alarm, change of sub-state, return to normal, or acknowledgment.
LCT But OAT on ACK	(Last Changed Time, but Original Alarm Time on acknowledgement) The last changed time is used while the alarm is unacknowledged, then original alarm time is used after the alarm has been acknowledged.

7. Select the alarm information to be printed.

Option	Description
Alarm State	Prints the alarm state. For example, UnAck, Ack.
Alarm Class	Prints the alarm class. For example, VALUE, DEV, ROC.
Alarm Type	Prints the alarm type. For example, HIHI, LO, MAJDEV.

Option	Description
Priority	Prints the alarm priority (1-999).
7.1 Default (button)	Selects the default settings for the Alarm Printer utility for InTouch version 7.1.
7.11 Default (button)	Selects the default settings for the Alarm Printer utility for InTouch version 7.11.
Remove Trailing Spaces	Removes the extra trailing spaces from a printed field when the length of the actual field value is less than the value configured for that field.
Minimum Column Spacing	Reduces the spacing between columns so that more fields can fit on the printed page.
Alarm Name	Prints the alarm name (tag). In the Length box, type the number of characters (64 characters maximum) for the alarm name.
Group Name	Prints the alarm group name. In the Length box, type the number of characters (64 characters maximum) for the alarm group name.
Alarm Provider	Prints the name of the alarm provider. In the Length box, type the number of characters (64 characters maximum) for the alarm provider name.
Value at Alarm	Prints the value of the tag. In the Length box, type the number of characters (32 characters maximum) for the alarm value.
Limit	Prints the tag's alarm limit. In the Length box, type the number of characters (32 characters maximum) allowed for alarm limit. The number should be large enough to provide the desired level of precision.
Operator Node	Prints the operator node associated with the alarm condition. In the Length box, type the number of characters (64 characters maximum) allowed for the operator's node. In a Terminal Services environment, this is the name of the client computer that the respective operator established the Terminal Services session from. If the node name can't be retrieved, the node's IP address is used instead.

Option	Description
Operator Name	Prints the operator name associated with the alarm condition. In the Length box, type the number of characters (16 characters maximum) allowed for the operator’s name.
Operator Full Name	Prints the logged-on operator's full name.
Operator Domain	Prints the logged-on operator's domain associated with the alarm condition.
Comment	Prints the alarm comment associated with the tag. In the Length box, type the number of characters (131 characters maximum) allowed for the comment.
Tag Comment	Prints the comment for the tag.
User1	Prints the numerical values of User Defined Number 1 corresponding to the alarm.
User2	Prints the numerical values of User Defined Number 2 corresponding to the alarm.
User3	Prints the string value of the user-defined string property associated with the alarm. The maximum number of characters is 131.

8. Select **Apply**.

Configure log files for alarms

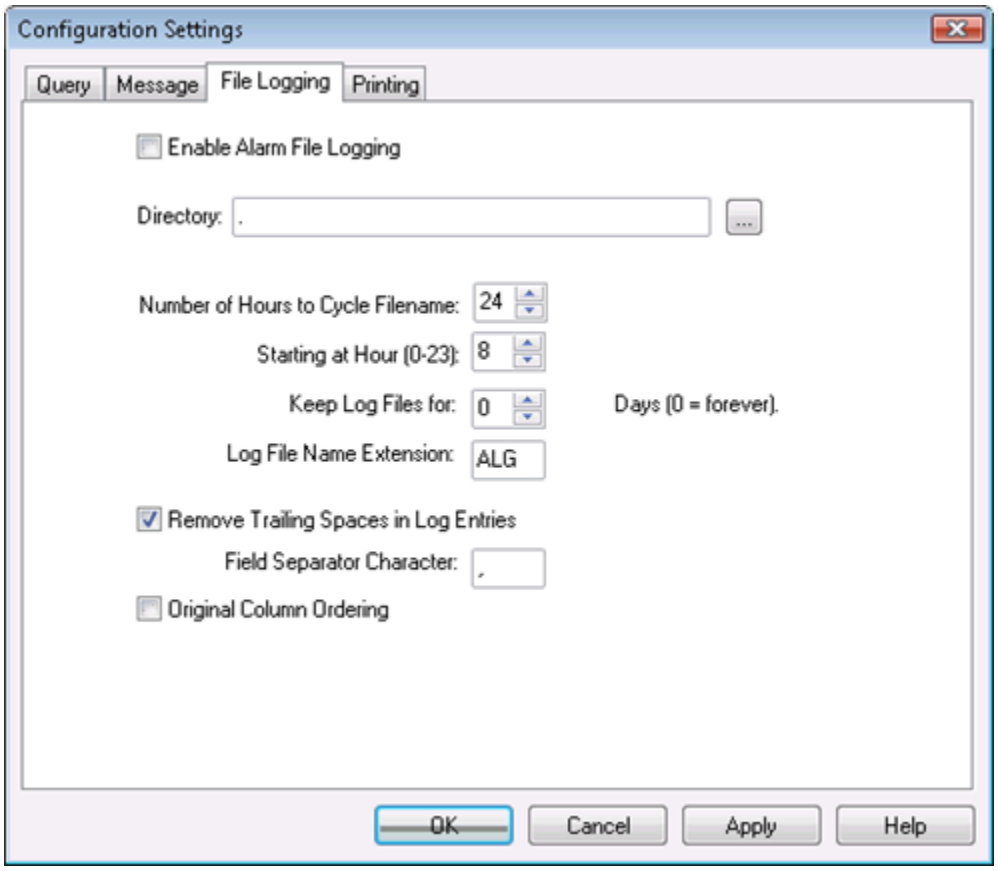
You can log alarm records to a file. By default, the Alarm Printer assigns names to log files based upon the following naming convention:

YYMMDDHH.ALG

Notation	Description
YY	Last two digits of the year when the log file is created.
MM	Two-digit number of the month (01-12) when the log file is created.
DD	Two-digit day of the month (01-31) when the log file is created.
HH	Two-digit hour of the day (00-23) when the log file is created.

Configure alarm record logging

1. Open the Alarm Printer utility. Do the following:
   a. In the WindowMaker **Tools** view, expand **Applications**.
   b. Double-click **Alarm Printer**.
2. On the menu bar, select **Configure**. The **Configuration Settings** dialog box appears.
3. Select the **File Logging** tab.



4. Select the **Enable Alarm File Logging** checkbox to save alarm records to log files.
5. In the **Directory** box, type the path or browse to a folder location to save the alarm log files.
6. In the **Number of Hours to Cycle Filename** box, enter the number of hours worth of alarm records to save to an individual log file.

   Valid entries are 1 to 24. If your InTouch application runs continuously, select a file logging interval that creates a set of equal length daily log files. For example, setting **Number of Hours to Cycle Filename** to 6 creates 4 equal length daily log files.
7. In the **Starting at Hour** box, enter the starting hour to begin logging alarm records to a file each day.

   Valid entries are 0 to 23.

   For example, an oil refinery operates three daily work shifts. The first shift begins at 6:00 AM. Management wants alarm records logged for each shift. To do this, enter 8 for the **Number of Hours to Cycle Filename** option. Enter 6 for the **Starting at Hour (0-23)** option. The Alarm Printer creates a log file from 06:00 to 14:00, another from 14:00 to 22:00, and a third from 22:00 to 06:00.
8. In the **Keep Log Files for** box, enter the number of days to retain log files.

The Alarm Printer saves log files for the number of specified days plus the current day. The Alarm Printer deletes log files older than the retention period. To save log files indefinitely, enter 0.

9. In the **Log File Name Extension** box, accept the default ALG file name extension or assign another three-character extension to log files.

   If you use a .csv extension, you can import the log file directly into Excel or Notepad.

10. To remove spaces at the end of entries within a log file, select the **Remove Trailing Spaces in Log Entries** checkbox.

    You can also specify a field separator character placed at the end of each record in the log file.

11. Select the **Original Column Ordering** checkbox to maintain the same order from the alarm display to log file records.
12. Select **Apply**.

Save and load configuration files

When you start the Alarm Printer from WindowMaker, the **Alarm Printer** dialog box shows the default configuration settings. These configuration settings are saved in an alarm configuration file (.alc).

You can save your printer configuration settings to a file that can be loaded before printing alarms.

A specific alarm configuration file can be shown if it is opened in run time from the command prompt or by double-clicking its *.alc file name.

Create a new Alarm Printer configuration file

1. Open the Alarm Printer utility. Do the following:
   a. In the WindowMaker **Tools** view, expand **Applications**.
   b. Double-click **Alarm Printer**.
2. On the **File** menu, select **New** to show the Alarm Printer with its default values.
3. On the menu bar, select **Configure**. The **Configuration Settings** dialog box appears.
4. Configure the alarm settings.
5. On the **File** menu, select **Save**.

Edit an existing Alarm Printer configuration file

1. On the **File** menu, select **Open**.
2. Select the Alarm Printer Configuration file that you want to edit.
3. Edit the file.
4. On the File menu, select **Save**. Select **Save as** to save the changes to a new file without changing the existing file.

Print alarms with the alarm printer

Each query logs all of the alarms specified in the Alarm Printer configuration file (.alc) that is currently open. If no file has been specified, the settings currently selected during Alarm Printer configuration are used.

You can run multiple queries with Alarm Printer. Each query uses different parameters and is associated with a separate instance of the Alarm Printer. If two instances of Alarm Printer are running the same query, the entries

are duplicated.

While Alarm Printer is running, you can manually start or stop queries. Be sure that you have printing enabled.

Start an alarm query

- On the **Query** menu, select **Start/Stop**.

Stop an alarm query

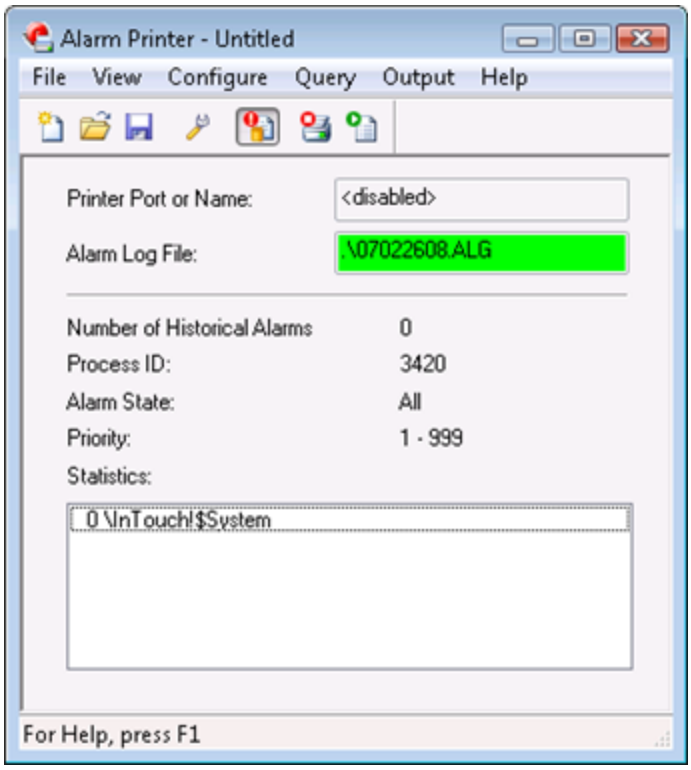
- On the **Query** menu, select **Start/Stop**.

Log alarms to a file

You can use the Alarm Printer to log alarm records to a file. You should have already configured logging from the **Configuration Settings** dialog box.

Log alarms to a file

1. Open the Alarm Printer utility.
   a. In the WindowMaker **Tools** view, expand **Applications**.
   b. Double-click **Alarm Printer**.
2. If necessary, configure the query to collect alarm records for logging.
3. Select the **File Logging** button.
4. Run an alarm query.



Alarm records collected by the query are written to the configured log file.

Start alarm printer with a specific configuration

You can automatically start the Alarm Printer and open a specific file when your system starts up by using the following command in a batch file:

```
ALMPRT.EXE MYQUERY.ALC
```

Where, `MYQUERY.ALC` is the name of the Alarm Printer configuration file that opens. Specifying the .exe file name extension is optional.

Make sure that the batch file switches to the folder where the InTouch HMI is installed.

To prevent the loss of any query data due to a system inadvertently being shut down and restarted, you can automatically start the Alarm Printer and automatically run a specific query by running the following command from a batch file:

```
ALMPRT.EXE -q MYQUERY.ALC
```

By using the `-q` in the command, your query runs automatically when the system starts up.

Control the alarm printer using scripting

You can use Alarm Printer functions in scripts to control alarm printing.

Alarm printer functions return an integer error code that indicates whether the function completed successfully or not. The following table shows the error codes.

Error Code	Error Message
0	Success
1	Instance not found or not running
2	Interface not initialized
3	Failure to access virtual memory
4	Invalid error code
5	Too many instances already running
6	Result string would be too long
7	Invalid instance index passed to the function
8	Failed to post the message to the alarm printer application
9	Failed to wait for a response from the alarm printer application
20	To priority must be equal to or greater than From priority
21	Invalid priority value
22	Invalid alarm state
23	Failed to execute the command because the query is running
24	Query string is not valid
25	Invalid query processing state
26	Invalid print state selector
27	Command received by the alarm printer window is not recognized
28	Query could not be started

Stop and start an alarm printer instance or query

Use the following functions to start and stop the Alarm Printer instance and the alarm query.

- APUStartInstance() Function

- APUStartQuery() Function
- APUStopInstance() Function
- APUStopQuery() Function

APUStartInstance() Function

Starts an instance of the Alarm Printer in a minimized state with values specified from a configuration file.

Category

View

Syntax

```
[Result=] APUStartInstance(sFilePath, iTagInstance);
```

Arguments

sFilePath

Full path to a configuration file (input string).

iTagInstance

Integer tag. The function returns an instance number to it if the function executes successfully.

Remarks

You can start up to sixteen instances of the Alarm Printer. This function writes the instance number (0 - 15) to the iTagInstance parameter. The instance number increments when a new Alarm Printer instance starts.

This instance number can be used by other Alarm Printer functions to identify the Alarm Printer instance.

This function returns 0 or, in the case of error, an error code.

The Alarm Printer program does not automatically begin processing alarms from the alarm memory. Use the **APUStartQuery()** function to begin processing data from the alarm memory for the instance.

Example

```
Status = APUStartInstance("c:\MyAlarmCfg\Area1Alarms.alc", Inst);
```

See Also

APUStartQuery(), APUStopInstance(), APUStopQuery()

APUStartQuery() Function

Sets the date and time limits for records to be processed from the alarm memory and then starts the query.

Category

View

Syntax

```
[Result=] APUStartQuery(iInstance, iYear, iMonth, iDay, iHour, iMinute);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iYear

The number of the year.

iMonth

The number of the month.

iDay

The day of the month.

iHour

The hour number.

iMinute

The minute number.

Remarks

An error occurs if you try to start a query when a query is already running.

If you set all date and time values to 0, all alarms are printed. This is because 0 is interpreted as January 01, 1900 at midnight. The time and dates specified are in local time. A value of -1 for the year sets the date to the current time that the command is processed.

Returns an integer error code.

Example

```
Status = APUStartQuery(Inst,2007,4,16,22,12);
```

See Also

APUStartInstance(), APUStopInstance(), APUStopQuery()

APUStopInstance() Function

Stops a specified instance of the Alarm Printer. Any further addition of records to be printed stops, any currently executing print query stops, and the instance of the program closes.

Category

View

Syntax

```
[Result=] APUStopInstance(iInstance);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

Remarks

Returns an integer error code.

Example

```
Status = APUStopInstance(5);
```

See Also

APUStartInstance(), APUStartQuery(), APUStopQuery()

APUStopQuery() Function

Requests the specified instance to stop running its query. The application remains running, but it does not process any queries. A call to **APUStartQuery()** can cause the instance to start querying.

Category

View

Syntax

```
[Result=] APUStopQuery(iInstance);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

Remarks

Returns an integer error code.

Example

```
Status = APUStopQuery(5);
```

See Also

APUStartInstance(), APUStartQuery(), APUStopInstance()

Query alarm query information

Use the following functions to retrieve query parameters based on tag priorities and values set within the Alarm Printer configuration file.

- APUGetAlarmGroupText() Function
- APUGetQueryFromPriority() Function
- APUGetQueryToPriority() Function
- APUGetConfigurationFilePath() Function
- APUGetPrinterJobCount() Function
- APUGetQueryAlarmState() Function
- APUGetQueryProcessingState() Function

APUGetAlarmGroupText() Function

Gets the Alarm Query alarm group text.

Category

View

Syntax

```
[Result=] APUGetAlarmGroupText(iInstance, sTagGroup);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

sTagGroup

Text - alarm group

Remarks

The initial alarm group text is read from the .alc configuration file that the Alarm Printer with the specified instance is using. The alarm group text is passed to the sTagGroup parameter into an InTouch message tag.

Returns an integer error code.

Example

The TagGroup message tag can contain the following value after the function is run: \intouch!$system

```
Status = APUGetAlarmGroupText(Inst,TagGroup);
```

See Also

APUGetConfigurationFilePath(), APUGetPrinterJobCount(), APUGetQueryAlarmState(), APUGetQueryFromPriority(), APUGetQueryProcessingState(), APUGetQueryToPriority()

APUGetQueryFromPriority() Function

Gets the From Priority value for a query.

Category

View

Syntax

```
[Result=] APUGetQueryFromPriority(iInstance, iTagPriority);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iTagPriority

An integer tag that receives the From Priority value.

Remarks

The initial priority is read from the .alc file that the Alarm Printer with the specified instance is using. The From Priority value is passed to the iTagPriority parameter into an InTouch integer tag.

Returns an integer error code.

Example

In this example, FromPri is an integer tag that contains the From Priority value, such as 1.

```
Status = APUGetQueryFromPriority(Inst, FromPri);
```

See Also

APUGetAlarmGroupText(), APUGetConfigurationFilePath(), APUGetPrinterJobCount(), APUGetQueryAlarmState(), APUGetQueryProcessingState(), APUGetQueryToPriority()

APUGetQueryToPriority() Function

Gets the To Priority from the query.

Category

View

Syntax

```
[Result=] APUGetQueryToPriority(iInstance,iPriority);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iPriority

An integer tag that receives the To Priority value.

Remarks

Another query cannot be running at the same time as the script that includes the APUGetQueryToPriority() function. The To Priority value is written to the iPriority parameter of the function, which is an integer tag.

Returns an integer error code.

Example

The integer tag ToPri receives the To Priority value, such as 999.

```
Status = APUGetQueryToPriority(Inst,ToPri);
```

See Also

APUGetAlarmGroupText(), APUGetConfigurationFilePath(), APUGetPrinterJobCount(), APUGetQueryAlarmState(), APUGetQueryFromPriority(), APUGetQueryProcessingState()

APUGetConfigurationFilePath() Function

Returns the full file path of the .alc configuration file used for a query.

Category

View

Syntax

```
[Result=] APUGetConfigurationFilePath(iInstance, sTagFilePath);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

sTagFilePath

A message tag to retrieve the name of the file path to the configuration file the Alarm Printer instance is using.

Remarks

The file path text is returned to an message tag that you can specify as sTagFilePath parameter of this function.

Returns an integer error code.

Example

The CfgFilePath message tag receives the file path of the configuration file associated with the Alarm Printer instance contained in the Inst integer tag. For example: c:\MyAlarmCfg\Area1Alarms.alc

```
Status = APUGetConfigurationFilePath(Inst, CfgFilePath);
```

See Also

APUGetAlarmGroupText(), APUGetPrinterJobCount(), APUGetQueryAlarmState(), APUGetQueryFromPriority(), APUGetQueryProcessingState(), APUGetQueryToPriority()

APUGetPrinterJobCount() Function

Returns the most recent Windows printer status job count for the printer used by this instance.

Category

View

Syntax

```
[Result=] APUGetPrinterJobCount(iInstance, iTagCount);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iTagCount

An integer tag that receives the count value.

Remarks

This function returns the count value in the iTagCount parameter, which is an integer tag.

The results are not current unless a query is running. The results are not current unless an alarm has been printed - the job count is typically updated when the printer is initially opened, and then each time an alarm line is printed.

The returned job count value is only valid for Windows printers and does not have much meaning for printers associated with a parallel or serial port.

Returns an integer error code.

Example

PJCount is an integer tag that receives the count value from the specified instance.

```
Status = APUGetPrinterJobCount(Inst, PJCount);
```

See Also

APUGetAlarmGroupText(), APUGetConfigurationFilePath(), APUGetQueryAlarmState(), APUGetQueryFromPriority(), APUGetQueryProcessingState(), APUGetQueryToPriority()

APUGetQueryAlarmState() Function

Returns the alarm state for the query.

Category

View

Syntax

```
[Result=] APUGetQueryAlarmState(iInstance, iTagState);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iTagState

An integer tag that receives the alarm state of the query associated with the specified instance. The value has following meanings:

0 = All

1 = Acknowledged

2 = Unacknowledged

Remarks

The initial alarm state is read from the .alc file. This function returns it in the iTagState parameter of the function which is an integer tag.

Returns an integer error code.

Example

AlmState is an integer tag that contains 0, 1 or 2.

```
Status = APUGetQueryAlarmState(Inst,AlmState);
```

See Also

APUGetAlarmGroupText(), APUGetConfigurationFilePath(), APUGetPrinterJobCount(), APUGetQueryFromPriority(), APUGetQueryProcessingState(), APUGetQueryToPriority()

APUGetQueryProcessingState() Function

Returns the status of the alarm query processing.

Category

View

Syntax

```
[Result=] APUGetQueryProcessingState(iInstance, iTagState);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iTagState

An integer tag that receives the processing state from the function. It has following meaning:

0 = Stop

1 = Start

Remarks

This function returns an integer value to the iTagState parameter, which is an integer tag.

Returns an integer error code.

Example

ProcState is an integer tag that is set to 0 if the query is not running, or 1 if the query is running.

```
Status = APUGetQueryProcessingState(Inst, ProcState);
```

See Also

APUGetAlarmGroupText(), APUGetConfigurationFilePath(), APUGetPrinterJobCount(), APUGetQueryAlarmState(), APUGetQueryFromPriority(), APUGetQueryToPriority()

Query instance information

Use the following functions to return the current status of an instance of the Alarm Printer.

- APUFindAlarmGroupInstance() Function
- APUFindFileInstance() Function
- APUFindPrinterInstance() Function
- APUGetInstanceCount() Function
- APUIsInstanceUsed() Function

APUFindAlarmGroupInstance() Function

Returns the first instance of the Alarm Printer using the specified alarm group string.

Category

View

Syntax

```
[Result=] APUFindAlarmGroupInstance(sGroup, iInstance);
```

Arguments

sGroup

The name of the alarm group to be found among the instances.

iInstance

An integer tag that receives the value of a found instance that uses the specified group name.

Remarks

This function returns the instance number to an integer tag as the instance parameter. The initial alarm group string is read from the .alc file. If no instance is found, the function returns 1 as the error code (no instance available) and writes 0 to the integer tag parameter.

Returns an integer error code.

Example

FoundInstance is an integer tag that receives the number of the first instance found that uses the $System as its query.

```
Status = APUFindAlarmGroupInstance("$System", FoundInstance);
```

See Also

APUFindFileInstance(), APUFindPrinterInstance(), APUGetInstanceCount(), APUIsInstanceUsed()

APUFindFileInstance() Function

Finds the first instance of the Alarm Printer using the specified .alc configuration file.

Category

View

Syntax

```
[Result=] APUFindFileInstance(sFilePath, iInstance);
```

Arguments

sFilePath

The path of the .alc configuration file for which the instance is to be found.

iInstance

An integer tag that receives the number of the instance.

Remarks

This function returns the instance number to an integer tag as the instance parameter. Use this function to initially obtain the desired instance of the Alarm Printer. The file path string match is not case-sensitive.

If no instance is found, the function returns 1 as the error code (no instance available) and writes 0 to the integer tag parameter.

Returns an integer error code.

Example

InstFound is an integer tag that receives the number of the first instance that uses the configuration file c:\MyAlarmCfg\Area1Alarms.alc.

```
Status = APUFindFileInstance("c:\MyAlarmCfg\Area1Alarms.alc", InstFound);
```

See Also

APUFindAlarmGroupInstance(), APUFindPrinterInstance(), APUGetInstanceCount(), APUIsInstanceUsed()

APUFindPrinterInstance() Function

Finds the first instance of the Alarm Printer using the specified printer name or port.

Category

View

Syntax

```
[Result=] APUFindPrinterInstance(sPrinter, iInstance);
```

Arguments

sPrinter

The name of the printer for which the instance is to be found.

iInstance

An integer tag that receives the number of the instance.

Remarks

This function returns the instance number to an integer tag as the instance parameter. Use this function to initially obtain the desired instance of an Alarm Printer. The printer name is stored and read from the .alc file. The printer name string match is not case-sensitive. If no instance is found, the function returns 1 as the error code (no instance available) and writes 0 to the integer tag parameter.

Returns an integer error code.

Example

FoundInst is an integer tag that receives the number of the first instance that uses LPT1 as printer name in its associated .alc file.

```
Status = APUFindPrinterInstance("LPT1", FoundInst);
```

See Also

APUFindAlarmGroupInstance(), APUFindFileInstance(), APUGetInstanceCount(), APUIsInstanceUsed()

APUGetInstanceCount() Function

Returns the number of running instances of the Alarm Printer, up to a maximum of 16 instances.

Category

View

Syntax

```
[Result=] APUGetInstanceCount(iCount);
```

Arguments

iCount

An integer tag that receives the number of instances.

Remarks

This function returns the number of instances to an integer tag as parameter. Any instances beyond the first sixteen running simultaneously are not dynamically controlled nor can their status be obtained.

Returns an integer error code.

Example

ICount is an integer tag. A run-time value of 7 means that there are currently seven instances of Alarm Printer running.

```
Status = APUGetInstanceCount( iCount );
```

See Also

APUFindAlarmGroupInstance(), APUFindFileInstance(), APUFindPrinterInstance(), APUIsInstanceUsed()

APUIsInstanceUsed() Function

Returns a discrete value that indicates the instance is currently in use.

Category

View

Syntax

```
[Result=] APUIsInstanceUsed(iInstance);
```

Arguments

iInstance
The number of an instance of Alarm Printer (0 to 15).

Remarks

The result is either 0 or 1:
0 = instance is not used.
1 = instance is used.

Example

InUse is a discrete tag that is set to true (1), if instance 5 is in use, or set to false (0), if instance 5 is not in use.

```
InUse = APUIsInstanceUsed(5);
```

See Also

APUFindAlarmGroupInstance(), APUFindFileInstance(), APUFindPrinterInstance(), APUGetInstanceCount()

Query printer information

Use the following functions to return the status of the Alarm Printer.

- APUGetPrinterName() Function
- APUGetPrinterStatus() Function

APUGetPrinterName() Function

Returns the Windows printer name or port name of the printer used by this instance.

Category

View

Syntax

```
[Result=] APUGetPrinterName(iInstance, sTagPrinter);
```

Arguments

iInstance

The number of the Alarm Printer instance (0 to 15).

sTagPrinter

A message tag that receives the printer name or port name of the configuration associated with the specified instance.

Remarks

This function returns the value NONE if no printer is configured for an instance. The printer name is stored and read from the .alc file. This function returns the printer name or port name to a message tag as the sTagPrinter parameter.

Returns an integer error code.

Example

PrtName is a message tag that receives the printer name or port name of the configuration associated with instance 3 of Alarm Printer.

```
Status = APUGetPrinterName(3,PrtName);
```

See Also

APUGetPrinterStatus()

APUGetPrinterStatus() Function

Returns the most recent status of the Windows printer used by this instance.

Category

View

Syntax

```
[Result=] APUGetPrinterStatus(iInstance, iSelector, iTagStatus);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iSelector

An integer value specifying the following:

0 = Get status for Alarm Printer Error

1 = Get status for Alarm Printer No Paper

2 = Get status for Alarm Printer Offline

3 = Get status for Alarm Printer Overflow

iTagStatus

An integer or real tag that receives the status of the printer associated with specified instance number and the type of selection made by the iSelector parameter.

Remarks

This function returns the printer status to an integer or real tag as the iTagStatus parameter. The results are not current unless a query is running and an alarm has been printed. The status typically updates when the printer initially opens, and then each time an alarm line prints.

This status information is being queried to the printer based on Microsoft or Windows driver standards. Not all printer manufacturers follow these standards. Therefore, not all printers return status information.

Returns an integer error code.

Example

PrtStat is an integer tag that receives the "Printer Offline" status from the printer associated with instance 5.

```
Status = APUGetPrinterStatus(5, 2, PrtStat);
```

See Also

APUGetPrinterName()

Set alarm query information

Use the following functions to set parameters used in the Alarm Printer query.

- APUSetAlarmGroupText() Function
- APUSetQueryAlarmState() Function
- APUSetQueryFromPriority() Function
- APUSetQueryToPriority() Function
- APUSetTimeoutValues() Function

APUSetAlarmGroupText() Function

Sets the Alarm Query alarm group text.

Category

View

Syntax

```
[Result=] APUSetAlarmGroupText(iInstance, sGroup);
```

Arguments

iInstance
The instance of Alarm Printer (0 to 15).
sGroup
Alarm group text.

Remarks

A query cannot be running for this function to succeed.
Returns an integer error code.

Example

This example sets the query of the Alarm Printer instance 1 to \InTouch!GroupA.

```
Status = APUSetAlarmGroupText(1, "\InTouch!GroupA");
```

See Also

APUSetQueryAlarmState(), APUSetQueryFromPriority(), APUSetQueryToPriority(), APUSetTimeoutValues()

APUSetQueryAlarmState() Function

Sets the alarm state for the query.

Category

View

Syntax

```
[Result=] APUSetQueryAlarmState(iInstance, iState);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iState

An integer with following possible values:

0 = All

1 = Acknowledged

2 = Unacknowledged

Remarks

A query cannot be running simultaneously while running a script using the **APUSetQueryAlarmState()** function.

Returns an integer error code.

Example

This example sets the query of the Alarm Printer instance 3 to query for acknowledged alarms only.

```
Status = APUSetQueryAlarmState(3, 1);
```

See Also

APUSetAlarmGroupText(), APUSetQueryFromPriority(), APUSetQueryToPriority(), APUSetTimeoutValues()

APUSetQueryFromPriority() Function

Sets the lower boundary or from priority of an alarm query.

Category

View

Syntax

```
[Result=] APUSetQueryFromPriority(iInstance, iPriority);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iPriority

An integer value for the From Priority (1 to 999).

Remarks

A query cannot be running for this function to succeed.

Returns an integer error code.

Example

This example sets the From Priority value of the query associated with the instance value of the Inst integer tag to the value of the FromPri integer tag.

```
Status = APUSetQueryFromPriority(Inst, FromPri);
```

See Also

APUSetAlarmGroupText(), APUSetQueryAlarmState(), APUSetQueryToPriority(), APUSetTimeoutValues()

APUSetQueryToPriority() Function

Sets the To Priority for the query.

Category

View

Syntax

```
[Result=] APUSetQueryToPriority(iInstance, iPriority);
```

Arguments

iInstance

The instance of Alarm Printer (0 to 15).

iPriority

An integer value for the To Priority in the range of 1 to 999. It should also be set higher than or equal to the From Priority value of the same query of the specified instance.

Remarks

The To priority must be equal to or greater than the From priority to set a valid alarm priority range. A script containing the **APUSetQueryToPriority()** function cannot run at the same time as a query.

Returns an integer error code.

Example

This example sets the To Priority value of the query associated with the instance 0 to 240.

```
Status = APUSetQueryToPriority(0,240);
```

See Also

APUSetAlarmGroupText(), APUSetQueryAlarmState(), APUSetQueryFromPriority(), APUSetTimeoutValues()

APUSetTimeoutValues() Function

The **APUSetTimeoutValues()** function sets time-out intervals in seconds. A time-out interval controls how many errors caused by memory access or failing to obtain valid responses are observed while program is running.

The default memory access time-out is two seconds. The default short response wait time is 10 second and the default long response wait time is 20 seconds.

Category

View

Syntax

```
[Result=] APUSetTimeoutValues(iMemory, iShort, iLong);
```

Arguments

iMemory

Integer - access time out

iShort

Short response wait time (integer)

iLong

Long response wait time (integer)

Example

```
Status = APUSetTimeoutValues(iMemory,iShort,iLong);
```

See Also

APUSetAlarmGroupText(), APUSetQueryAlarmState(), APUSetQueryFromPriority(), APUSetQueryToPriority()

Handle alarm printer errors

Use the APUTranslateErrorCode() function to convert an Alarm Printer error code to an error message.

APUTranslateErrorCode() Function

Converts an error code returned by one of the APU functions into an English string that briefly describes the error code.

Category

View

Syntax

```
[Result=] APUTranslateErrorCode(iErrorCode, sTagMessage);
```

Arguments

iErrorCode

An integer error code, normally returned by most other APU functions.

sTagMessage

A message tag that receives the error message.

Remarks

This function returns the error message to a message tag as sTagMessage parameter. This function can fail if an unknown error code is passed.

Returns an integer error code.

Example

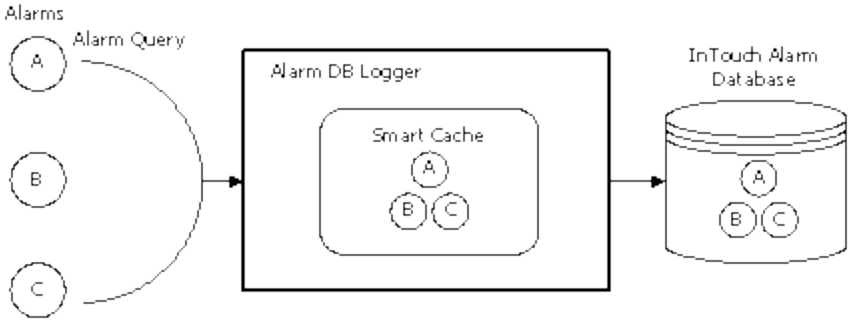
This example sets the message tag errmsg to "No instance available." if there is no instance 15 of Alarm Printer currently running.

```
Status = APUTranslateErrorCode(APUSetAlarmGroupText(15,"$system"), ErrMsg);
```

Record alarms into an alarm database

The InTouch Distributed Alarm system includes the Alarm DB Logger utility that logs alarms and events to the alarm database.

Alarm DB Logger is an alarm consumer. You configure it with one or more queries to select alarms from InTouch alarm providers. The alarms selected by the queries are stored in a transient memory cache, called the Smart Cache. The Alarm DB Logger writes the contents of the Smart Cache to the alarm database as alarm and event records at a periodic interval.



The Alarm DB Logger can auto-reconnect. When the connection to the database is lost, the logger checks for the database connection at regular intervals. Logging resumes when the connection to the alarm database is re-established.

The Alarm DB Logger reports all errors whether running as a service or a normal application to the Logger.

SQL Server accounts for Alarm DB Logger Manager

The Alarm DB Logger Manager creates and uses the following accounts in the SQL Server database:

Account	Password
wwAdmin	wwadmin
wwPower	wwpower
wwUser	wwuser

Use the Alarm DB Logger Manager

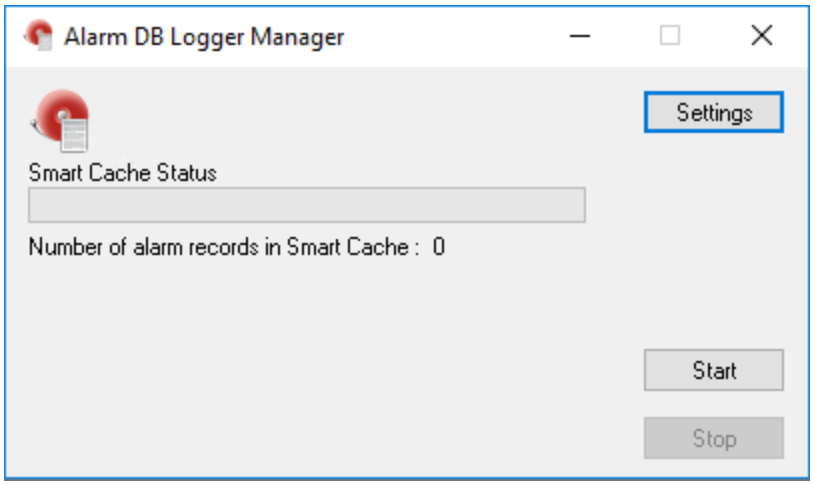
Use the Alarm DB Logger Manager to start and stop logging operations. The Alarm DB Logger Manager can start as a service or a normal application.

You configure alarm logging using the Alarm DB Logger configuration wizard, which you start from within the Alarm DB Logger Manager.

The Alarm DB Logger Manager shows the percentage fill of the Smart Cache with alarm records. Alarms are cached when the SQL Server connection is down or when alarms occur faster than Alarm DB Logger can log them to the alarm database.

Open the Alarm DB Logger Manager

1. In the **Tools** view, expand **Applications**.
2. Double-click **Alarm DB Logger Manager**. The Alarm DB Logger Manager appears.

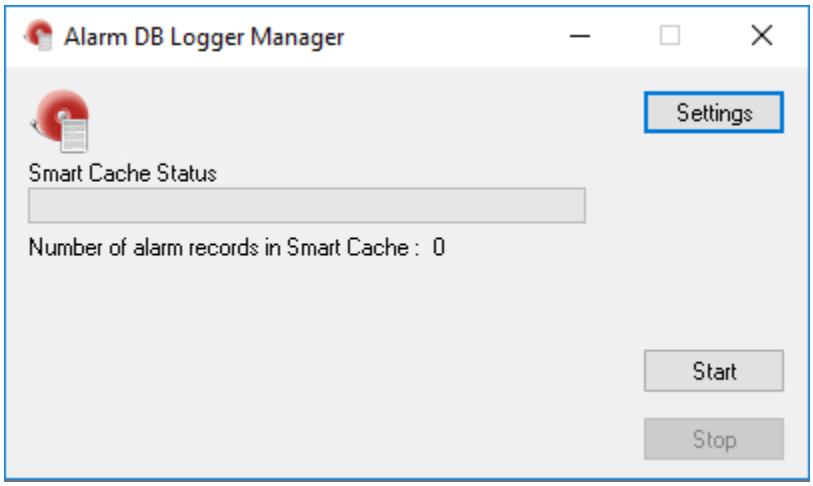


Start and stop alarm database logging

The Alarm DB Logger writes records to the alarm database. It starts and runs either as a service or a normal application (depending upon the running mode you select when you configure the Alarm DB Logger).

Start or stop alarm logging

1. In the **Tools** view, expand **Applications**.
2. Double-click **Alarm DB Logger Manager**. The Alarm DB Logger Manager appears.



   The **Smart Cache Status** shows the percentage of the in-memory cache holding alarm records.
3. Select **Start** to begin the alarm logging process.
4. Select **Stop** to end the alarm logging process.

Configure alarm database logging

To log InTouch alarm and event data to the alarm database, do the following from within the Alarm DB Logger Manager:

- Configure the connection to the alarm database

- Select which alarms to log to the alarm database
- Set the interval to log records to the alarm database
- Select which method to run the Alarm DB Logger

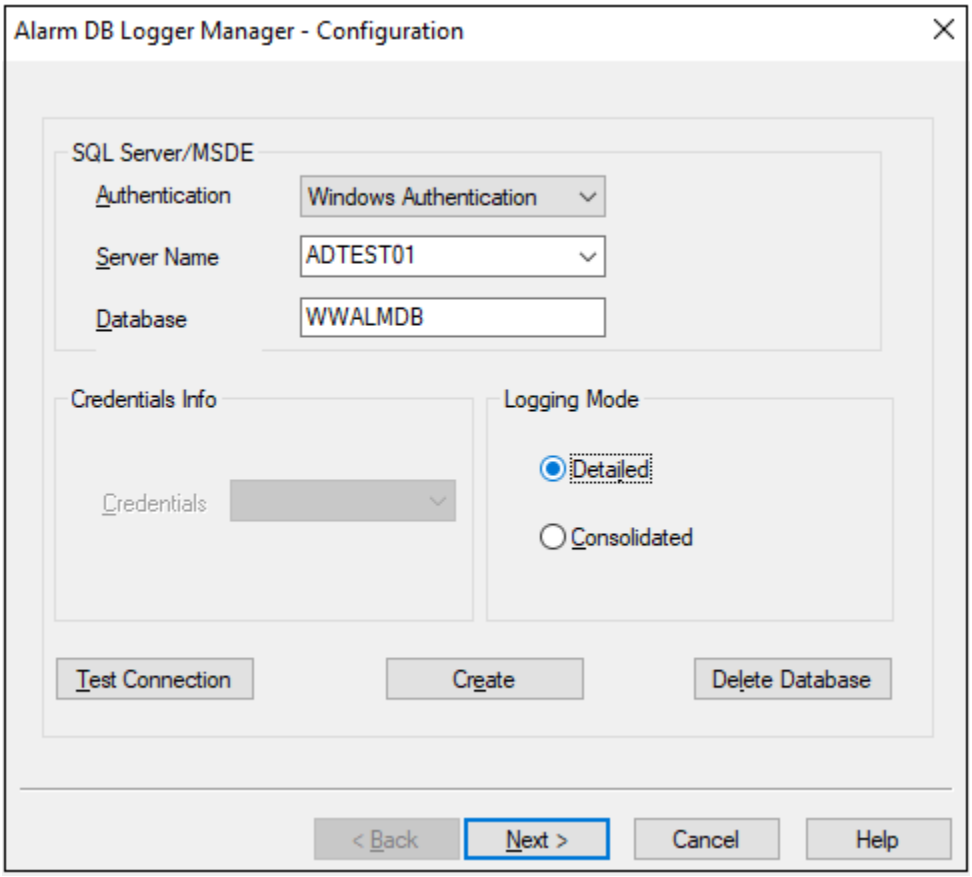
Establish the database connection

You must establish a connection between the Alarm DB Logger and the alarm database.

The Alarm DB Logger works with SQL Server and Windows authentication. The SQL Server installation must be set to mixed mode.

Configure the database connection

1. Open the Alarm DB Logger Manager. Do the following:
   a. In the Tools view, expand **Applications**.
   b. Double-click **Alarm DB Logger Manager**.
2. Select **Settings**. The **Alarm DB Logger Manager - Configuration** wizard appears.



3. In the **SQL Server/MSDE** section, do the following:
   a. In the **Authentication** list, select the authentication method: SQL Server Authentication or Windows Authentication (default).

   Note: Windows authentication can provide better application security than SQL Authentication. If you switch from Windows Authentication to SQL Authentication, a dialog will appear recommending that you use Windows Authentication for this reason. If you choose to ignore this warning and proceed with SQL

Authentication, select **OK**. A similar message will be logged in the OCMC Log Viewer.

b. In the **Server Name** box, enter the node name of the computer where the alarm database is installed.

c. In the **Database** box, type the name of the InTouch alarm database.

4. In the **Credentials Info** section, from the **Credentials** drop-down list, select the credentials for authentication.

Note:
- The Credentials drop-down is enabled only when the SQL Server Authentication type is selected. The Windows Authentication method uses the credentials of the user currently logged in, and disables the user name and password fields.
- For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credentials Manager in AVEVA™ InTouch HMI Application Development.

5. In the **Logging Mode** area, configure how records are stored. Do either of the following:

a. Select **Detailed** to store a separate record for each alarm condition (in alarm, acknowledged, and returned to normal).

b. Select **Consolidated** to store all states of an alarm (in alarm, acknowledged, and returned to normal) in a single record with time stamps for each transition.

6. Select **Create** to create the database, if required.

7. Select **Test Connection** to verify connectivity to the alarm database. A message indicates a successful connection to the database.

8. Select **Next** to configure which alarms to log. See Configure which alarms to log for detailed instructions.

Configure which alarms to log

In the second page of the Alarm DB Logger Manager configuration wizard, you define one or more queries to select alarms from InTouch or Galaxy alarm providers. You also select a range of alarm priority values that are part of the database query.

Use the following query syntax for remote nodes:

\\NodeName\Provider!AlarmGroup

Use the following query syntax for the local node:

\Provider!AlarmGroup

Example:

\\ProdSvr\InTouch!$System

The following is an example of using a Galaxy alarm provider. This query gives you all alarms from a specific area.

\Galaxy!Area

Note: The Alarm DB Logger Manager only supports 50 Queries and 3072 characters.

Configure which alarms to log

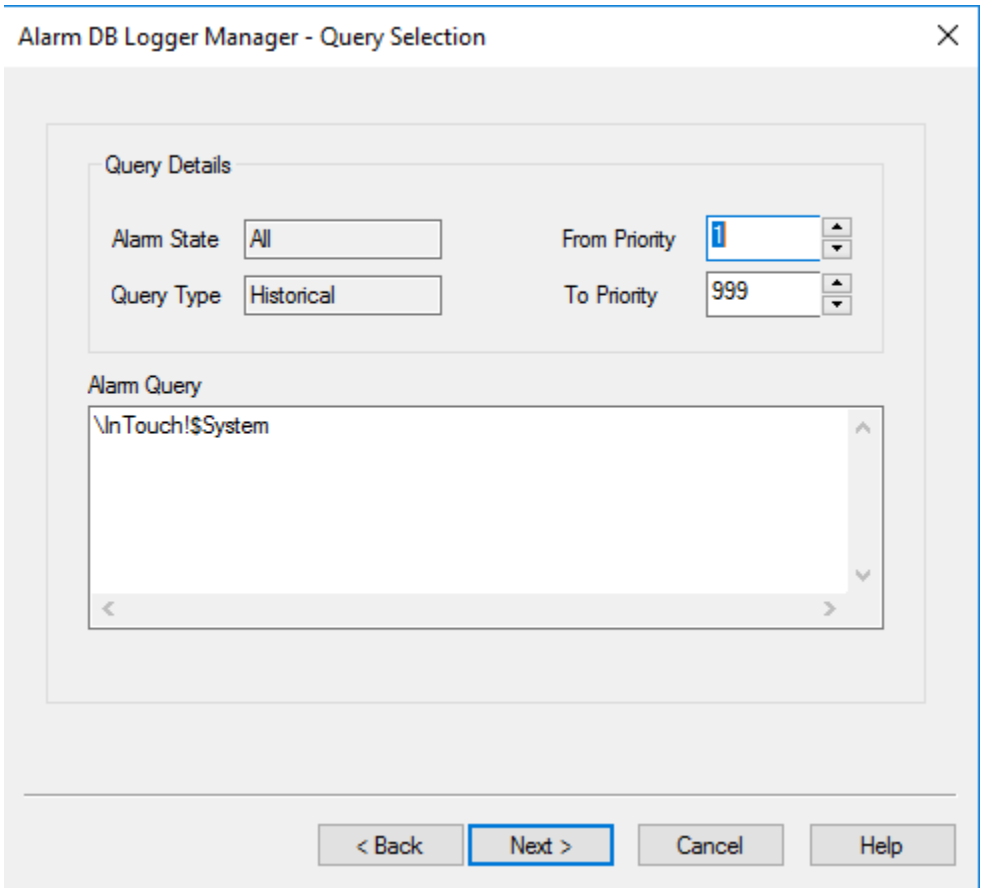
1. Open the Alarm DB Logger Manager and start the configuration wizard. Do the following:

a. In the **Tools** view, expand **Applications**.

b. Double-click **Alarm DB Logger Manager**.

c. Select **Settings**. The **Alarm DB Logger Manager - Configuration** wizard appears.

2. Select **Next**. The **Alarm DB Logger Manager - Query Selection** page appears.



The read-only **Alarm State** box shows the alarm state for logging. The read-only **Query Type** box shows the type of query.

3. In the From Priority box, enter the starting value of the alarm priority range.
4. In the To Priority box, enter the ending value of the alarm priority range.
5. In the Alarm Query box, type the alarm queries that you want to use to store or retrieve data from the alarm database.

Note: Area name with 128 characters is not supported by Historian.

6. Select **Next** to configure the logging interval. See Configure the logging interval.

Configure the logging interval

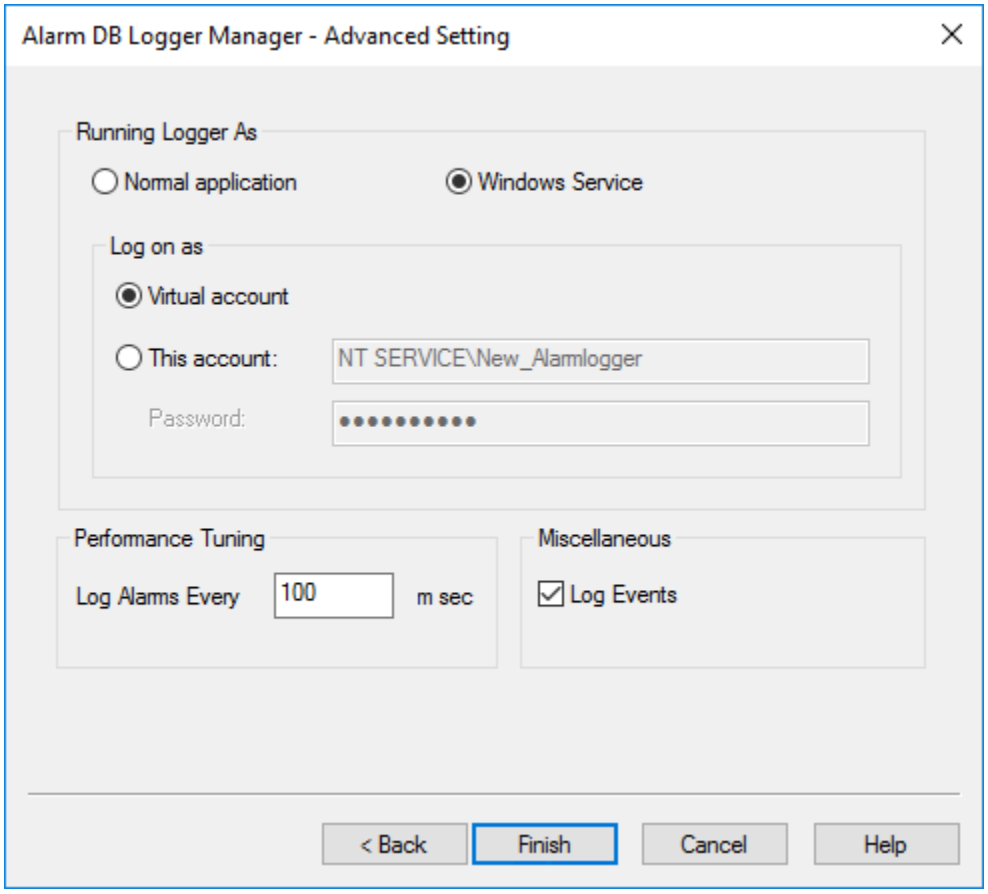
In the third page of the Alarm DB Logger Manager configuration wizard, you configure advanced query settings. You can log events to the alarm database. You can also tune the performance of alarm logging by setting the frequency at which the alarm records are written from internal alarm memory to the database.

The logging interval is not the same as the reconnect rate for the SQL Server. The logging interval is the interval at which alarms should be read. The reconnect rate depends on the time out for a connection attempt associated with the SQL Server.

Setting the logging interval too low impacts system performance.

Configure the logging interval

1. Open the Alarm DB Logger Manager and start the configuration wizard. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Logger Manager**.
   c. Select **Settings**. The **Alarm DB Logger Manager - Configuration** wizard appears.
2. Select **Next**. The **Alarm DB Logger Manager - Query Selection** page appears.
3. Select **Next**. The **Alarm DB Logger Manager - Advanced Setting** page appears.



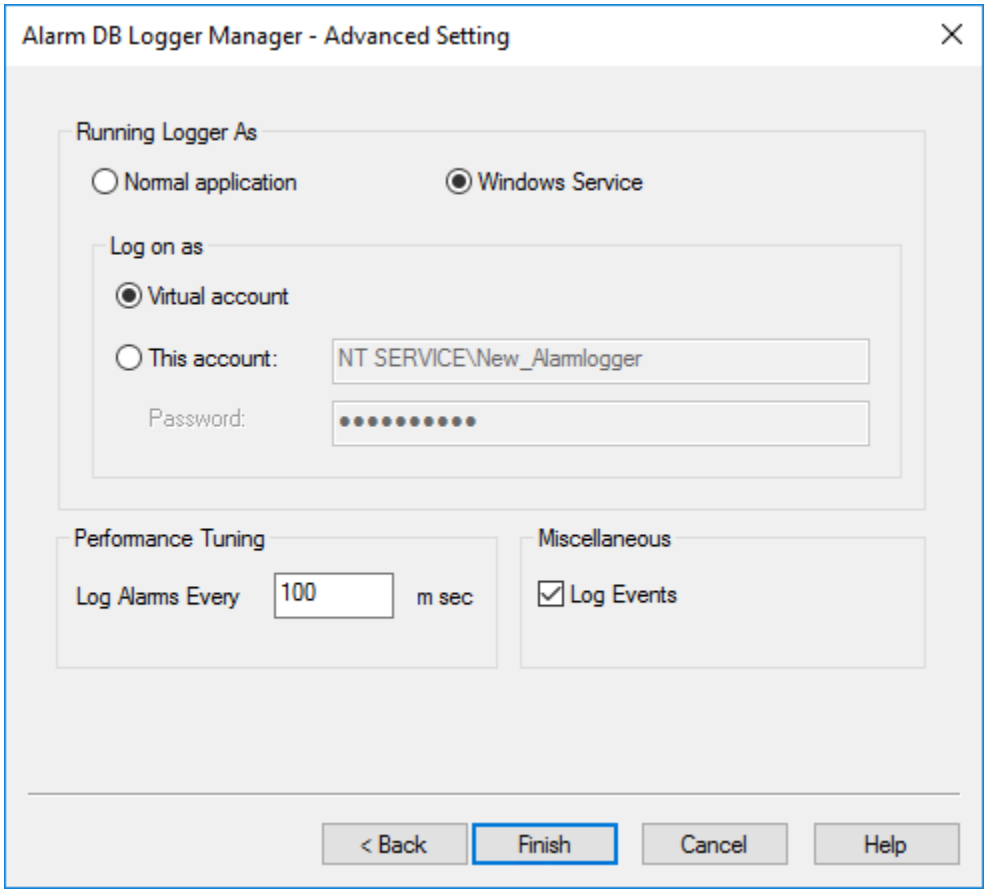
4. Select the **Log Events** checkbox if you want to store InTouch event records to the alarm database. You can also store events coming from an Application Server Galaxy.
5. In the **Performance Tuning** area, type the interval in milliseconds at which alarm records are written to the alarm database.
6. Select **Finish**.

Configure Alarm DB Logger as a service

You can configure the Alarm DB Logger to run as a Windows service. To reduce the security exposure of running the Alarm DB Logger with administrator privileges, the user account permissions have been set to non-interactive to run the Alarm DB Logger as a service.

1. Log in to the computer as an administrator.

2. Open the Alarm DB Logger Manager and start the configuration wizard. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Logger Manager**.
   c. Select **Settings**. The **Alarm DB Logger Manager - Configuration** wizard appears.
3. Select **Next**. The **Alarm DB Logger Manager - Query Selection** page appears.
4. Select **Next**. The **Alarm DB Logger Manager - Advanced Setting** page appears.



5. In the **Running Logger As** area, select either **Normal Application** or **Windows Service**.

   The credentials will depend on the authentication method selected in the **Alarm DB Logger Manager - Configuration** wizard page.
6. In the **Log on as** area, select the login account based on the following criteria, and select **Finish**.

Authentication Method	Running Logger As	Log on as	User Name	Password
SQL Server Authentication	Windows Service	Virtual account	--	--
Windows Authentication	Windows Service	Virtual account	--	--
Windows Authentication	Windows Service	This account	<username>	<password>

Note: For more information on Virtual account, see Use virtual accounts.

By default, selecting **This account** will display the user name of the user currently logged in, in which case password is not required. If you change the user name then you must provide a password.

You must provide a user account with appropriate permissions, to complete the configuration.

The next time you launch the Alarm DB Logger it will pre-fill the configuration details.

Use virtual accounts

Choosing virtual account, allows you an additional layer of security. Selecting this option, will start the Alarm DB Logger with an NT Service account called New_AlarmLogger.

The Virtual Account is enabled for both SQL and Windows Authentication. For earlier systems configured with 'LocalSystem', the account will be migrated as follows:

- If previous system is configured with "LocalSystem" in "This account" OR "Local System account" option, then the "This account" with be prefilled with "LocalSystem" as the user.
- If previous system is configured with "This account" as "NT Service\New_AlarmLogger", then it will migrated to "Virtual Account".

Alarm database views

You can use a set of SQL Server database views to easily query past and current alarm and event occurrences.

A database view is a single logical table that combines information from multiple, underlying database tables. Database views are often called as virtual database tables because they can be queried using standard SQL statements, just as if they were real tables. When a view is queried, it returns a set of records (or rows). Each row has several columns of information that contain the data for the record.

You can use the alarm database views to perform complex queries. For example, you can retrieve alarm records for all HiHi alarms that occurred during a particular time span in a particular area of the plant. Or, you can retrieve all data change events logged by a certain alarm provider node.

All strings are Unicode in the views.

Alarm history view

The v_AlarmHistory view contains all historical alarms and alarm transition events that occurred over a selected time range. The query specifies start and end date and time (the EventStamp or EventStampUTC column). The returned records include alarm origination, alarm acknowledge, alarm enable, alarm disable, and alarm return-to-normal events.

The following table describes the view columns:

Column Name	Datatype	Description
EventStamp	Datetime	Date and time of alarm event (in local time of database).
AlarmState	nChar	State of alarm: one of UNACK, UNACK_RTN, ACK, ACK_RTN,

Column Name	Datatype	Description
		LATCHED.
TagName	nChar	Name of the object that generated the alarm, such as TIC101.
Description	nVarchar	Description string of the alarm. Can default to object description (or comment in the InTouch HMI). Or acknowledge comment for acknowledged records.
Area	nChar	Name of the Area or Group for the alarm.
Type	nChar	Type of alarm, such as Hi, HiHi, ROC, PV.HiAlarm.
Value	nChar	Value of alarm variable at time of alarm.
CheckValue	nChar	Value of alarm limit at time of alarm.
Priority	Integer	Alarm priority.
Category	nChar	Alarm class or alarm category. Such as Value, Dev, ROC, Process, Batch, System, and so on.
Provider	nChar	Provider of alarm: node/InTouch, or GalaxyName.
Operator	nChar	Name of the operator.
DomainName	nChar	Name of the domain.
UserFullName	nChar	Full name of user in operator (for example, Joseph P. Smith).
UNACKDuration	Float	The time between the most recent alarm transition (alarm or sub-state) and the acknowledgement, if any.
User1	Float	User-defined field number 1.
User2	Float	User-defined field number 2.
User 3	nChar	User-defined field, string.

Column Name	Datatype	Description
EventStampUTC	DateTime	UTC date/ time of alarm event.
Millisec	Small Int	Fractional seconds for event stamp in increments of 0.1 msec.
OperatorNode	nvarchar(32)	Name of the node where the operator acknowledged the alarm.

The v_AlarmHistory2 view is a variation of the v_AlarmHistory view.

Column Name	Datatype	Description
EventStamp	Datetime	Date and time of alarm event (in local time of database).
AlarmState	nChar	State of alarm: one of UNACK, UNACK_RTN, ACK, ACK_RTN, ACK_ALM, UNACK_ALM, LATCHED
TagName	nChar	Name of the object that generated the alarm, such as TIC101.
Description	nVarchar	Description string of the alarm. Can default to object description (or comment in InTouch). Or acknowledge comment for acknowledgement records.
Area	nChar	Name of the Area or Group for the alarm.
Type	nChar	Type of alarm, such as Hi, HiHi, ROC, PV.HiAlarm.
Value	nChar	Value of alarm variable at time of alarm.
CheckValue	nChar	Value of alarm limit at time of alarm.
Priority	Integer	Alarm priority.
Category	nChar	Alarm class or alarm category. Such as Value, Dev, ROC, Process, Batch, System, and so on.
Provider	nChar	Provider of alarm: node/InTouch, or GalaxyName.

Column Name	Datatype	Description
Operator	nChar	Name of operator: JoeR (if any).
DomainName	nChar	Name of domain.
UserFullName	nChar	Full name of user in operator (for example, Joseph P. Smith).
AlarmDuration	Float	The time between onset of the alarm and return to normal.
User1	Float	User-defined field number 1.
User2	Float	User-defined field number 2.
User 3	nChar	User-defined field, string.
EventStampUTC	DateTime	UTC date/ time of alarm event.
Millisec	Small Int	Fractional seconds for event stamp in increments of 0.1 msec.

Example query of alarm history view

This example selects all records from the Alarm History view:

```
SELECT * FROM v_AlarmHistory
```

This example selects all records from the Alarm History view with a priority greater than 100:

```
SELECT * FROM v_AlarmHistory WHERE Priority >100
```

Event history view

The v_EventHistory database view lists all historical events that occurred over a selected time range. The query client specifies the starting and ending date and time range. The returned records include all non-alarm events.

Column Name	Datatype	Description
EventStamp	Datetime	Date and time of event.
TagName	nChar	Name of the object that generated the event, such as Pump1.
Description	nVarChar	Description string of the event. Can default to object description, or comment in InTouch.
Area	nChar	Name of the Area or Group for the event.

Column Name	Datatype	Description
Type	nChar	Type of event, such as "Operator data change", "Startup", and so on.
Value	nChar	New Value (if any).
CheckValue	nChar	Old Value (if any).
Category	nChar	Event category or class, such as Value, Process, Batch, System, and so on.
Provider	nChar	Generator of event, such as node/ InTouch, or View Engine name for user change.
Operator	nChar	Name of operator1: JoeR (if any).
DomainName	nChar	Name of domain.
UserFullName	nChar	Full name of user in operator (for example, Joseph P. Smith).
User1	Float	User-defined field number 1.
User2	Float	User-defined field number 2.
User 3	nChar	User-defined field, string.
EventStampUTC	DateTime	UTC date/ time of event.
Millisec	Small Int	Fractional seconds for event stamp in increments of 0.1 msec.

Alarm event history view

The v_AlarmEventHistory database view provides a historical list of all events and alarms that occurred over a selected time range. The query client specifies start and end date and time. The returned records include all alarms and events. This view combines the alarm view and event view into one and represents the union of the records from those views.

Column Name	Datatype	Description
EventStamp	Datetime	Date and time of event.
AlarmState	nChar	State of alarm: one of UNACK, UNACK_RTN, ACK, ACK_RTN, LATCHED. Does not apply for events.

Column Name	Datatype	Description
TagName	nChar	Name of the object that generated the alarm, such as TIC101.
Description	nVarchar	Description string of the alarm/event. Can default to object description (or comment in InTouch). Or acknowledge comment for acknowledged records.
Area	nChar	Name of the Area or Group for the alarm.
Type	nChar	Type of alarm or event, such as Hi, HiHi, ROC, PV.HiAlarm, Operator data change, and so on.
Value	nChar	Value of alarm variable at time of alarm.
CheckValue	nChar	Value of alarm limit at time of alarm, or old value for event.
Priority	Integer	Alarm priority.
Category	nChar	Alarm or event class, or alarm category, such as Value, Process, Batch, System, and so on.
Provider	nChar	Provider of alarm, such as node/InTouch, or GalaxyName.
Operator	nChar	Name of acknowledgement operator or data change operator.
DomainName	nChar	Name of domain.
UserFullName	nChar	Full name of user in operator (for example, Joseph P. Smith).
UNACKDuration	Float	Number of milliseconds from the most recent alarm transition to ACK.
User1	Float	User-defined field number 1.
User2	Float	User-defined field number 2.
User 3	nChar	User-defined field, string.

Column Name	Datatype	Description
EventStampUTC	DateTime	UTC date/ time of event.
Millisec	Small Int	Fractional seconds for event stamp in increments of 0.1 msec.

The v_AlarmEventHistory2 view is a variation of the v_AlarmEventHistory view.

Column Name	Datatype	Description
EventStamp	Datetime	Date and time of event.
AlarmState	nChar	State of alarm: one of UNACK, UNACK_RTN, ACK, ACK_RTN, LATCHED. Does not apply for events.
TagName	nChar	Name of the object that generated the alarm, such as TIC101.
Description	nVarchar	Description string of the alarm/ event. Can default to object description (or comment in InTouch). Or acknowledge comment for acknowledged records.
Area	nChar	Name of the Area or Group for the alarm.
Type	nChar	Type of alarm or event, such as Hi, HiHi, ROC, PV.HiAlarm, Operator data change, and so on.
Value	nChar	Value of alarm variable at time of alarm.
CheckValue	nChar	Value of alarm limit at time of alarm, or old value for event.
Priority	Integer	Alarm priority.
Category	nChar	Alarm or event class, or alarm category, such as Value, Process, Batch, System, and so on.
Provider	nChar	Provider of alarm, such as node/ InTouch, or GalaxyName.

Column Name	Datatype	Description
Operator	nChar	Name of acknowledgement operator or data change operator.
DomainName	nChar	Name of domain.
UserFullName	nChar	Full name of user in operator (for example, Joseph P. Smith).
AlarmDuration	Float	Number of milliseconds from the onset of alarm to the return to normal (RTN).
User1	Float	User-defined field number 1.
User2	Float	User-defined field number 2.
User 3	nChar	User-defined field, string.
EventStampUTC	DateTime	UTC date/ time of event.
Millisec	Small Int	Fractional seconds for event stamp in increments of 0.1 msec.

Example query of alarm event history view

The following example selects the data in the TagName, Area and Type columns for all records that have a tagname of MyTag1, an alarm state of ACK_RTN or ACK. The results are ordered by the alarm provider.

```
SELECT TagName, Area, Type FROM v_AlarmEventHistory
   WHERE TagName='MyTag1'
      AND (AlarmState='ACK_RTN' OR AlarmState='ACK')
      ORDER BY Provider
```

Alarm database stored procedures

You can use a set of SQL Server stored procedures to easily retrieve information for analysis and reporting of alarms and events.

A stored procedure is a collection of SQL statements that perform a specific function and return data from the database. Stored procedures can accept input parameters that govern how they are to operate, and they return data in the form of a result set.

The returned results are a set of records and appear as a SQL record set very similar to a SQL Server database view. Stored procedures can improve performance because they exist in the database with their embedded SQL statements and reduce round-trips back to the client.

For more information about stored procedures, including how to view the definitions for them, see your Microsoft SQL Server documentation.

Call a stored procedure

Call a stored procedure using the EXECUTE statement.

```
EXECUTE sp_AlarmCounter @StartDate='2007-01-01', @EndDate='2007-03-31', @Tagname = 'tag1',
@Type = 'LO', @Provider = 'WW21353\InTouch', @Comment = 'SSAADD'
```

In this example, the StartDate and EndDate parameters are required. The remaining parameters are optional. If you do not provide a value for a parameter, it is not used to filter the result set.

If a stored procedure includes a date/time variable, you can use any valid format specified in the SQL Server documentation.

AlarmCounter database stored procedure

This stored procedure returns the number of times a unique alarm occurred in a time interval. A unique alarm is identified by its TagName, Provider, Alarm Type, and Category.

The counter only applies to the number of alarm originations (not transitions such as acknowledges or returns-to-normal). So, for example, if an alarm occurred and then was acknowledged and then returned to normal, the count for that alarm is only 1 (not 3).

The query must specify the starting and ending date and time. It has five optional parameters: TagName, Class, Type, Provider and Comment.

An example question answered by this view: How many times did object TIC101 (TagName) on provider Node1|InTouch (Provider) go into a Value alarm (Category) that was HiHi (Type) during a time span?

Note: The AlarmCounter stored procedure applies only to the Detailed logging mode and is not supported by Consolidated logging.

```
sp_AlarmCounter
```

Column Name	Datatype	Description
TagName	Nchar	Name of the object that generated the alarm, such as TIC101.
GroupName	Nchar	Name of the area or group for the alarm
AlarmType	Nchar	Type of alarm such as Hi, HiHi, ROC, PV, HiAlarm.
AlarmClass	Nchar	Alarm class or alarm category such, as Value, Process, Batch, etc.
AlarmCount	Integer	Number of onsets of the alarm during the time range. If an alarm occurs prior to the starting date and time, it is not included in the count.
Priority	Integer	Alarm priority.

Column Name	Datatype	Description
Provider	Nchar	Provider of alarm, such as node/ InTouch, or GalaxyName.
Comment	Nchar	The alarm comment.

An example query is:

```
EXECUTE sp_AlarmCounter @StartDate='2007-01-01 23:23:23', @EndDate='2007-03-31 23:23:23',
@Tagname = '$NewAlarm'
```

EventCounter database stored procedure

This database stored procedure provides the count of the number of events of a certain type on a certain tag that occurred in a time interval. The query client must specify start and end date and time. It has three optional parameters: TagName, Provider and Comment. The counter applies only to non-alarm events. The purpose of this view is to show frequency of each event. For example, how many times did the pump turn on? The TagName, Provider, Category and Type are used to uniquely identify an event and do the counting.

```
sp_EventCounter
```

Column Name	Datatype	Description
TagName	NChar	Name of the object that generated the alarm, such as TIC101
Area	NChar	Name of the Area or Group for the alarm.
Type	NChar	Type of event.
Category	NChar	Alarm class or alarm category, such as Value, Process, Batch, etc.
EventCount	Integer	Number of times the event of this Type for the TagName has occurred in the specified time range.
Provider	NChar	Provider of event: node/InTouch, or GalaxyName.
Comment	NChar	The event comment.

An example query is:

EXECUTE sp_EventCounter @StartDate='2007-01-01 23:23:23', @EndDate='2007-03-31 23:23:23', @Tagname = '$NewAlarm'

View recorded alarms

You use the Alarm DB View ActiveX control to visualize data from the alarm database. Use this control to show all

alarm and event information generated from an InTouch application during run time.

For this control, you can configure:

- Context sensitive menu features
- Display mode
- List control options
- Colors for different properties
- Font type, style, and size
- Database specifications (server name, user ID, and password)
- Query filters
- Column management
- Sorting

Run-time users can change options while the application is running to select the data they are viewing. They can:

- Sort the information within a column
- Update the display
- Perform a query
- Resize the width of a column

For more information about ActiveX controls, see ActiveX controls.

Configure the Alarm DB View control

When you configure the Alarm DB View control, you:

- Configure the connection to the alarm database.
- Configure the appearance of the grid.
- Select a display font.
- Configure the visual appearance.
- Set which features the user can access while the application is running.
- Configure which alarms to show.
- Create custom filters.
- Set the colors for the types of alarm records.
- Configure the time formats as shown.
- Configure the sort order of alarm records shown.

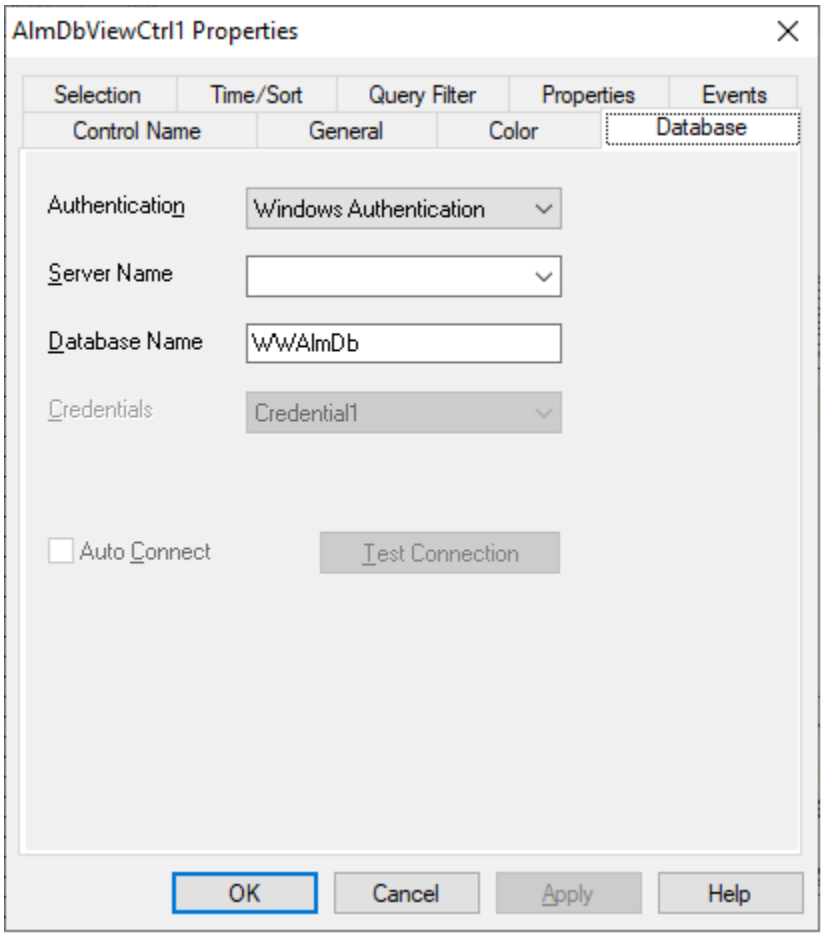
Connect the Alarm DB View to the alarm database

You must configure the connection between the Alarm DB View control and the alarm database.

Use an account with appropriate read-only access to the alarm database and not a system administrator account.

Configure the connection to the alarm database

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDbViewCtrl Properties** dialog box appears.
2. Select the **Database** tab.



3. Configure the connection to the alarm database. Do the following:
   a. In the **Authentication** list, select the authentication method: **SQL Server Authentication** or **Windows Authentication** (default).

      Note: Windows authentication can provide better application security than SQL Authentication. If you switch from Windows Authentication to SQL Authentication, a dialog will appear recommending that you use Windows Authentication for this reason. If you choose to ignore this warning and proceed with SQL Authentication, select **OK**. A similar message will be logged in the SMC Log Viewer.

   b. In the **Server Name** box, enter the node name of the computer where the alarm database is installed.
   c. In the **Database Name** box, type the database name.
   d. In the **Credentials** box, select the credentials for authentication.

Note:
- The Credentials drop-down is enabled only when the SQL Server Authentication type is selected. The Windows Authentication method uses the credentials of the user currently logged in, and disables the user name and password fields.
- For standalone InTouch applications, the credentials are retrieved from the Application Manager. For

managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credentials Manager in AVEVA™ InTouch HMI Application Development.

4. Select the **Auto Connect** checkbox to have the Alarm DB View control automatically connect to the alarm database when the InTouch application starts running.

   If you do not select the **Auto Connect** checkbox, you must configure the Alarm DB View control to connect to the alarm database by explicitly calling the Connect() method. For more information about the Connect method, see Connect().

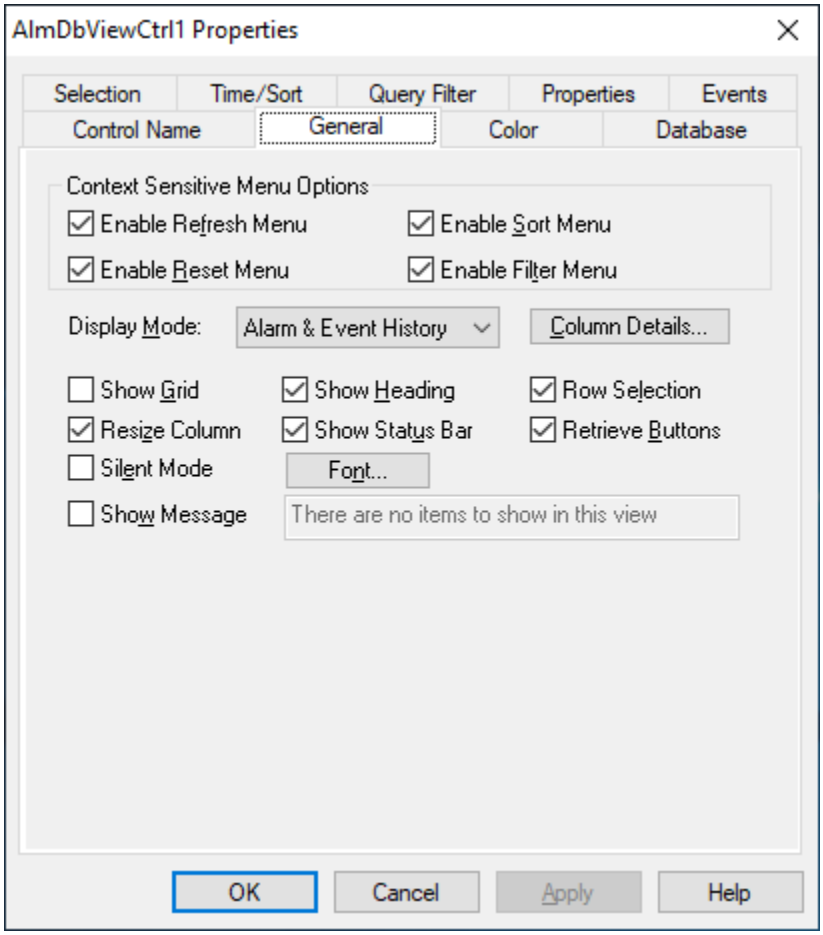
5. Select **Test Connection** to verify connectivity to the alarm database. A message indicates a successful connection.
6. Select **Apply**.

Configure the appearance properties of the grid

You can configure properties that determine the visual appearance of the Alarm DB View control.

Configure the appearance of the grid

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDbViewCtrl Properties** dialog box appears.
2. Select the **General** tab.



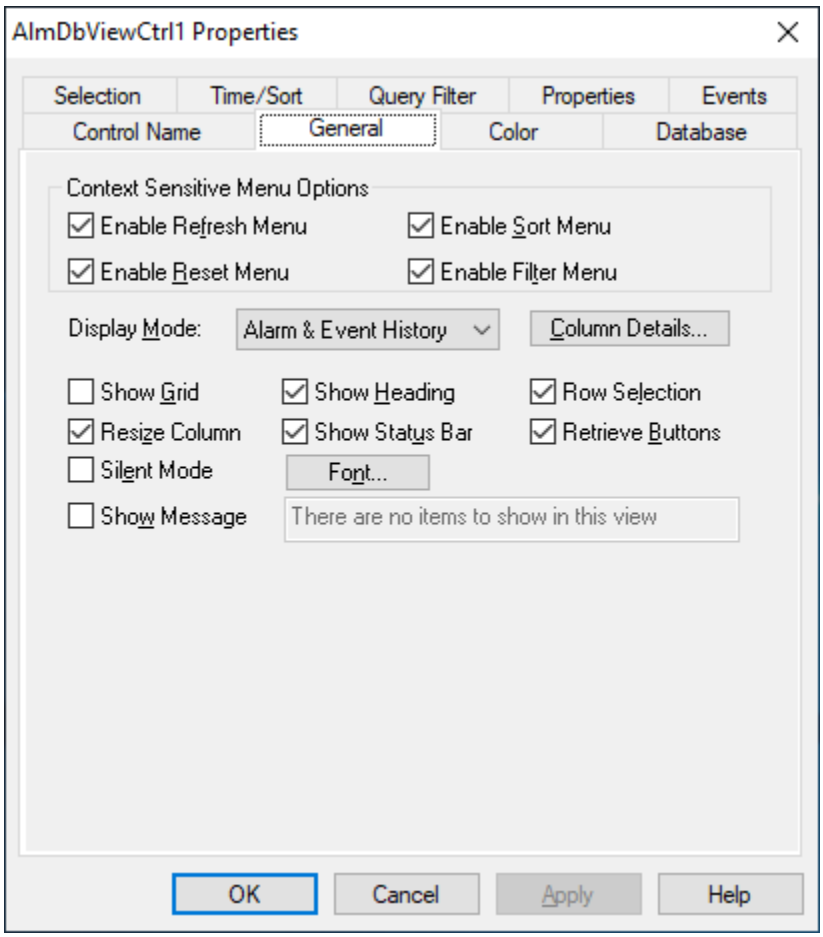
3. Configure the general appearance. Do the following:
   - Select the **Show Grid** checkbox to show the grid.
   - Select the **Silent Mode** checkbox to prevent the control from showing any error messages at run time.
   - Select the **Show Message** checkbox to show a message if the alarm query does not return any records. Type the message to show in the box.
   - Select the **Show Heading** checkbox to show the control heading.
   - Select the **Show Status Bar** checkbox to show the status bar.
4. Select **Apply**.

Configure the display font

You can select the font of all text that appears in the Alarm DB View control.

Configure the font properties

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDbViewCtrl Properties** dialog box appears.
2. Select the **General** tab.



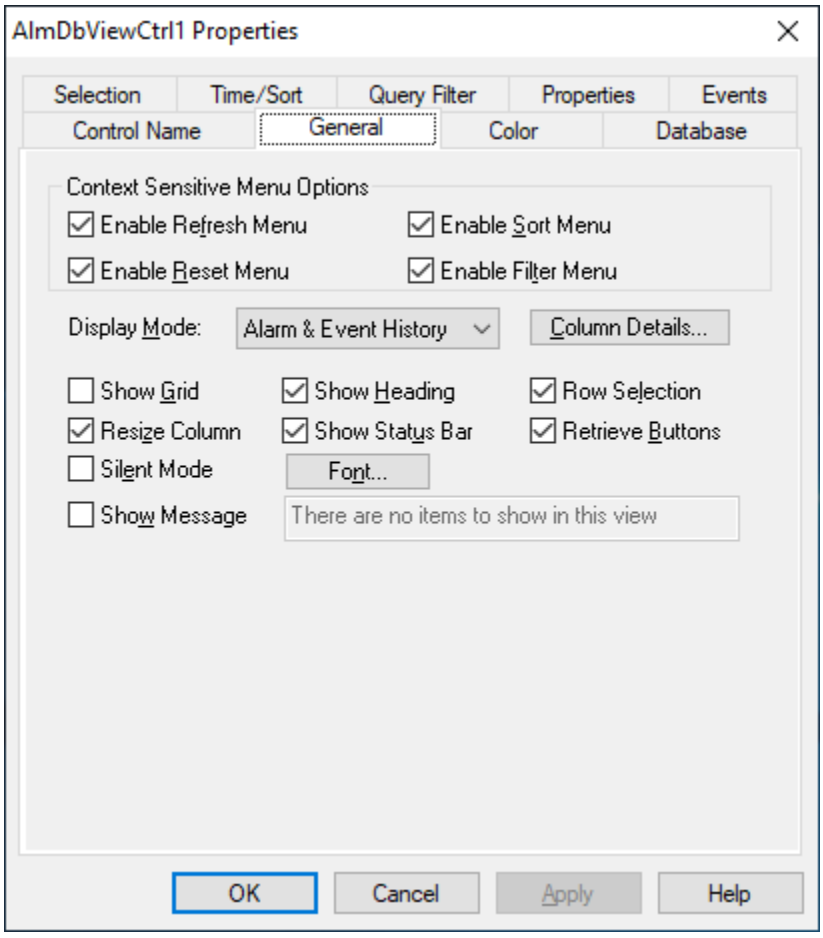
3. Select **Font**. The standard Windows Font dialog box appears. Configure the font and then select **OK**.
4. Select **Apply**.

Choose alarm or event data

You can configure if alarm records, event records, or both appear in an Alarm DB View control.

Select the type of data

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDbViewCtrl Properties** dialog box appears.
2. Select the **General** tab.



3. In the **Display Mode** list, select the type of historical records from the list:
   a. Select **Alarm & Event History** to show both alarm and event historical database records.
   b. Select **Alarm History** to show only historical alarm records.
   c. Select **Event History** to show only historical event records.
4. Select **Apply**.

Select and configure display columns

For the Alarm DB View control, you can:

- Select and order the columns.
- Set the width of a column in pixels.

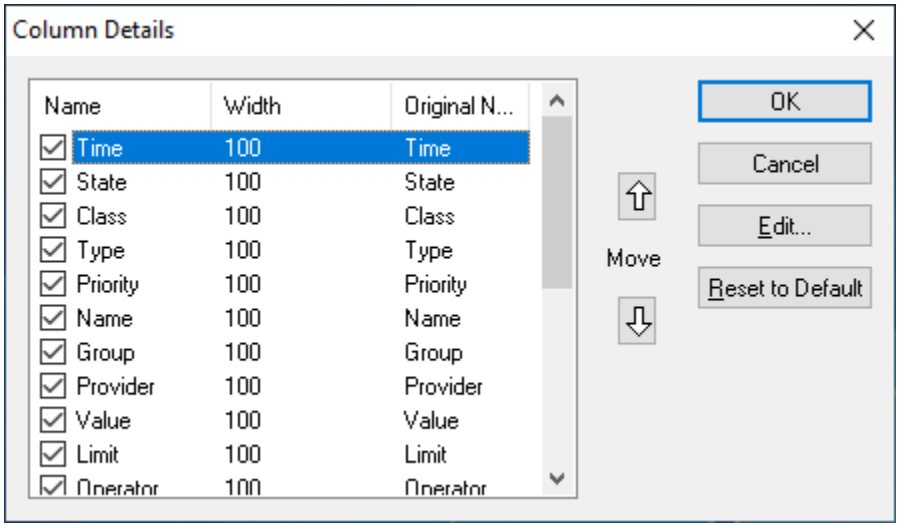
- Rename a column.

At least one column must be selected.

Important: The column names are reset to default column names if the display mode is changed. It is better to select the display mode first before changing the column names.

Configure the display column details

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDbViewCtrl Properties** dialog box appears.
2. Select the **General** tab.
3. Select **Column Details**. The **Column Details** dialog box appears.

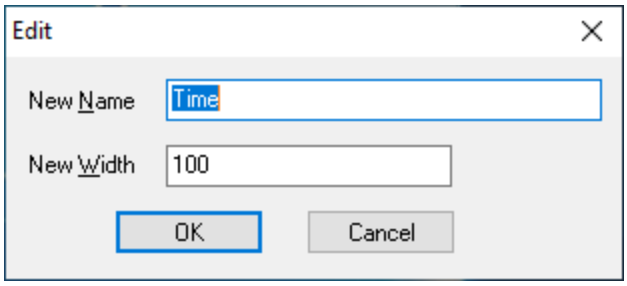


4. In the **Name** column, select the checkboxes next to the names of the columns that you want to appear in the Alarm DB View control. You must select at least one column from the list.

Column Name	Description
Time	Shows the time at which the alarm occurred.
State	Shows the state of the alarm.
Class	Shows the alarm category.
Type	Shows the alarm type.
Priority	Shows the alarm priority.
Name	Shows the tag or source that caused the alarm or event.
Group	Shows the alarm group name.
Provider	Shows the name of the alarm provider.

Column Name	Description
Value	Shows the value of the tag when the alarm occurred.
Limit	Shows the alarm limit value of the tag.
Operator	Shows the logged-on operator's ID associated with the alarm condition.
Operator Full Name	Shows the logged-on operator's full name.
Operator Node	Shows the logged on operator's node associated with the alarm condition. In a Terminal Services environment, this is the name of the client computer that the operator established the Terminal Services session from. If the node name can't be retrieved, the node's IP address is used instead.
Operator Domain	Shows the logged on operator's domain associated with the alarm condition.
Alarm Comment	Shows the tag's comments. This comment was typed in the **Alarm Comment** box when the tag's alarm was defined in the database. When an acknowledgement comment is introduced for alarms, the new comment is updated in this comment column.
User1	Shows the numerical values of **User Defined Number 1** corresponding to the alarm.
User2	Shows the numerical values of **User Defined Number 2** corresponding to the alarm.
User3	Shows the string value of the user defined string property associated with the alarm.
Duration	Shows the unacknowledgement duration or the alarm duration, depending on what has been selected by the operator.
UTC Time	Shows the time of the alarm in UTC.

5. Rearrange columns by selecting the column name and using the up and down arrows. The column name appearing at the top of the **Column Details** dialog box is the left-most column of the alarm control.
6. To change the name of a column or its width, select the column and select **Edit**. The Edit dialog box appears.



a. In the **New Name** box, type the new column name.
b. In the **New Width** box, type the column width. The column width can range from 1 to 999 pixels.
c. Select **OK**.

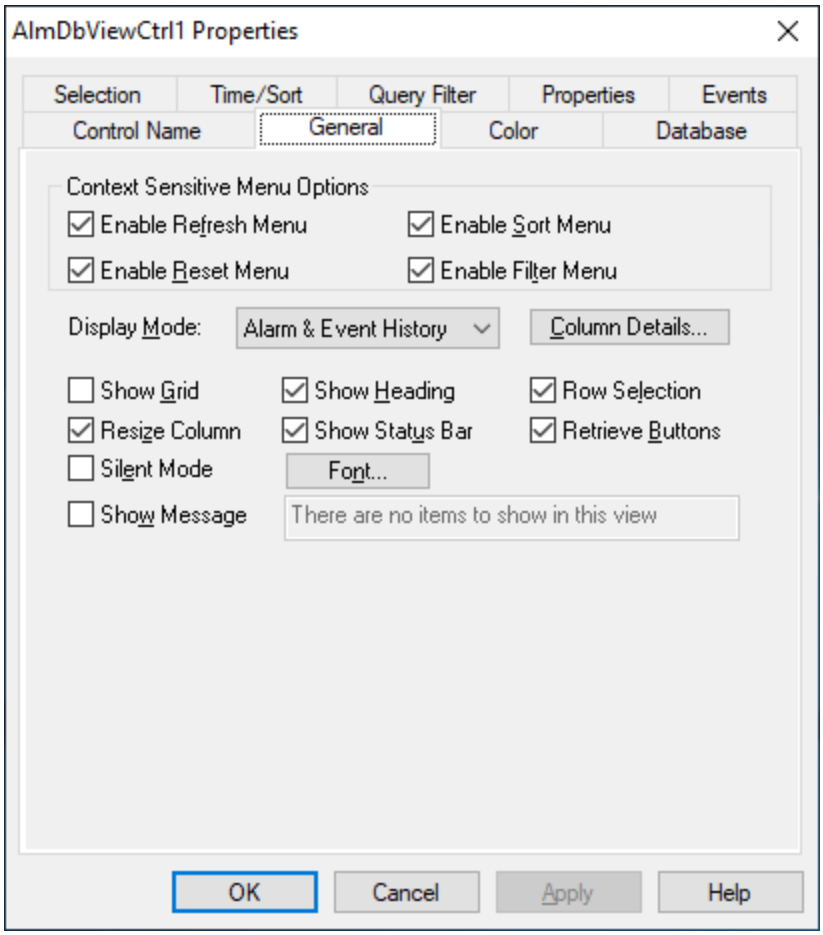
7. Select **Reset to Default** to return to the default column detail settings.
8. Select **OK** in the **Column Details** dialog box.
9. Select **Apply**.

Control which features you can access at run time

You can enable and disable the shortcut menu features available at this control.

Configure run-time features

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDbViewCtrl Properties** dialog box appears.
2. Select the **General** tab.



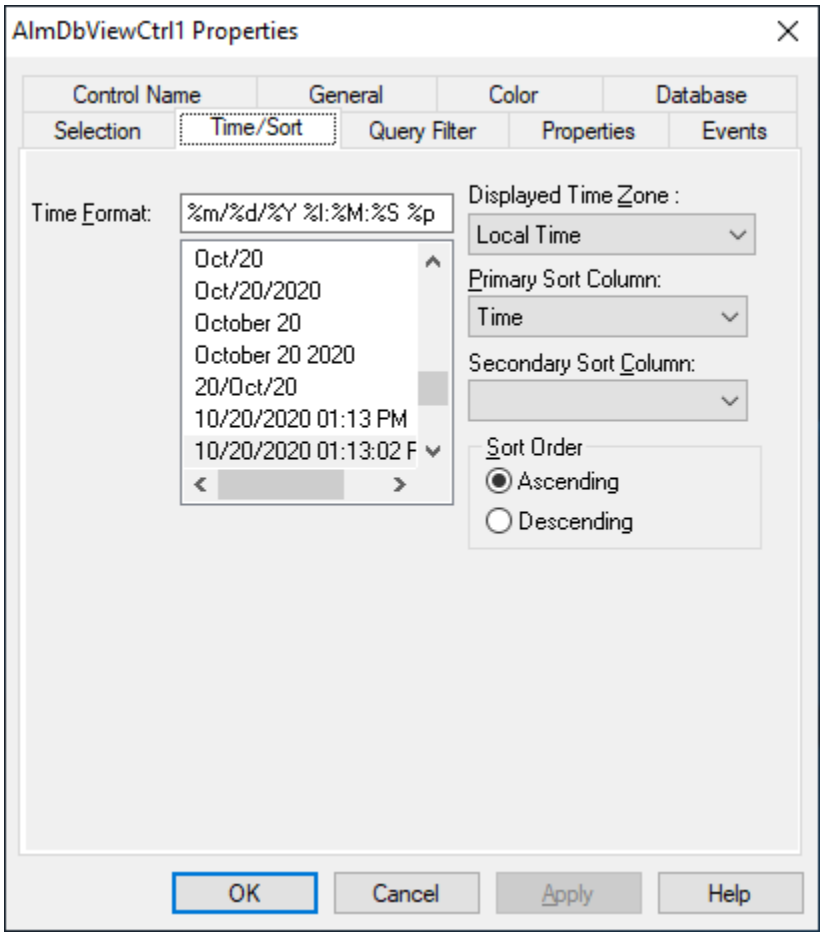
3. In the **Context-Sensitive Menu Options** area, configure which menu commands are available at run time.
   - Select the **Enable Refresh Menu** checkbox to enable the **Refresh** menu option in the shortcut menu of the control at run time. The **Refresh** menu refreshes the control to the database, and if the connection is successful, it shows the set of records in the range 1 to number defined by the MaxRecords property.
   - Select the **Enable Reset Menu** checkbox to enable the **Reset** menu option in the shortcut menu of the control at run time. The **Reset** menu arranges all the columns to the settings saved at design time.
   - Select the **Enable Sort Menu** checkbox to enable the Sort menu option on the shortcut menu of the control at run time. This menu shows the **Secondary Sort** menu used to set the user-defined sorting of columns.
   - Select the **Enable Filter Menu** checkbox to enable the **Filter** menu option on the shortcut menu of the control at run time. This menu shows the filter menu used to set the user-defined filtering criteria.
4. Select the **Resize Column** checkbox to allow resizing the columns.
5. Select the **Row Selection** checkbox to enable selecting alarm records.
6. Select the **Retrieve** buttons checkbox to show the retrieval buttons at the right side of the control.
7. Select **Apply**.

Configure the shown time format and time zone for alarm records

You can configure the time format and time zone for alarm records shown in the Alarm DB View control.

Configure time format

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDBViewCtrl Properties** dialog box appears.
2. Select the **Time/Sort** tab.



3. In the **Time Format** list, select a time format:

String character	Description	Example
d	Two-digit day	31
b	Three-letter month abbreviation	Aug
Y	Four-digit year	2007
m	Two-digit month	11
/	Date division function	06/2007
y	Two-digit year	07
#x	Full day and date	Friday, August 09, 2002

String character	Description	Example
B	Full month name	September
-	Dash date division punctuation	06-07
.	Period date division punctuation	06.07
,	Comma date division punctuation	Aug 09, 2007
H	24 hour time	22:15
:	Time division punctuation as in 4:41	
M	Minute	00:41
p	AM or PM display	
S	Seconds	16:41:07
s	Fractions of a second	16:41:07.390
I	12 hour time with AM/PM designation	04:41 PM

You can also manually enter your own format string in the list box using custom text and the formatting characters. Some sample time format character strings are shown below:

Time Format String	Display
%d %b	09 Aug
%m/%d/%Y	08/09/2002
%#x	Friday, August 09, 2002
%Y-%m-%d	2002-08-09
%m/%d/%Y %H:%M %p	08/09/2002 16:56 PM
%m/%d/%Y %H:%M:%s %p	08/09/2002 16:56:38.07
%I:%M %p	04:56 PM

4. In the **Displayed Time Zone** list, select the desired time zone:

Time Zone	Description
GMT	Alarm time stamps use Greenwich Mean Time.
Local Time	Alarm time stamps are shown with the local time of the client hosting the Alarm DB View control.
Origin Time	Alarm time stamps are shown with the local time of the alarm provider.

5. Select **Apply**.

Select the time period of data from the alarm database

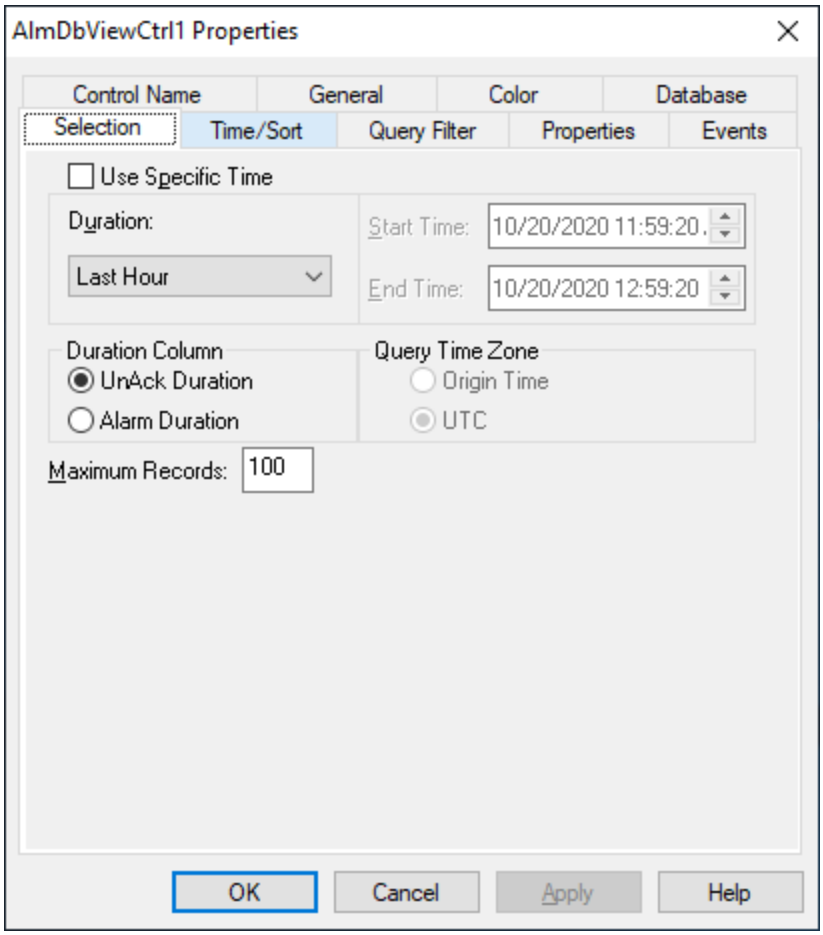
You can set query values to select records based on the selected time. You can also configure the maximum number of records to view, the start and end time of the alarm query, and the query time zone.

The number of records retrieved by an alarm database query affects the time required to show the records in the Alarm DB View control. The Alarm DB View control takes approximately 30 seconds to show 50000 records retrieved from the alarm database.

To improve the performance of the Alarm DB View control, reduce the **Maximum Records** value to display records more quickly. Also, you can select a shorter record retrieval interval in the **Duration** field to improve the performance of the Alarm DB View control.

Select the time period of data

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDBViewCtrl Properties** dialog box appears.
2. Select the **Selection** tab.



3. To use a pre-defined time interval that always queries for data using the UTC time zone, select an interval from the **Duration** list.
4. To use a specific start time and end time, select **Use Specific Time** and then configure the details.
   a. In the **Start Time** box, enter the start time to retrieve the alarm records. The string must be in MM/DD/YYYY HH:MM:SS format. Use any date in any time zone from midnight, January 1, 1970, to January 18, 19:14:07, 2038.
   b. In the **End Time** box, enter the end time to stop retrieving alarm records. The string must be in MM/DD/YYYY HH:MM:SS format. Use any date in any time zone from midnight, January 1, 1970, to January 18, 19:14:07, 2038.
   c. In the **Query Time Zone** area, select either **UTC** or **Origin Time**. UTC time is Greenwich Mean Time, also known as Coordinated Universal Time or Zulu. Origin time is the current time in the operator's time zone. For more information refer to Query Time Zones topic.
5. In the **Duration Column** area, configure which type of duration is shown. Duration is measured in milliseconds.
   - Select **UnAck Duration** to show the time between the most recent alarm transition (alarm or sub-state) and the acknowledgement, if any.
   - Select **Alarm Duration** to show the amount of time elapsed between the initial occurrence of an alarm and the time at which it returned to a normal state.

     For more information on the calculation of UnAck Duration and Alarm Duration for LATCHED state refer to Alarm Duration and UNACK Duration for LATCHED Alarms topic.

6. In the **Maximum Records** box, type the number of records to view from the control at one instance. The valid range of maximum records is from 1 to 1000.
7. Select **Apply**.

Query Time Zones

The Query Time Zone options are **Origin Time** and **UTC**.

- Origin time is the local time in the operator's time zone.
- UTC time is Greenwich Mean Time, also known as Coordinated Universal Time or Zulu.

If Use Specific Time is selected, you can select between the two query time zone options. If Use Specific Time is not selected, the selected Duration predefined time interval always queries for data using the UTC time zone.

To avoid problems during the Daylight Saving Time switch, we recommend that you always use UTC in Query Time Zone. If you use Local Time instead, it is possible that alarm records will be missing for the transition period from Daylight Saving Time to Normal Time.

If you are running several different computers with different time zone settings, and they are all logging to the same alarm database, each record will get the time stamps in UTC, plus the time zone offset and daylight saving adjustment needed to convert that time stamp to the corresponding Origin Time. As a result, every entry in the database has two time stamps: the UTC time and the Origin Time from the computer that did the logging. This makes retrievals faster. In the table entries, the UTC time is identified as the "Transition Time" and the Origin Time is identified as the "EventStamp."

Alarm Duration and UNACK Duration for LATCHED Alarms

This section describes the calculation of Alarm Duration and UNACK Duration for LATCHED state. Alarm Duration is the time duration between active alarm and inactive alarm. UNACK Duration is the time duration between UNACK Alarm and ACK Alarm. The Alarm Duration and UNACK Duration for the LATCHED state can be calculated in two ways:

Option 1: Alarm is acknowledged and then returned

Alarm State	Denominated As	Alarm Duration	UNACK Duration
UNACK	Tt	-	-
ACK	Ta	-	Tf-Tt
LATCHED	Tf	Tf-Tt	Tf-Tt
ACKRTN	Td	Tf-Tt	-

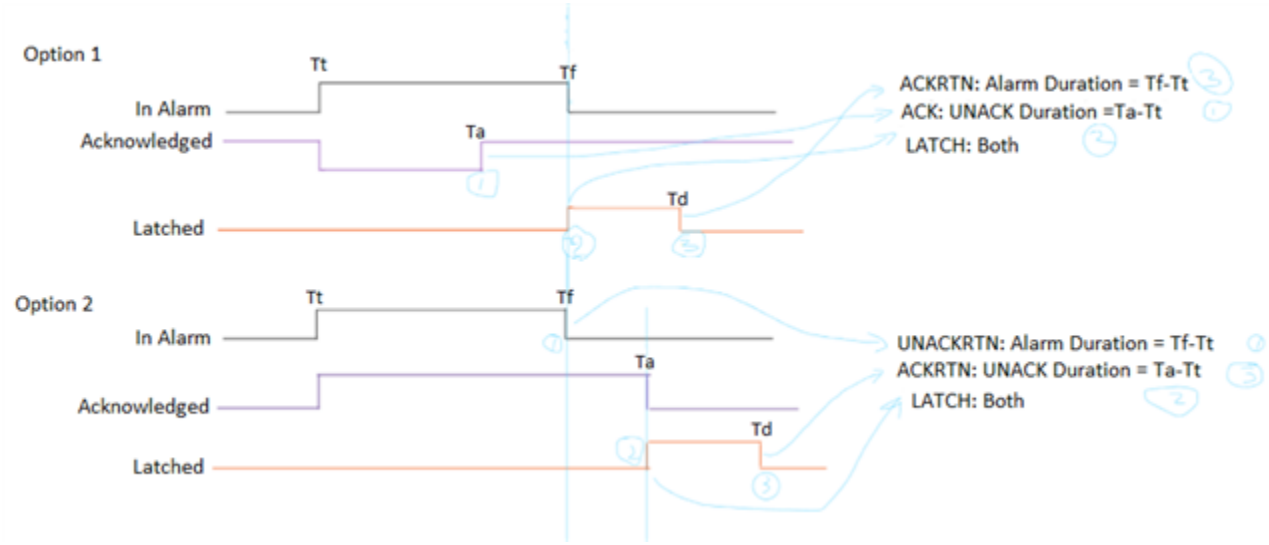
When the alarm is acknowledged, the UNACK Duration will be the time difference between ACK and LATCHED state, that is Ta-Tt. There will not be any Alarm Duration. When the alarm state is changed to LATCHED, then the Alarm Duration will be updated with the time difference of LATCHED and UNACK. that is Tf- Tt. The UNACK Duration remains same. When the LATCHED state is changed to ACKRTN, the Alarm Duration will be same, that is Tf-Tt and there will not be any UNACK Duration.

Option 2: Alarm is returned and then acknowledged

Alarm State	Denominated As	Alarm Duration	UNACK Duration
UNACK	Tt	-	-
UNACKRTN	Tf	Tf-Tt	-
LATCHED	Ta	Tf-Tt	Tf-Tt
ACKRTN	Td	-	Tf-Tt

When the alarm is returned, the Alarm Duration will be the time difference between UNACKRTN and UNACK, that is Tf-Tt. When the alarm state is changed to LATCHED, the alarm duration will be the same. The UNACK duration will be the time duration between LATCHED and UNACK, that is Ta-Tt. When the alarm changes to ACKRTN the UNACK Duration will be the same and there will not be any alarm duration.

The below image shows the timing diagram for both the options of Alarm Duration and UNACK Duration for LATCHED Alarms:



Create custom filters and use filter favorites

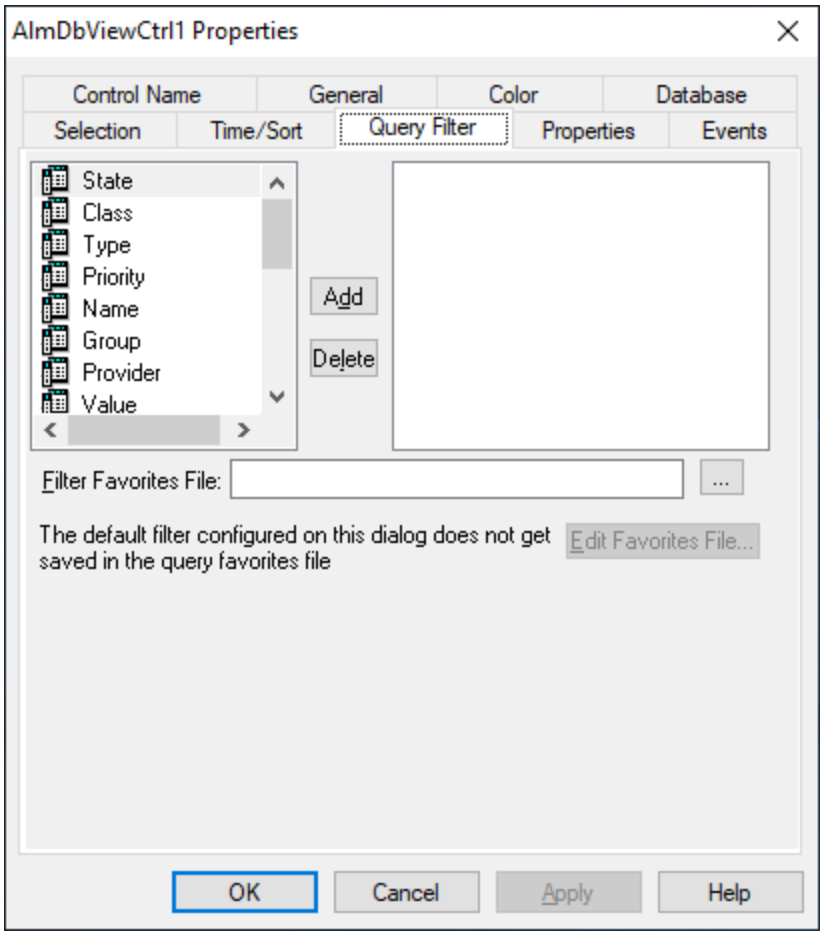
You can select which records are included in your query results. For example, you can select a filter by the date of a record or the state of the alarm. You can select multiple fields to limit or expand your query results.

If you do not define a custom filter, all records are returned.

If you translated strings for language switching, you cannot use those strings as filter criteria. Translated strings are stored outside of the alarm database.

Create custom filters

1. Right-click the **Alarm DB View** control and then select **Properties**. The AlmDBViewCtrl Properties dialog box appears.
2. Select the **Query Filter** tab.



3. In the left pane, select filter fields and then select **Add** to include them in the filter, which is shown in the right pane. The filter fields are described in the following table:

Field name	Filters query by:
State	Alarm state. For more information, see Values for the State column.
Class	Alarm class.
Type	Alarm type.
Priority	Alarm priority.
Name	Alarm name.
Group	Alarm group name.
Provider	Alarm provider.
Value	Alarm value. Values in the Value column are shown as alphanumeric values. The comparisons of these values in the Query Filter are done as string comparisons.

Field name	Filters query by:
Limit	Alarm limit. Values are alphanumeric. The comparisons of these values in the Query Filter are done as a string comparisons.
Operator	Operator.
OperatorFullName	Operator's full name.
OperatorNode	Operator's node name associated with the alarm.
OperatorDomain	Operator's domain name associated with the alarm.
Comment	Alarm comment.
User1	Alarm user-defined numeric value 1.
User2	Alarm user-defined numeric value 2.
User3	Alarm user-defined string value.
Duration	Unacknowledgement and alarm duration. A Duration column set equal to zero does not produce records with a null value in the query.

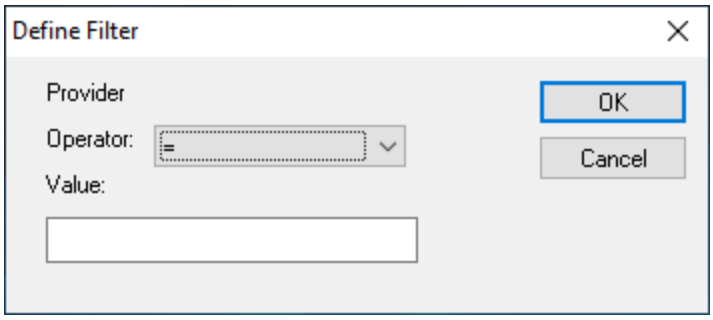
4. To remove a field from the filters pane, select the field you want to delete and select **Delete**. Deleting a filter cannot be undone. When a message appears, select **Yes**.
5. Configure the criteria for each filter field. For more information, see Define the column filter criteria.
6. Configure the operators and grouping for the filter. For more information, see Group alarm columns.
7. Configure the query favorites file. Do the following:
   a. In the **Filter Favorites File** box, type the network path and file name or select the ellipse button to browse for the file.
   b. To edit the **Filter Favorites** file, select the **Edit Favorites File** button. The **Filter Favorites** window opens, allowing you to add, modify, or delete filters from your favorites file. When you are done, select **OK** to save your changes and close the window.
8. Select **Apply**.

Define the column filter criteria

For each column filter you include in the query, you must configure the filter criteria. For example, you might want to only see alarms for a specific operator.

Define a column filter

1. Right-click the field and then select **Edit Filter**. The **Define Filter** dialog box appears.



2. In the **Operator** list, select the operator you need.
3. In the **Value** box, type the criteria that must be matched. The **Value** box does not accept values that cannot be processed for the selected query. The **Value** box accepts the following wildcard characters when the **Like** and **Not Like** filter operators are used for alphanumeric column names:

Character	Finds
%	Any string of zero or more characters
_	Any single character
[ ]	Any single character within the specified range, for example [a-f], or within a set, for example [abcdef].
[^]	Any single character not within the specified range, for example [^a-f], or set, for example [^abcdef].

The following Value box limits apply to different fields:

Field	Limit
All fields	All alphanumeric characters except User1, User2 and Priority.
Priority	Accepts integer values from 1 to 999.
User1, User2	Accepts only negative, positive or fractional numbers.

4. Select **OK**.

Group alarm columns

When more than one field is defined, the columns are combined using Boolean operators.

- The AND operator returns records that meet all values of the selected fields.
- The OR operator returns records that meet the values of any of the selected fields.

To use AND/OR operators to set the filter selection criteria, the respective fields must be grouped together. Only a single filter expression can be created on an item in the filters pane. If multiple expressions are needed, then

the item must be added to the filters pane again.

By default, the grouped fields have the AND operator.

The AND and OR operators are parent nodes. The fields selected under each parent node are child nodes. You cannot drag fields parent nodes to child nodes.

Group alarm columns

1. Right-click the field and then select **Group**.
2. Drag a field onto another field.

Copy or move query filters

If you have more than one instance of Alarm Pareto control and want to use the same filters for multiple instances, you can copy or cut the defined filters from one instance and paste them to another filter.

Copy filters

1. Define the filters in the first instance of the Alarm Pareto control.
2. Right-click the filters and select **Copy**. To move the filters, select **Cut**.
3. Close the first instance of the Alarm Pareto control.
4. Open the next instance of the Alarm Pareto control and select the **Query Filter** tab.
5. Position the arrow in the right pane. Right-click on a selected filter and select **Paste**.

Values for the State column

When you add the State column to your filter query, you may assign values from the **Value** menu in the **Define Filter** dialog box. The values available are described in the following table:

Value	Description
ACK	Produces a query of all system acknowledgements.
ACK_ALM	Produces a query of all acknowledged alarms.
UNACK_ALM	Produces a query of all unacknowledged alarms.
ACK_RTN	Produces a query of all acknowledged alarms that have returned to normal.
UNACK_RTN	Produces a query of unacknowledged alarms that have returned to normal.
All UNACK Records	Produces a query of all unacknowledged records.
All ACK Records	Produces a query of all acknowledgement records.

Value	Description
All ALM Records	Produces a query of all alarm records.
All RTN Records	Produces a query of all alarms that have returned to normal.

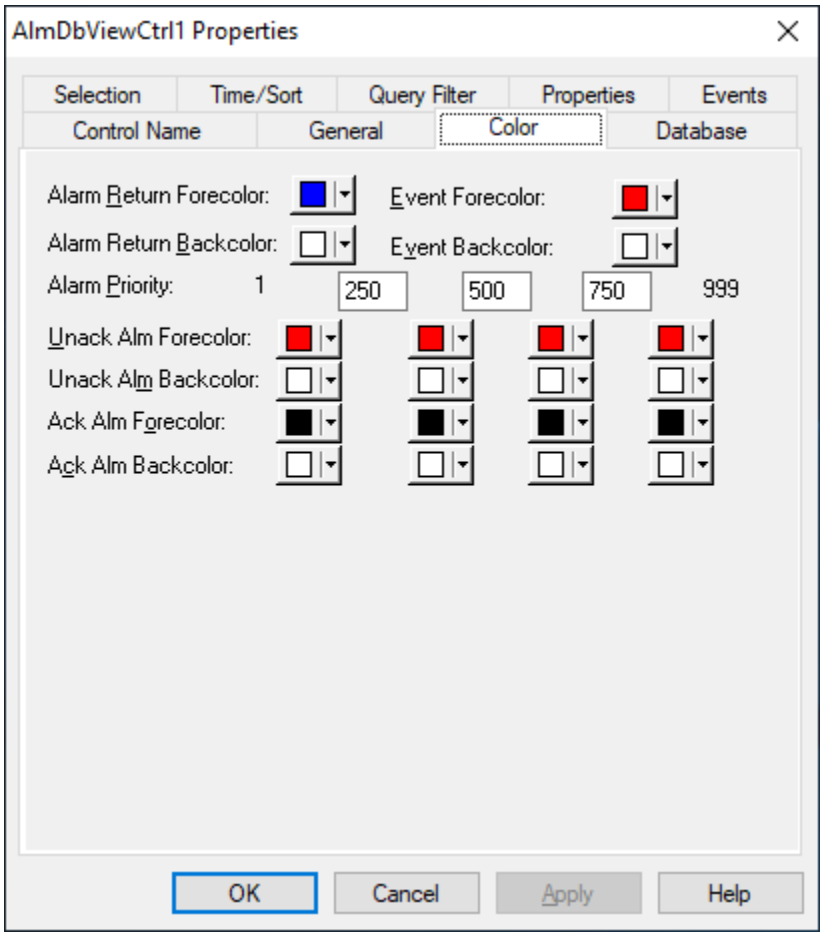
Note: When a tag in expanded summary alarm mode is used to create an alarm, which returns to normal when the main alarm is acknowledged, two records are created. The first record is the "acknowledged and returned to normal" record as the new alarm is already in a returned to normal state. The second record is acknowledged, which corresponds to acknowledging the main alarm. The previous implementation of ACK_ALM has been changed to ACK.

Configure colors for various types of alarm records

You can set the colors for alarm and event records.

Configure alarm display colors

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDBViewCtrl Properties** dialog box appears.
2. Select the **Color** tab.



3. Configure the color for each of the following by selecting the color box to open the palette.

Option	Description
Alarm Return Forecolor	Sets the foreground color of records for alarms that return to normal.
Alarm Return Backcolor	Sets the background color of records for alarms that return to normal.
Event Forecolor	Sets the foreground color of events records
Event Backcolor	Sets the background color of events records

4. In the Alarm Priority boxes, type the breakpoint values for the alarm display. You can assign breakpoint values so that alarms will appear in different colors depending on the Alarm Priority. The default minimum and maximum alarm priority values are 1 and 999, respectively.

   For example, you set the **Unack Alm Forecolor** to be orange for the priority range of 250 to 499 and red for the priority range of 1 to 249. If an alarm occurs with a priority of 254, it is shown in an orange font. If an alarm occurs with a priority of 12, it is shown in a red font.

5. Configure the color for each of the following by selecting the color box to open the palette.

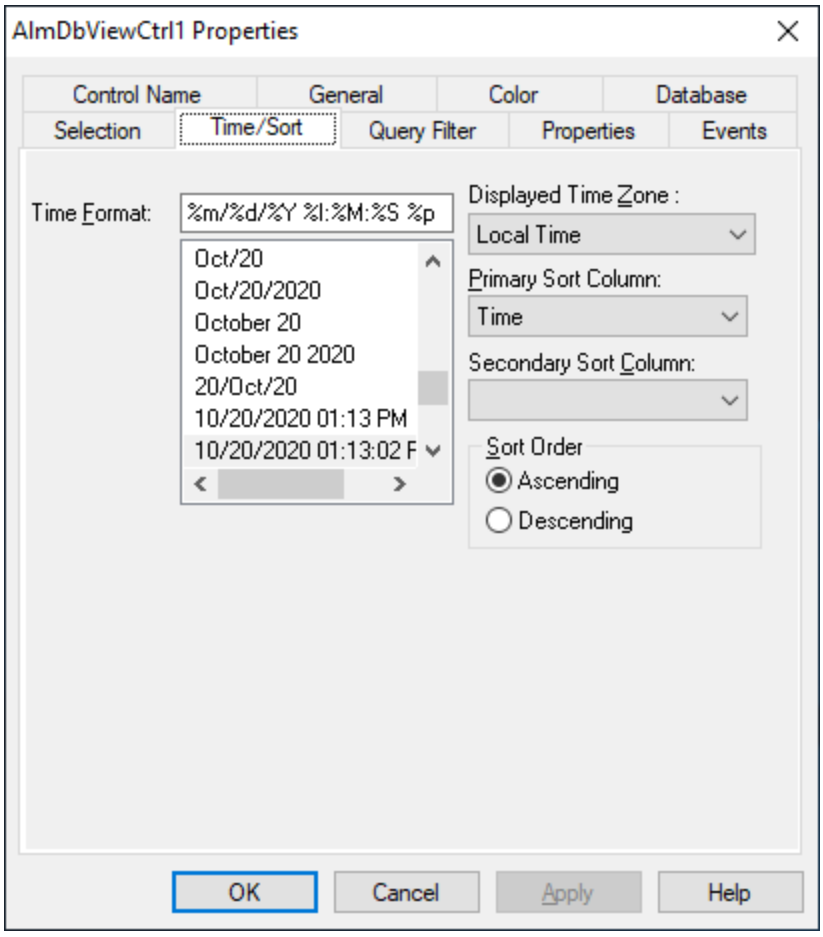
Option	Description
Unack Alm Forecolor	Sets the foreground color for each color priority range for unacknowledged alarms.
Unack Alm Backcolor	Sets the background color for each color priority range for unacknowledged alarms.
Ack Alm Forecolor	Sets the foreground color for each color priority range for acknowledged alarms.
Ack Alm Backcolor	Sets the background color for each color priority range for acknowledged alarms.

6. Select **Apply**.

Configure the sort order for alarm records

You can control the way the alarm records are sorted in the control.

1. Right-click the Alarm DB View control and then select **Properties**. The **AlmDBViewCtrl Properties** dialog box appears.
2. Select the **Time/Sort** tab.



3. In the **Primary Sort Column** list, select the name of the primary sort column. Only visible columns appear in the **Sort Column** list. If you do not see the column you want, choose the **General** tab and select the column from Column Details.
4. In the **Secondary Sort Column** list, select the name of the secondary sort column.
5. Select **Ascending** or **Descending** as the sort direction.
6. Select **OK**.

Use an Alarm DB View control at run time

Depending on how the control is configured, you can:

- Retrieve and refresh the data.
- Sort the data.
- Filter the data.
- Select rows.
- Change the column order by dragging them.
- Reset all the columns to the settings saved at design time.

Sort records

You can sort records in the display. Select the heading to sort all the rows.

Right-click in the control and select **Sort** to open the **Secondary Sort** dialog box, where you can do single and multiple column sorting, in ascending or descending order.

To specify the columns to be sorted, select the checkbox beside the column name. Use the **Sort Order** arrow keys to rearrange the columns.

If multiple alarm events have the same time stamp, they may not appear in the expected order.

For example, if the desired sorting is in descending order based on the alarm state first, then:

1. Select both the **Date** and **State** checkboxes.
2. Select the **State** row.
3. Select the **up sort** order arrow.
4. In the **Sort Type** area, select **Descending**.
5. Select **OK**.

Understand status bar information

The status bar shows the current status of the control.

- The left frame shows the server name and the database connected.
- The middle frame shows the number of the records that is shown out of the total number of records that is returned by the query.
- The right side of the frame shows the connection status with the server.

Use Alarm DB View ActiveX properties

You can set the value an Alarm DB View control property directly using a script or you can assign it to an InTouch tag or I/O reference. For more information about setting properties, see Scripting ActiveX Controls.

For more information on setting color values, see Configure colors for ActiveX Controls.

AckAlmBackColor Property

Gets or sets the acknowledged alarm background color. This setting overrides the individual range color settings for acknowledged alarms (AckAlmBackColorRange1 to AckAlmBackColorRange4).

Type

Integer

Default

White

Syntax

```
Object.AckAlmBackColor [= color]
```

Value

color

A value or constant that determines the color of the background.

AckAlmBackColorRange1 Property

Gets or sets the acknowledged alarm background color. This color applies to the records shown in the control with state ACK_ALM with priorities in the range 1 to ColorPriorityRange1.

Type

Integer

Default

White

Syntax

```
Object.AckAlmBackColorRange1 [= color]
```

Value

color

A value or constant that determines the color of the background.

AckAlmBackColorRange2 Property

Gets or sets the acknowledged alarm background color. This color applies to the records shown in the control with state ACK_ALM with priorities in the range ColorPriorityRange1 to ColorPriorityRange2.

Type

Integer

Default

White

Syntax

```
Object.AckAlmBackColorRange2 [= color]
```

Value

color

A value or constant that determines the color of the background.

AckAlmBackColorRange3 Property

Gets or sets the acknowledged alarm background color. This color applies to the records shown in the control with state ACK_ALM with priorities in the range ColorPriorityRange2 to ColorPriorityRange3.

Type

Integer

Default

White

Syntax

```
Object.AckAlmBackColorRange3 [= color]
```

Value

color

A value or constant that determines the color of the background.

AckAlmBackColorRange4 Property

Gets or sets the acknowledged alarm background color. This color applies to the records shown in the control with state ACK_ALM with priorities in the range ColorPriorityRange3 to 999.

Type

Integer

Default

White

Syntax

```
Object.AckAlmBackColorRange4 [= color]
```

Value

color

A value or constant that determines the color of the background.

AckAlmForeColor Property

Gets or sets the acknowledged alarm foreground color. This setting overrides the individual range color settings for acknowledged alarms (AckAlmForeColorRange1 to AckAlmForeColorRange4).

Type

Integer

Default

Black

Syntax

```
Object.AckAlmForeColor [= color]
```

Value

color

A value or constant that determines the color of the text.

AckAlmForeColorRange1 Property

Gets or sets the acknowledged alarm foreground color. The color applies to the records shown in the control with state ACK_ALM with priorities in the range 1 to ColorPriorityRange1.

Type

Integer

Default

Black

Syntax

```
Object.AckAlmForeColorRange1 [= color]
```

Value

color

A value or constant that determines the color of the text.

AckAlmForeColorRange2 Property

Gets or sets the acknowledged alarm foreground color. The color applies to the records shown in the control with state ACK_ALM with priorities in the range ColorPriorityRange1 to ColorPriorityRange2.

Type

Integer

Default

Black

Syntax

```
Object.AckAlmForeColorRange2 [= color]
```

Value

color

A value or constant that determines the color of the text.

AckAlmForeColorRange3 Property

Gets or sets the acknowledged alarm foreground color. The color applies to the records shown in the control with state ACK_ALM with priorities in the range ColorPriorityRange2 to ColorPriorityRange3.

Type

Integer

Default

Black

Syntax

```
Object.AckAlmForeColorRange3 [= color]
```

Value

color

A value or constant that determines the color of the text.

AckAlmForeColorRange4 Property

Gets or sets the acknowledged alarm foreground color. The color applies to the records shown in the control with state ACK_ALM with priorities in the range ColorPriorityRange3 to 999.

Type

Integer

Default

Black

Syntax

```
Object.AckAlmForeColorRange4 [= color]
```

Value

color

A value or constant that determines the color of the text.

AckRtnBackColor Property

Gets or sets the background color of acknowledged alarms that return to normal (ACK_RTN).

Type

Integer

Default

White

Syntax

```
Object.AckRtnBackColor [= color]
```

Value

color
A value or constant that determines the color of the background.

AckRtnForeColor Property

Gets or sets the text color of acknowledged alarms that return to normal (ACK_RTN).

Type

Integer

Default

Blue

Syntax

```
Object.AckRtnForeColor [= color]
```

Value

color
A value or constant that determines the color of the text.

AlmRtnBackColor Property

Gets or sets the returned alarm background color. This color applies to the records shown with state ALM_RTN.

Type

Integer

Default

White

Syntax

```
Object.AlmRtnBackColor [= color]
```

Value

color
A value or constant that determines the color of the background.

AlmRtnForeColor Property

Gets or sets the returned alarm foreground color. This color applies to the records shown with state ALM_RTN.

Type

Integer

Default

Blue

Syntax

```
Object.AlmRtnForeColor [= color]
```

Value

color
A value or constant that determines the color of the text.

Authentication Property

Returns the Authentication Type selected in the Database tab.

Type

Message

Syntax

```
AType = AlarmDBCtrl1.Authentication;
```

Value

AType

A message value; either 'SQL Authentication' or 'Windows Authentication'.

AuthenticationMode Property

Sets the Authentication Mode.

Type

Integer

Syntax

```
Object.AuthenticationMode = [Int]
```

Value

Int

A integer value of 0 or 1. 0 denotes SQL Authentication and 1 denotes Windows Authentication.

AutoConnect Property

Gets or sets a value that determines whether the control automatically connects to the database at run time.

Data Type

Integer

Default

False

Syntax

```
Object.AutoConnect [= Integer]
```

Value

Integer

An integer expression specifying whether the control connects to the database at run time.
True = Connects to the database.
False = (Default) Does not connect to the database.

Remarks

You must explicitly call the Connect() method to connect to the database.

ColorPriorityRange1 Property

Sets the boundary of the priority range in which alarms are to be shown. The value of this property must be greater than one and less than the value for ColorPriorityRange2.

Type

Integer

Default

250

Syntax

```
Object.ColorPriorityRange1 [= integer or priority]
```

ColorPriorityRange2 Property

Sets the boundary of the priority range in which alarms are to be shown. The value of this property must be greater than the value for ColorPriorityRange1 and less than the value for ColorPriorityRange3.

Type

Integer

Default

500

Syntax

```
Object.ColorPriorityRange2 [= integer or priority]
```

ColorPriorityRange3 Property

Sets the boundary of the priority range in which alarms are to be shown. The value of this property must be greater than the value of ColorPriorityRange2 and less than 999.

Type

Integer

Default

750

Syntax

```
Object.ColorPriorityRange3 [= integer or priority]
```

ColumnResize Property

Returns or sets a value that determines whether the columns can be resized.

Type

Discrete

Default

True

Syntax

```
Object.ColumnResize [= Discrete]
```

Value

Discrete

True = (Default) Columns can be resized at runtime.
False = Columns cannot be resized.

ConnectStatus Property

Returns the status of the connection. This property is read-only.

Data Type

Message

Syntax

```
Object.ConnectStatus
```

Values

Connected = The control is connected to the database.
Not Connected = The control is not connected to the database.
In Progress = The control is connecting to the database.

Example

The name of the control is AlmDbView1 and tagname is a message tag.

```
Tagname = #AlmDbView1.ConnectStatus;
```

CustomMessage Property

Gets or sets the message that the Alarm DB View control shows when no alarm records can be retrieved from the alarm database.

Type

Message

Default

There are no items to show in this view.

Syntax

```
Object.CustomMessage [= string]
```

DatabaseName Property

Specifies the database to connect to.

Type

Message

Syntax

```
Object.DatabaseName [= text]
```

DisplayMode Property

Returns the display mode of the control, which determines if just alarms, just events, or both alarms and events are shown. This property is read-only.

Type

String

Default

Alarms & Events History

Syntax

```
Object.DisplayMode
```

Remarks

Possible values are:

Alarms & Events History
Alarms History
Events History

Example

The name of the control is AlmDbView1 and tag is a message tag.

```
tag = #AlmDbView1.DisplayMode;
```

DisplayedTimeZone Property

Gets or sets the shown time zone.

Type

String

Default

Local Time

Syntax

```
Object.DisplayedTimeZone [= message]
```

Remarks

Possible values are:

GMT - Alarm time stamps use Greenwich Mean Time.

Local Time - Alarm time stamps are shown with the local time of the client hosting the Alarm DB View control.

Origin Time - Alarm time stamps are shown with the local time of the alarm provider.

Duration Property

Gets or sets the duration used to set the start and end time.

Type

Message

Default

"Last Hour"

Syntax

```
Object.Duration [= text]
```

Value

text

A string expression that contains the duration. This property must have one of the following strings:
Last Minute
Last 5 Minutes
Last 15 Minutes
Last Half Hour
Last Hour
Last 2 Hours
Last 4 Hours
Last 8 Hours
Last 12 Hours
Last Day
Last 2 Days
Last 3 Days
Last Week
Last 2 Weeks
Last 30 days
Last 90 days

EndTime Property

Returns or sets the end time.

Type

Message

Syntax

```
Object.EndTime [= text]
```

Value

text

A string expression that evaluates to the end time. The string returned is always in the format (MM/DD/YYYY HH:MM:SS). The same format is also required to set the value of the string. This property handles date in any time zone from midnight, January 1, 1970, to January 18, 19:14:07, 2038.

EventBackColor Property

Gets or sets the event alarm background color. This color applies to the records shown in the control with state EVT_EVT.

Type

Integer

Default

White

Syntax

```
Object.EventBackColor [= color]
```

Value

color

A value or constant that determines the color of the background.

EventForeColor Property

Gets or sets the event alarm foreground color. This color applies to the records shown in the control with state EVT_EVT.

Type

Integer

Default

Red

Syntax

```
Object.EventForeColor [= color]
```

Value

color

A value or constant that determines the color of the text.

FilterFavoritesFile Property

Gets or sets the filter favorites file. This file is used by the **Filter Favorites** dialog box to read or write filter favorites.

Type

String

Default

Null

Syntax

```
Object.FilterFavoritesFile [= String]
```

FilterMenu Property

Gets or sets a value that determines whether the **Filter** menu item is shown in the shortcut menu.

Type

Discrete

Default

True

Syntax

```
Object.FilterMenu [= Discrete]
```

Value

True = **Filter** menu item is shown (default).

False = **Filter** menu item is not shown.

FilterName Property

Returns the name of the current filter (if any).

Type

String (read-only)

Default

Null

Syntax

```
Object.FilterName [= String]
```

FromPriority Property

Gets or sets the From Priority value of the control.

Type

Integer

Default

1

Syntax

```
Object.FromPriority [= integer]
```

Remarks

You can use this property to filter which alarm records are shown. For example, if you set this property to 760, then only alarms with priority from 760 to the ToPriority property value are shown.

GroupExactMatch Property

When the GroupExactMatch property is true, only alarms with alarm group names that exactly match the GroupName property value are shown. When it is false, then the group name need only specify part of the alarm group names it is filtering for.

Type

Discrete

Default

False

Syntax

```
Object.GroupExactMatch [= discrete]
```

Remarks

Use this property together with the GroupName property to filter the Alarm DB View control.

Example

For example:

```
#AlarmDBViewCtrl1.GroupName = "Group"
#AlarmDBViewCtrl1.GroupExactMatch = 0;
#AlarmDBViewCtrl1.Refresh();
```

GroupName Property

Gets or sets a alarm group name filter for the current Alarm DB View control.

Type

String

Default

(none)

Syntax

```
Object.GroupName [= GroupName]
```

Remarks

Setting this property to "GroupA" and re-querying the display shows only tags belonging to the GroupA alarm group.

MaxRecords Property

Returns or sets the maximum records to be retrieved.

Type

Integer

Default

100

Syntax

```
Object.MaxRecords [=integer]
```

Value

integer

An integer expression specifying the number of records to be retrieved at a given time. The maximum records can be in the range from 1 to 1000. For best performance keep this value as small as needed.

Password Property

Returns or sets the SQL Server password for retrieving data.

Type

Message

Syntax

```
Object.Password [= text]
```

Value

text

A string expression that evaluates to the password.

PrimarySort Property

Gets or sets the primary column name used to sort the alarm display.

Type

Message

Default

(none)

Syntax

```
Object.PrimarySort [= message]
```

ProviderExactMatch Property

When the ProviderExactMatch property is true, only alarms with alarm provider names that exactly match the ProviderName property value are shown. When it is false, then the provider name need only specify part of the alarm provider names it is filtering for.

Type

Discrete

Default

False

Syntax

```
Object.ProviderExactMatch [= discrete]
```

Remarks

Use this property together with the ProviderName property to filter the Alarm DB View control.

Example

For example:

```
#AlarmDBViewCtrl1.ProviderName = "Provider"
#AlarmDBViewCtrl1.ProviderExactMatch = 0;
#AlarmDBViewCtrl1.Refresh();
```

ProviderName Property

Gets or sets a alarm provider name filter for the current Alarm DB View control.

Type

String

Default

(none)

Syntax

```
Object.ProviderName [= ProviderName]
```

Remarks

If a tag belongs to the Provider1 alarm provider, then setting this property to "Provider1" and re-querying the display shows only tags belonging to the Provider1 alarm provider.

QueryTimeZoneName Property

Gets or sets the time zone when a specific time is used for the query.

Type

Discrete

Default

False

Syntax

```
Object.QueryTimeZone [= Discrete]
```

Value

True = GMT

False = Origin time, which is the local time of the alarm provider.

RefreshMenu Property

Gets or sets a value that determines whether the **Refresh** menu item is shown in the shortcut menu.

Type

Discrete

Default

True

Syntax

```
Object.RefreshMenu [= Discrete]
```

Value

True = **Refresh** menu item is shown (default).

False = **Refresh** menu item is not shown.

ResetMenu Property

Gets or sets a value that determines whether the **Reset** menu item is shown in the shortcut menu.

Type

Discrete

Default

True

Syntax

```
Object.ResetMenu [= Discrete]
```

Value

True = **Reset** menu item is shown (default).

False = **Reset** menu item is not shown.

RowCount Property

Returns the number of records shown in the control. This property is read-only.

Type

Integer

Syntax

```
Object.RowCount
```

Example

The name of the control is AlmDbView1 and tagname an integer tag.

```
tagname = #AlmDbView1.RowCount;
```

RowSelection Property

Returns or sets a value that determines whether the row selection is allowed at run time.

Type

Discrete

Default

True

Syntax

```
Object.RowSelection [= Discrete]
```

Value

Discrete

True = (Default) Row selection is allowed.

False = Row Selection is not allowed.

Remarks

If row selection is not allowed, no Click or DoubleClick events are generated.

SecondarySort Property

Gets or sets the secondary column name used to sort the alarm display.

Type

Message

Default

(none)

Syntax

```
Object.SecondarySort [= text]
```

ServerName Property

Returns or sets the server name to which the control connects to retrieve data.

Type

Message

Syntax

```
Object.ServerName [= text]
```

ShowFetch Property

Returns or sets a value that determines whether the retrieval buttons are shown.

Type

Discrete

Default

True

Syntax

```
Object.ShowFetch [= Discrete]
```

Value

Discrete

True = (Default) Retrieve buttons are shown.
False = Retrieve buttons are not shown.

ShowGrid Property

Returns or sets a value that determines whether the grid lines are shown.

Type

Discrete

Default

False

Syntax

```
Object.ShowGrid [= Discrete]
```

Value

Discrete

True = Grid lines are shown.
False = (Default) Grid lines are not shown.

ShowHeading Property

Returns or sets a value that determines whether the column headings are shown.

Type

Discrete

Default

True

Syntax

```
Object.ShowHeading [= Discrete]
```

Value

Discrete

True = (Default) Column headings are shown.
False = Column headings are not shown.

ShowMessage Property

Determines if the customized message for "There are no items to show in this view" is shown when there are no records in the alarm database.

Type

Discrete

Default

False

Syntax

```
Object.ShowMessage [= discrete]
```

ShowStatusBar Property

Returns or sets a value that determines whether the status bar is shown.

Type

Discrete

Default

True

Syntax

```
Object.ShowStatusBar [= Discrete]
```

Value

Discrete

True = (Default) Status bar is shown.
False = Status bar is not shown.

SilentMode Property

Gets or sets a value that determines whether the control is in Silent mode.

Type

Discrete

Default

False

Syntax

```
Object.SilentMode [= Discrete]
```

Value

True = Silent mode is on.

False = Silent mode is off (default).

SortMenu Property

Returns or sets a value that determines whether the **Sort** menu item is shown in the shortcut menu.

Type

Discrete

Default

True

Syntax

```
Object.SortMenu [= Discrete]
```

Value

A discrete expression.

True = (Default) **Sort** menu item is shown

False = **Sort** menu item is not shown.

SortOrder Property

Gets or sets the sort order of the alarms according to the column to be sorted (the primary sort column).

Type

Discrete

Default

True

Syntax

```
Object.SortOrder [= discrete]
```

Value

An discrete expression.

True = Ascending order.

False = Descending order.

SpecificTime Property

Returns or sets a value that determines whether the control uses the StartTime and EndTime properties, or computes the start time and end time based on the value of the Duration property.

Type

Discrete

Default

False

Syntax

```
Object.SpecificTime [= Discrete]
```

Value

True = StartTime and EndTime properties are used.

False = (Default) StartTime and EndTime are computed based on the "Duration" property.

StartTime Property

Returns or sets the start time.

Type

Message

Syntax

```
Object.StartTime [= text]
```

Value

text

A string expression that evaluates to the Start Time. The string returned is always in the format (MM/DD/YYYY HH:MM:SS). The same format is also required to set the value of the string. This property handles date in any time zone from midnight, January 1, 1970, to January 18, 19:14:07, 2038

Time Property

Gets and sets the time format to be used in the display.

Type

Message

Default

%m/%d/%Y %I:%M:%S %p

Syntax

```
Object.Time [= message]
```

Remarks

For more information on the time format strings, see Configure the shown time format and time zone for alarm records.

ToPriority Property

Gets or sets the To Priority value of the control.

Type

Integer

Default

999

Syntax

```
Object.ToPriority [= integer]
```

Remarks

Use this property to filter which alarm records are shown. For example, if you set this property to 900, then only alarms with priority from the FromPriority property value to 900 are shown.

TotalRowCount Property

Returns the total number of records for the current query. This property is read-only.

Type

Integer

Syntax

```
Object.TotalRowCount
```

Remarks

The row count is the number of rows returned in the current query, which usually would be same as MaxRecords property except for the case when number of records retrieved are less than the MaxRecords property. For example, if there are 950 records for a specific criterion and the MaxRecords property is 100, then in the last page there would be 50 records and the row count would be 50. In the same example, the TotalRowCount property would always be 950.

Example

The name of the control is AlmDbView1 and tagname is an integer tag.

```
tagname = #AlmDbView1.TotalRowCount;
```

UnAckAlmBackColor Property

Gets or sets the unacknowledged alarm background color. This color applies to all records shown in the control

with state UNACK_ALM. It overrides any settings made by the UnAckAlmBackColorRange1 to UnAckAlmBackColorRange4 property values.

Type

Integer

Default

White

Syntax

```
Object.UnAckAlmBackColor [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckAlmBackColorRange1 Property

Gets or sets the unacknowledged alarm background color. This color applies to the records shown in the control with state UNACK_ALM with priorities in the range 1 to ColorPriorityRange1.

Type

Integer

Default

White

Syntax

```
Object.UnAckAlmBackColorRange1 [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckAlmBackColorRange2 Property

Gets or sets the unacknowledged alarm background color. This color applies to the records shown in the control with state UNACK_ALM with priorities in the range ColorPriorityRange1 to ColorPriorityRange2.

Type

Integer

Default

White

Syntax

```
Object.UnAckAlmBackColorRange2 [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckAlmBackColorRange3 Property

Gets or sets the unacknowledged alarm background color. This color applies to the records shown in the control with state UNACK_ALM with priorities in the range ColorPriorityRange2 to ColorPriorityRange3.

Type

Integer

Default

White

Syntax

```
Object.UnAckAlmBackColorRange3 [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckAlmBackColorRange4 Property

Gets or sets the unacknowledged alarm background color. This color applies to the records shown in the control with state UNACK_ALM with priorities in the range ColorPriorityRange3 to 999.

Type

Integer

Default

White

Syntax

```
Object.UnAckAlmBackColorRange4 [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckAlmForeColor Property

Gets or sets the unacknowledged alarm text color. This color applies to all records shown in the control with state UNACK_ALM. It overrides any settings made by the UnAckAlmForeColorRange1 to UnAckAlmForeColorRange4 property values.

Type

Integer

Default

Red

Syntax

```
Object.UnAckAlmBackColor [= color]
```

Value

color

A value or constant that determines the color of the text.

UnAckAlmForeColorRange1 Property

Gets or sets the unacknowledged alarm foreground color. This color applies to the records shown in the control with state UNACK_ALM with priorities in the range 1 to ColorPriorityRange1.

Type

Integer

Default

Red

Syntax

```
Object.UnAckAlmForeColorRange1 [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckAlmForeColorRange2 Property

Gets or sets the unacknowledged alarm foreground color. This color applies to the records shown in the control with state UNACK_ALM with priorities in the range ColorPriorityRange1 to ColorPriorityRange2.

Type

Integer

Default

Red

Syntax

```
Object.UnAckAlmForeColorRange2 [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckAlmForeColorRange3 Property

Gets or sets the unacknowledged alarm foreground color. This color applies to the records shown in the control with state UNACK_ALM with priorities in the range ColorPriorityRange2 to ColorPriorityRange3.

Type

Integer

Default

Red

Syntax

```
Object.UnAckAlmForeColorRange3 [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckAlmForeColorRange4 Property

Gets or sets the unacknowledged alarm foreground color. This color applies to the records shown in the control with state UNACK_ALM with priorities in the range ColorPriorityRange3 to 999.

Type

Integer

Default

Red

Syntax

```
Object.UnAckAlmForeColorRange4 [= color]
```

Value

color

A value or constant that determines the color of the specified object.

UnAckOrAlarmDuration Property

The duration column shows either UNACK Duration or Alarm Duration. FALSE (0) is UNACK Duration and TRUE (1) is Alarm Duration.

Type

Integer

Default

False

Syntax

```
Object.UnAckOrAlarmDuration [= integer]
```

UserID Property

Returns or sets the user ID used as the control to connect to the SQL Server to retrieve the data.

Type

Message

Syntax

```
Object.UserID [= text]
```

Value

text
A string expression that evaluates the user ID.

Use Alarm DB View ActiveX methods

Use the Alarm DB View ActiveX methods to:

- Control the database connection.
- Retrieve records from the alarm database.
- Retrieve information about an alarm.
- Reset the display appearance.
- Sort alarm records.
- Show the **Context** menu.

- Access filter favorites.

For more information about calling methods, see Scripting ActiveX Controls.

Control alarm database connections

Use the Connect() method to connect to the alarm database and the Disconnect() method to disconnect.

Connect()

Connects the control to the database and if the connection is successful, shows the set of records in the range 1 to MaxRecords.

Syntax

```
Object.Connect()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.Connect();
```

Disconnect()

Disconnects the control from the database.

Syntax

```
Object.Disconnect()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.Disconnect();
```

Retrieve alarm records from the database

Use the following methods to select a query, retrieve records from the database, and refresh the display:

- SelectQuery()
- GetPrevious()
- GetNext()
- Refresh()

SelectQuery()

Sets the current display to the query name specified in the .xml file.

Syntax

```
Object.SelectQuery(“QueryName”);
```

Parameters

QueryName

Name of a query defined in the filter favorites file.

Example

This example applies the filter criteria defined by the query called "HighPriority" in the filter favorites file that is currently associated with the AlmDbView1 control.

```
#AlmDbView1.SelectQuery("HighPriority");
```

GetPrevious()

Retrieves the previous set of records from the database (if any).

Syntax

```
Object.GetPrevious();
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.GetPrevious();
```

GetNext()

Retrieves the next set of records from the database (if any).

Syntax

```
Object.GetNext()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.GetNext();
```

Refresh()

Refreshes the control from the database, and if the connection is successful, displays the set of records in the range 1 to MaxRecords.

Syntax

```
Object.Refresh
```

Remarks

After initiating a refresh of the Alarm DB View control's display by calling its Refresh() method, the value of the RowCount and TotalRowCount properties change to -1 until the refresh is complete (that is, all relevant records are retrieved from the database). When the refresh is complete, both properties are updated with the correct, current row count.

The Refresh() method works asynchronously - it returns control to the calling script immediately and continues working in the background. This means that querying the value of RowCount and TotalRowCount immediately after calling Refresh() most likely returns -1, as their value is queried at a time when the refresh still hasn't completed. One way to get the correct values would be to use scripting to determine when the value of either property changes from -1 to a different value; this tells you that the correct values are now available.

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.Refresh();
```

Retrieve information about an alarm

Use the following methods to retrieve records from the database about a particular alarm:

- GetItem() method
- GetSelectedItem() Method

GetItem() method

Returns the data at a specified row and column as a string.

Syntax

```
Object.GetItem(Integer, Message)
```

Parameters

Integer

An integer expression that evaluates to a specific row in the control.

Message

A string expression that evaluates to the column name in the control.

Example

The name of the control is AlmDbView1 and tag is defined as a Message tag.

```
tag = #AlmDbView1.GetItem(1, "Group");
```

GetSelectedItem() Method

Returns the data for the selected row, given column as string

Syntax

```
Object.GetSelectedItem(Message)
```

Parameters

Message

An string expression that evaluates to the column name in the control

Example

The name of the control is AlmDbView1 and tag is defined as Message tag.

```
tag = #AlmDbView1.GetSelectedItem("State");
```

Sort the alarm records

Use the following methods to sort alarm records and reset column resizing:

- SortOnCol() Method
- ShowSort() Method
- Reset() Method

SortOnCol() Method

Performs primary sorting on alarm records that are shown.

Syntax

```
Object.SortOnCol(Message, Integer)
```

Parameters

Message

A string expression that evaluates to the column name in the control

Integer

Sort direction to be used. 0 = ascending, 1 = descending.

Example

The name of the control is AlmDbView1.

```
tag = #AlmDbView1.SortOnCol("Name",1);
```

ShowSort() Method

Shows the **Secondary Sort** dialog box if the SortMenu property is enabled.

Syntax

```
Object.ShowSort()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.ShowSort();
```

Reset() Method

Resets all the columns to the settings saved at design time.

Syntax

```
Object.Reset()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.Reset();
```

Show the context menu

Use the ShowContext() method to show the shortcut menu.

ShowContext() method

Shows the shortcut menu if any one of RefreshMenu or ResetMenu or SortMenu property is enabled.

Syntax

```
Object.ShowContext()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.ShowContext();
```

Access filter favorites

Use the ShowFilter() method to show the **Filter Favorites** dialog box.

ShowFilter() Method

Shows the **Filter Favorites** dialog box.

Syntax

```
Object.ShowFilter()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.ShowFilter();
```

Show other information

Use the AboutBox() method to show the **About** dialog box.

AboutBox() Method

Shows the **About** dialog box.

Syntax

```
Object.AboutBox()
```

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.AboutBox();
```

Handle errors with using methods and properties

Use the SilentMode property to determine whether the control is in silent mode or not. When the control is in silent mode, no error messages are shown. To see the error, call the GetLastError() method to return the error message.

GetLastError() Method

Returns the last error message if the Alarm DB View control is in silent mode.

Syntax

```
Object.GetLastError()
```

Example

The name of the control is AlmDbView1 and Tagname is defined as variant or string.

```
Tagname = #AlmDbView1.GetLastError();
```

Use Alarm DB View ActiveX events to trigger scripts

You can assign QuickScripts to Alarm DB View control events, such as a mouse click or double-click. When the event occurs, the QuickScript runs.

The Alarm DB View control supports the following events:

- Click
- DoubleClick
- ShutDown
- StartUp

The Click event has one parameter called ClicknRow, which identifies the row that is selected at run time.

The DoubleClick event has one parameter called DoubleClicknRow, which identifies the row that is double-clicked at run time.

Click and DoubleClick events are zero-based. When Click and/or DoubleClick events are published, the bar count in the display starts with 0.

Note: The Alarm DB View control ignores the user interface methods when they are called from StartUp event, because the control is not visible yet. These methods include: ShowSort(), ShowContext(), GetSelectedItem(), GetNext(), GetPrevious(), and AboutBox().

For more information about scripting ActiveX events, see Scripting ActiveX Controls.

Analyze alarm distribution across tags

Using the Alarm Pareto ActiveX control, you can analyze which alarms and events occur most frequently in a given production system. You can also analyze alarm frequency by the time periods during which they occur.

The analysis capabilities of the Alarm Pareto control identify the largest issues of your production systems. The Alarm Pareto control helps you recognize where you should focus your efforts to achieve the most significant improvements.

The Alarm Pareto control shows a bar chart representing alarm activity.

For more information about ActiveX controls, see ActiveX controls.

Configure an Alarm Pareto ActiveX control

When you configure an Alarm Pareto control, you specify:

- The connection to the alarm database.
- The appearance of the pareto control, including colors and fonts.
- Which features users can access at run time.
- Which alarms are shown in the chart and how they are presented.

Configure the connection to the alarm database

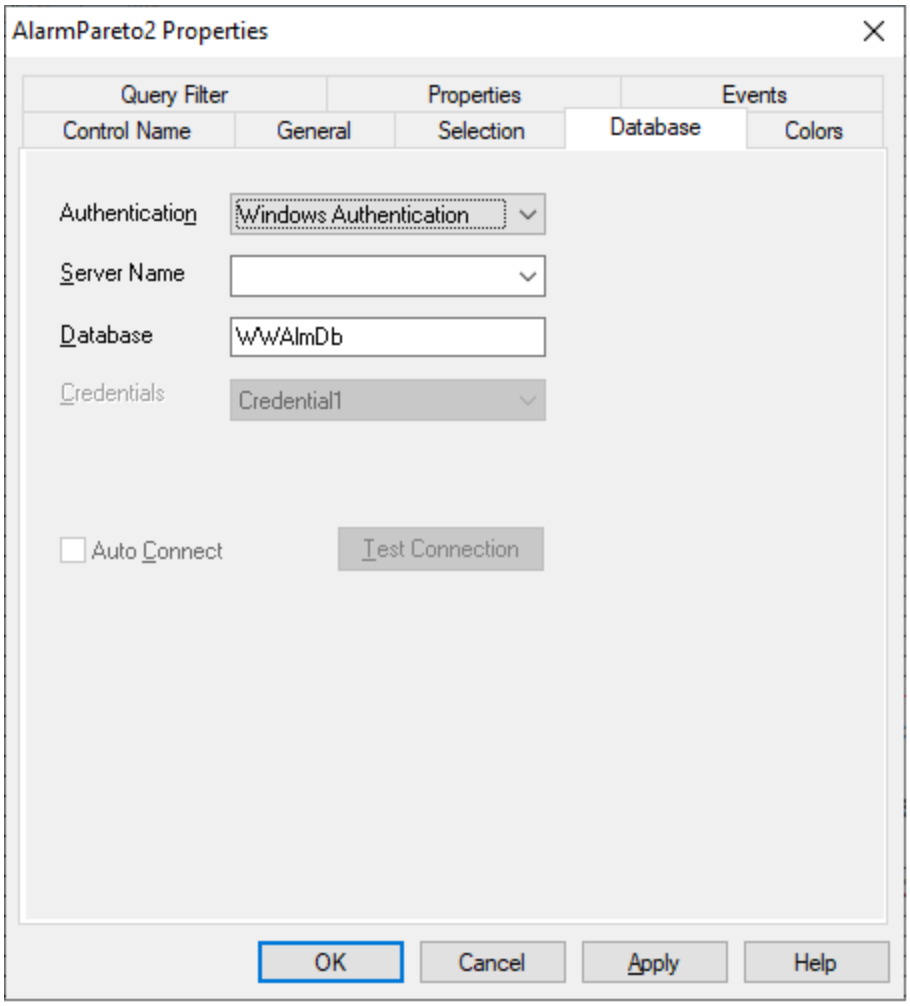
You must configure the connection between the Alarm Pareto control and the alarm database.

It is good practice to use an account with appropriate read-only access to the alarm database and not a system administrator account.

Configure the connection to the alarm database

1. Right-click the Alarm Pareto control and then select **Properties**.

   The **AlarmPareto Properties** dialog box appears.
2. Select the **Database** tab.



3. Configure the connection. Do the following:
   a. In the **Authentication** list, select the authentication method: **SQL Server Authentication** or **Windows Authentication** (default).

      Note: Windows authentication can provide better application security than SQL Authentication. If you switch from Windows Authentication to SQL Authentication, a dialog will appear recommending that you use Windows Authentication for this reason. If you choose to ignore this warning and proceed with SQL Authentication, select **OK**. A similar message will be logged in the OCMC Log Viewer.

   b. In the **Server Name** box, enter the node name of the computer where the alarm database is installed.
   c. In the **Database Name** box, type the name of the alarm database.
   d. From the **Credentials** drop down, select the credentials for authentication.

Note:
- The **Credentials** drop-down is enabled only when the SQL Server Authentication type is selected. The Windows Authentication method uses the credentials of the user currently logged in, and disables the user name and password fields.
- For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credential Manager in AVEVA™ InTouch HMI Application Development.

4. Select the **Auto Connect** checkbox if you want the Alarm Pareto control to automatically connect to the alarm database as soon as WindowViewer starts up.

   If you don't select the **Auto Connect** checkbox, you must configure the Alarm Pareto control to connect to the alarm database by explicitly calling the Connect() method. For more information about the Connect method, see Connect() Method.
5. Select **Test Connection** to verify connectivity to the alarm database. A message indicates a successful connection.
6. Select **Apply**.

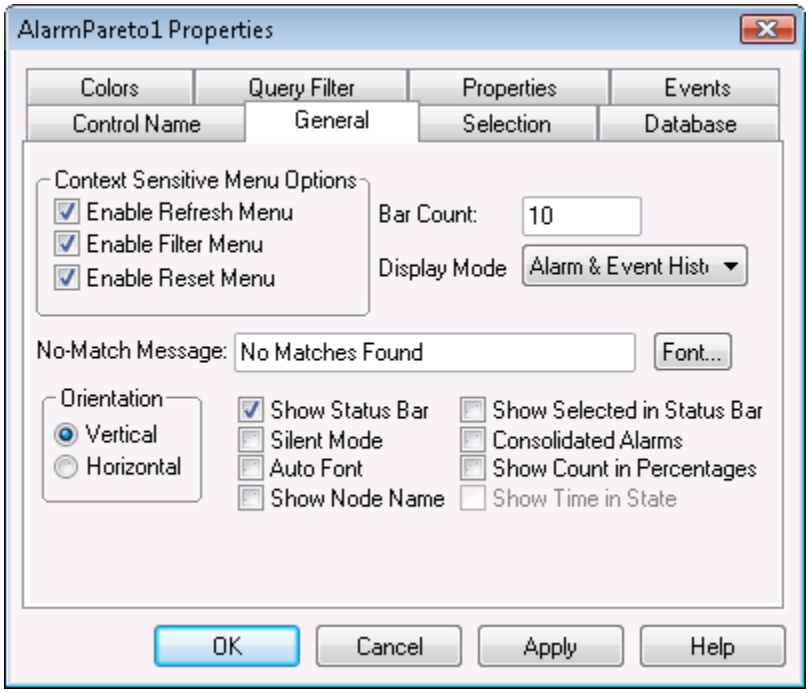
Configure the appearance and colors of the Alarm Pareto control

You can configure the visual appearance of the Alarm Pareto control. You can:

- Include a status bar.
- Set the orientation of the Pareto chart bars.
- Include descriptions of chart bars.
- Select the colors of the Alarm Pareto chart.

Set the appearance of the Alarm Pareto control

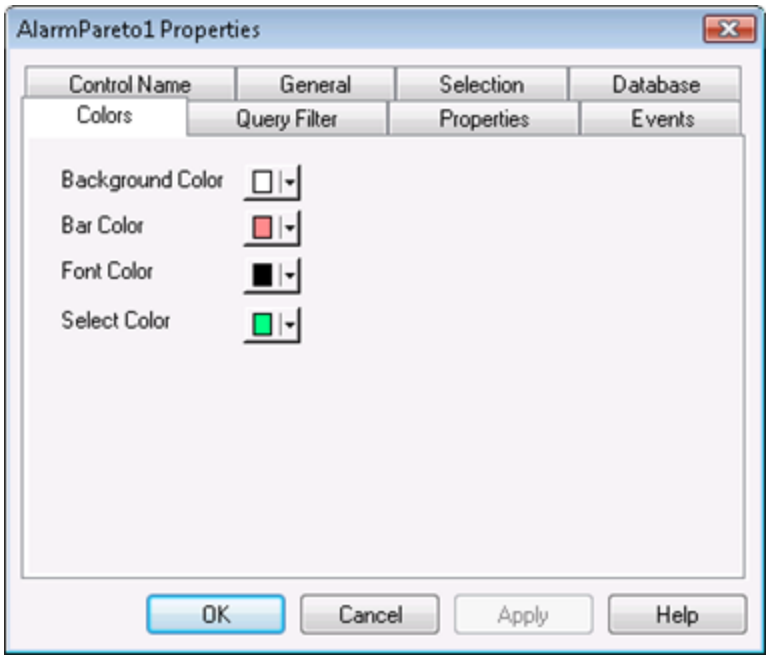
1. Right-click the Alarm Pareto control and then select **Properties**. The **AlarmPareto Properties** dialog box appears.
2. Select the **General** tab.



3. Configure the general options. Do the following:

Property	Description
Bar Count	Sets the number of bars to view in the Alarm Pareto control.
Display Mode	This list shows the available view options. The options are Alarm & Event History, Alarm History, and Event History.
No-Match Message	Sets the message to show when no data is processed from the Alarm Pareto control.
Vertical	Shows the bars on a vertical scale.
Horizontal	Shows the bars on a horizontal scale.
Show Status Bar	Enables the status bar.
Silent Mode	The Alarm Pareto control does not show run-time error messages. If it is not selected, the alarm display shows error messages. Error messages are always sent to the Logger.
Auto Font	When the space available is not enough to show the text on the selected bar correctly, Auto Font hides the text and only shows the text when the bar is selected.
Show Node Name	Shows the node name on the bar graph.
Show Selected in Status Bar	Shows the description of a selected bar on the status bar.
Consolidated Alarms	Consolidate the alarm states into two states. For example, if an analog tag has a three- state alarm: Hi, HiHi and Normal, the Hi and HiHi alarm states are classified as one state.
Show Count in Percentages	Shows the bars based on the percentage of count to the total count.
Show Time in State	Shows the Alarm Pareto control based on the time each tag is in an alarm state. This option is only enabled when the display mode is set to Alarm History.

4. Select **Apply**.
5. Select the **Colors** tab.



6. Select each color box to open a color palette.
7. Select the color that you want to assign for each of the following chart properties:

Property	Description
Background Color	Sets the background color of the Alarm Pareto chart
Bar Color	Sets the bar color of the chart
Font Color	Sets the color of text that appears in the chart
Select Color	Sets the color of a selected bar

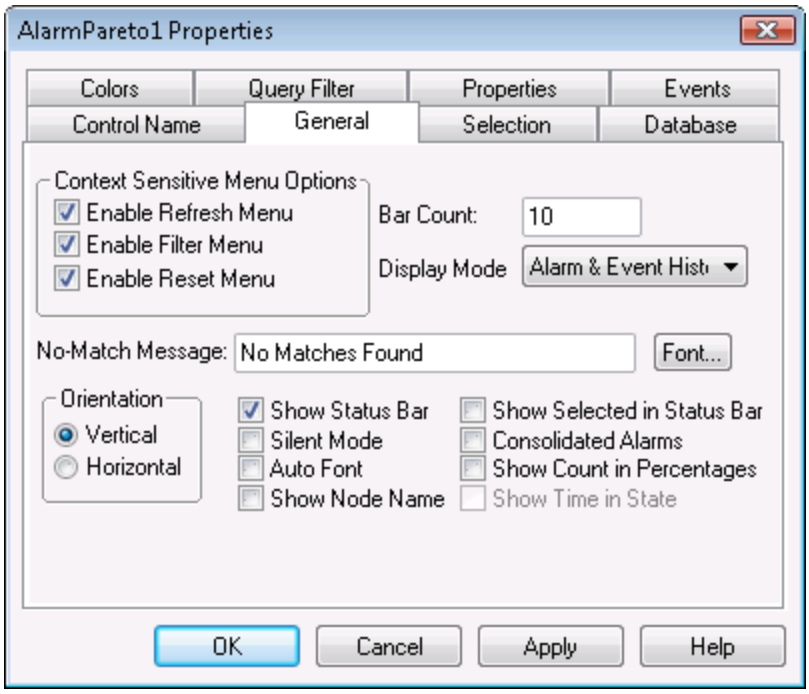
8. Select **Apply**.

Configure the display of the font

You can assign font properties to text that appears in your Alarm Pareto chart.

Configure the font properties

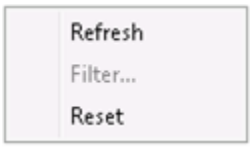
1. Right-click the Alarm Pareto control and then select **Properties**. The **AlarmPareto Properties** dialog box appears.
2. Select the **General** tab.



3. Select **Font**. The standard Windows **Font** dialog box appears. Configure the font and then select **OK**.
4. Select **OK**.

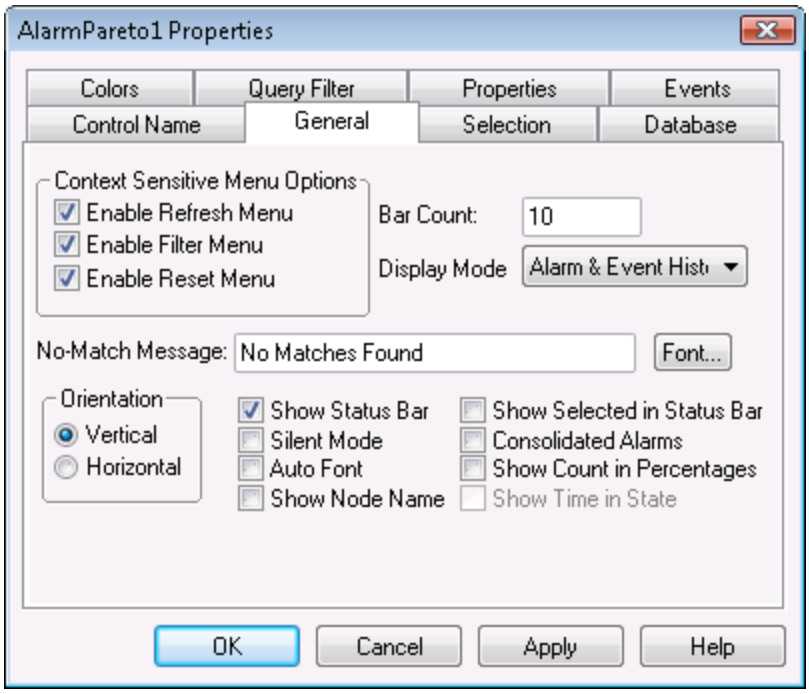
Configure which features users can access at run time

An Alarm Pareto chart includes a shortcut menu. When an operator right-clicks on the trend during run time, a menu appears with options to dynamically update the data shown in the trend.



Configure run time features

1. Right-click the Alarm Pareto control and then select **Properties**. The **AlarmPareto Properties** dialog box appears.
2. Select the **General** tab.



3. In the **Context Sensitive Menu Options** area, configure the menu commands:
   a. Select the **Enable Refresh Menu** checkbox to allow the run time user to refresh the data shown in the Alarm Pareto trend and show the records in the range from 1 to the number defined by the MaxRecords property.
   b. Select the **Enable Filter Menu** checkbox to allow the user to show the **Filter Favorites** dialog box to select a file containing database query values for the Alarm Pareto trend.
   c. Select the **Enable Reset Menu** checkbox to allow the user to restore the run-time Alarm Pareto chart to the original values specified from WindowMaker. All run-time changes made by an operator revert to the original design-time values.
4. Select **Apply**.

Configure which alarms to analyze

You configure which alarms to analyze using the Alarm Pareto chart. You specify:

- The type of database data (alarm or event data)
- A time period to select records from
- Criteria for filtering the data

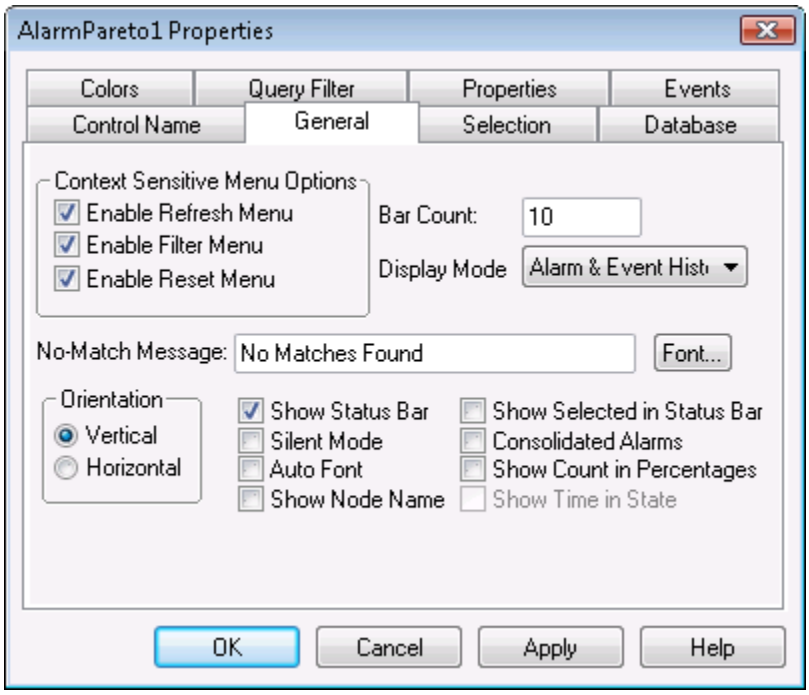
Select alarm or event data

You can configure if alarm records, event records, or both appear in an Alarm Pareto chart.

Select the type of data

1. Right-click the **Alarm Pareto** control and then select **Properties**. The AlarmPareto Properties dialog box appears.

2. Select the **General** tab.



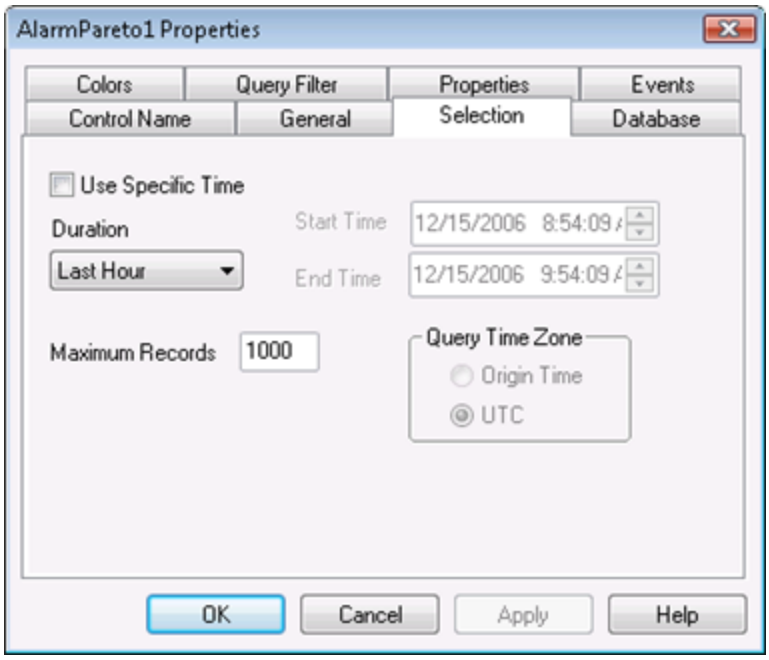
3. In the **Display Mode** list, configure the type of records. Do any of the following.
   - Select **Alarm & Event History** to show both alarm and event historical database records.
   - Select **Alarm History** to show only historical alarm records.
   - Select **Event History** to show only historical event records.
4. Select **Apply**.

Select the time period

You can set query values to select records based on the selected time. You can also configure the maximum number of records to view, the start and end time of the alarm query, and the query time zone.

Select the time period of data

1. Right-click the **Alarm Pareto** control and then select **Properties**. The AlarmPareto Properties dialog box appears.
2. Select the **Selection** tab.



3. To use a pre-defined time interval that always queries for data using UTC, select an interval from the **Duration** list.
4. To use a specific start time and end time, select **Use Specific Time** and then configure the details.
   a. In the **Start Time** box, enter the start time to retrieve the alarm records. The string must be in MM/DD/YYYY HH:MM:SS format. Use any date in any time zone from midnight, January 1, 1970, to January 18, 19:14:07, 2038.
   b. In the **End Time** box, enter the end time to stop retrieving alarm records. The string must be in MM/DD/YYYY HH:MM:SS format. Use any date in any time zone from midnight, January 1, 1970, to January 18, 19:14:07, 2038.
   c. In the **Query Time Zone** area, select either **UTC** or **Origin Time**. UTC time is Greenwich Mean Time, also known as Coordinated Universal Time or Zulu. Origin time is the current time in the operator's time zone.
5. In the **Maximum Records** box, type the number of records to view from the control at one instance. The valid range of maximum records is from 0 to 1000000.
6. Select **Apply**.

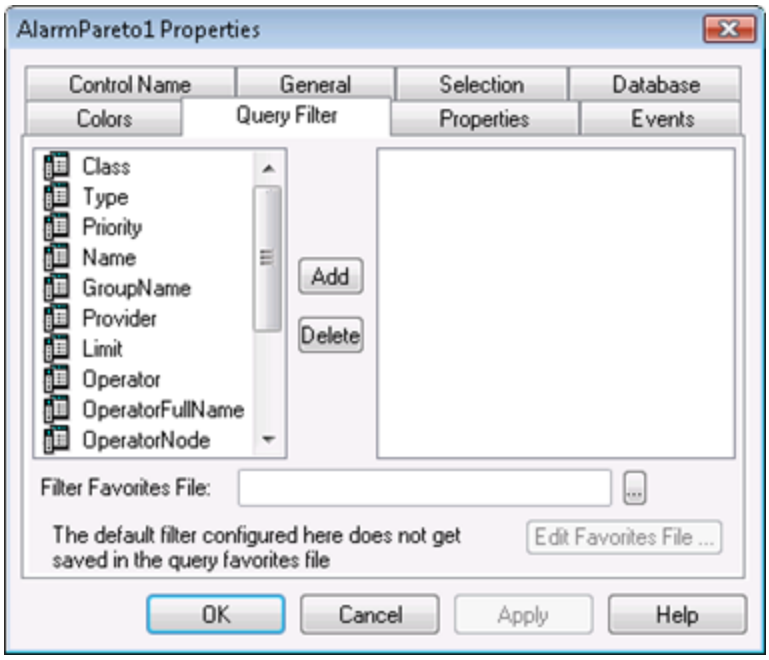
Create custom filters using filter favorites

You can select which records are included in your query results. For example, you can select a filter by the date of a record or the state of the alarm. You can select multiple fields to limit or expand your query results.

If you do not define a custom filter, then a default filter that queries all records is used.

Create custom filters

1. Right-click the **Alarm Pareto** control and then select **Properties**. The AlarmPareto Properties dialog box appears.
2. Select the **Query Filter** tab.



3. In the left pane, select filter fields and then select **Add** to include them in the filter, which is shown in the right pane. The filter fields are described in the following table:

Field name	Filters query by:
Class	Alarm class.
Type	Alarm type.
Priority	Alarm priority.
Name	Alarm name.
GroupName	Alarm group name.
Provider	Alarm provider.
Limit	Alarm limit. Values are alphanumeric. The comparisons of these values in the Query Filter are done as a string comparisons.
Operator	Operator.
OperatorFullName	Operator’s full name.
OperatorNode	Operator’s node name associated with the alarm.
OperatorDomain	Operator’s domain name associated with the alarm.
Comment	Alarm comment.

Field name	Filters query by:
User1	Alarm user-defined numeric value 1.
User2	Alarm user-defined numeric value 2.
User3	Alarm user-defined string value.
Duration	Unacknowledgement and alarm duration. A Duration column set equal to zero does not produce records with a null value in the query.

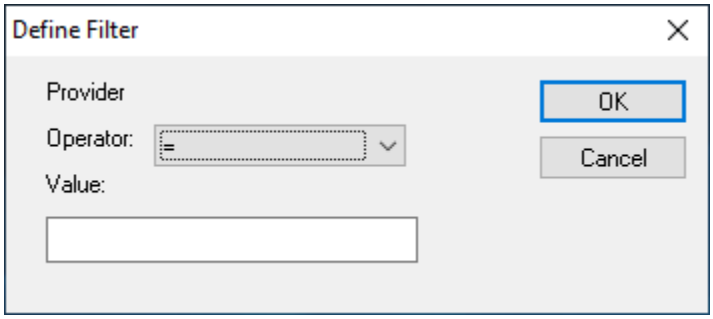
4. To remove a field from the filters pane, highlight the field you want to delete and select **Delete**. Deleting a filter cannot be undone. When a message appears, select **Yes**.
5. Configure the criteria for each filter field. For more information, see Define the column filter criteria.
6. Configure the operators and grouping for the filter. For more information, see Group alarm columns.
7. Configure the filter favorites file.
   a. In the Filter Favorites File box, type the network path and file name or select the ellipse button to browse for the file.
   b. To edit the **Filter Favorites** file, select the **Edit Favorites File** button. The **Filter Favorites** window opens, allowing you to add, modify, or delete filters from your favorites file. When you are done, select **OK** to save your changes and close the window.
8. Select **Apply**.

Define the column filter criteria

For each column filter you include in the query, you must configure the filter criteria. For example, you might want to only see alarms for a specific operator.

Define a column filter

1. Right-click the field and then select **Edit Filter**. The **Define Filter** dialog box appears.



2. In the **Operator** list, select the operator you need.
3. In the **Value** box, type the criteria that must be matched. The **Value** box does not accept values that cannot be processed for the selected query. The **Value** box accepts the following wildcard characters when the **Like** and **Not Like** filter operators are used for alphanumeric column names:

Character	Finds
%	Any string of zero or more characters
_	Any single character
[ ]	Any single character within the specified range, for example [a-f], or within a set, for example [abcdef].
[^]	Any single character not within the specified range, for example [^a-f], or set, for example [^abcdef].

The following Value box limits apply to different fields:

Field	Limit
All fields	All alphanumeric characters except User1, User2 and Priority.
Priority	Accepts integer values from 1 to 999.
User1, User2	Accepts only negative, positive or fractional numbers.

4. Select **OK**.

Group alarm columns

When more than one field is defined, the columns are combined using Boolean operators.

- The AND operator returns records that meet all values of the selected fields.
- The OR operator returns records that meet the values of any of the selected fields.

To use AND/OR operators to set the filter selection criteria, the respective fields must be grouped together. Only a single filter expression can be created on an item in the filters pane. If multiple expressions are needed, then the item must be added to the filters pane again.

By default, the grouped fields have the AND operator.

The AND and OR operators are parent nodes. The fields selected under each parent node are child nodes. You cannot drag fields parent nodes to child nodes.

Group alarm columns

1. Right-click the field and then select **Group**.
2. Drag a field onto another field.

Copy or move query filters

If you have more than one instance of Alarm Pareto control and want to use the same filters for multiple

instances, you can copy or cut the defined filters from one instance and paste them to another filter.

Copy filters

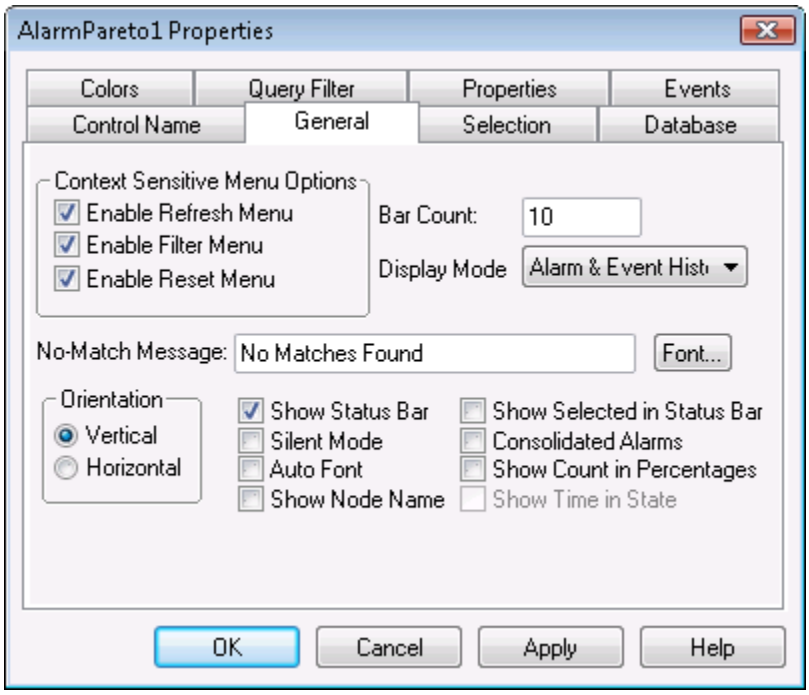
1. Define the filters in the first instance of the Alarm Pareto control.
2. Right-click the filters and select **Copy**. To move the filters, select **Cut**.
3. Close the first instance of the Alarm Pareto control.
4. Open the next instance of the Alarm Pareto control and select the **Query Filter** tab.
5. Position the arrow in the right pane. Right-click on a selected filter and select **Paste**.

Configure the presentation of the analysis results

You can configure the presentation of the alarms in the query results.

Configure the presentation

1. Right-click the **Alarm Pareto** control and then select **Properties**. The AlarmPareto Properties dialog box appears.
2. Select the **General** tab.



3. Select the **Consolidated Alarms** checkbox to combine alarms from a multi-state alarm into a single time stamped record.
4. Select the **Show Count in Percentages** checkbox to add a percentage of the total for each bar shown in the graph.
5. Select **Apply**.

Use an Alarm Pareto control at run time

Right-click on the Alarm Pareto control during run time to open a shortcut menu. The following table lists all possible options that appear in the shortcut menu.

Menu Option	Description
Refresh	Refreshes the display.
Filter	Allows you to edit the filter to change the data received by the Pareto control. This menu item is only available if a filter favorites file is set up.
Reset	Resets the graph to the default query.

Understand information shown on the status bar

The status bar for the Alarm Pareto control shows:

- The status of the database connection between the control and the alarm database.
- The update status of the graph to refresh the data shown in the graph.

Use Alarm Pareto ActiveX properties

You can set the value an Alarm Pareto control property directly using a script or you can assign it to an InTouch tag or I/O reference. For more information about setting properties, see Scripting ActiveX Controls.

The following table list the Alarm Pareto properties.

Property Name	Purpose
Authentication	Returns the Authentication Type selected in the Database tab.
AuthenticationMode	Sets a value that determines the Authentication Type (0 for SQL Authentication, 1 for Windows Authentication)
AutoConnect	Returns or sets a value that determines whether the control connects to the database as soon as the control is in run-time mode.
AutoFont	When the space available is not enough to show the text on the selected bar correctly, Auto Font hides the text and only shows the text on the selected bar when the bar is selected.
BackGndColor	Sets the background color of the Alarm Pareto chart

Property Name	Purpose
BarColor	Sets the bar color of the Alarm Pareto control.
BarCount	Sets the number of bars that appear on the Alarm Pareto control.
BarSelectColor	Sets the color of the selected bar in the Alarm Pareto Control.
Connected	Determines if the Alarm Pareto control is connected to a database.
ConsolidatedAlarms	Consolidates the alarm states into two states. For example, if an analog tag has a three state alarm: Hi, HiHi. and Normal, the Hi and HiHi alarm states are classified as one state.
DatabaseName	Sets the database name to which the Alarm Pareto control has to connect.
DisplayMode	Sets the display mode. The display mode options are Alarm & Event History, Alarm History or Event History.
Duration	Returns or sets the duration used by the control to set the "Start Time" and "End Time".
EnableRefresh	Enables or disables the context menu to refresh the Alarm Pareto control.
EnableReset	Enables or disables the context menu used to reset the Alarm Pareto control.
EnableSilentMode	Enables or disables the Silent Mode. If Silent Mode is disabled, the Alarm Pareto control shows error message boxes. If Silent Mode is enabled, the error message boxes do not appear. Error information is written to the logger.
EndTime	Returns or sets the end date and time.
FilterMenu	Enables or disables the context menu where you can edit the filter to change the data received by the Pareto control. This property is only enabled if Filter Menu Files is set.
FilterFavoritesFile	Specifies the filter favorites file, as a string.
Font	Sets the font for the records and the heading in the control.

Property Name	Purpose
FontColor	Sets the font color for the view of records in the Alarm Pareto control.
HorizontalChart	Shows the chart as horizontal bars. If HorizontalChart is disabled, the chart shows vertical bars.
MaxRecords	Returns or sets a value that specifies the maximum records to be retrieved at a given time.
NoMatchMessage	Sets the message that appears when there is no data to be processed for the Alarm Pareto control.
QueryTimeZone	Sets the time zone to either UTC Time or Origin Time.
ServerName	Returns the current server name.
ShowCountPercentage	If selected, shows the count for each bar as a percentage of the total count. If not selected, shows the actual count for each bar.
ShowNodeName	Sets the Alarm Pareto control to show the node name in addition to the other information on the bar.
ShowSelectedInStatusBar	Enables or disables to show the information from the selected bar on the status bar.
ShowStatusBar	Returns or sets a value that determines whether the status bar is shown.
ShowTimeinState	Returns or sets a value that determines whether the Alarm Pareto control shows the bars based on the time in which the tag has remained in alarm state. If disabled, the control shows the bars based on the number of times an alarm has occurred.
SpecificTime	Returns or sets a value that determines whether the control uses the "start time" and "end time" properties, or computes the start time and end time based on the value of the Duration property.
StartTime	Returns or sets the start date and time.
User	Returns or sets the User used as the control to connect to the SQL Server.

Use Alarm Pareto ActiveX methods

Use the Alarm Pareto ActiveX methods to:

- Control the database connection.
- Retrieve records from the database.
- Retrieving information about specific pareto bars.

For more information about calling methods, see Scripting ActiveX Controls.

Control database connections

Use the Connect() method to connect to the alarm database.

Connect() Method

Connects to the database configured from the **Database** tab of the Alarm Pareto control properties.

Syntax

```
Object.Connect()
```

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.Connect();
```

Retrieve records from the database

Use the following functions to retrieve records from the database:

- Refresh() Method
- SelectQuery() Method

Refresh() Method

Refreshes the control from the database, and if the connection is successful, shows the set of records in the range from 1 to the number defined by the MaxRecords property.

Syntax

```
Object.Refresh()
```

Example

```
#AlarmPareto1.Refresh();
```

SelectQuery() Method

Selects a filter that is configured as a query favorite file.

Syntax

```
Object.SelectQuery(Filter)
```

Parameter

Filter

The name of the query filter.

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.SelectQuery("MyFilter");
```

Retrieve information about specific pareto bars

Use the following functions to retrieve information about specific pareto bars:

- GetItemAlarmName() Method
- GetItemAlarmType() Method
- GetItemCount() Method
- GetItemTotalTime() Method
- GetItemEventType() Method
- GetItemProviderName() Method

GetItemAlarmName() Method

Gets the name of the alarm for a specific bar.

Syntax

```
Object.GetItemAlarmName(BarIndex)
```

Parameter

BarIndex

The index of the bar.

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.GetItemAlarmName(1);
```

GetItemAlarmType() Method

Gets the type of alarm for a specific bar.

Syntax

```
Object.GetItemAlarmType(BarIndex)
```

Parameter

BarIndex

The index of the bar.

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.GetItemAlarmType(1);
```

GetItemCount() Method

Gets the number of alarms in a bar.

Syntax

```
Object.GetItemCount(BarIndex)
```

Parameter

BarIndex

The index of the bar.

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.GetItemCount(1);
```

GetItemTotalTime() Method

Gets the total time, in seconds, of a tag in an alarm state.

Syntax

```
Object.GetItemTotalTime(BarIndex)
```

Parameter

BarIndex
The index of the bar.

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.GetItemTotalTime(1);
```

GetItemEventType() Method

Gets the type of event for a specific bar.

Syntax

```
Object.GetItemEventType(BarIndex)
```

Parameter

BarIndex
The index of the bar.

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.GetItemEventType(1);
```

GetItemProviderName() Method

Gets the name of the provider from the generated alarms for a specific bar.

Syntax

```
Object.GetItemProviderName(BarIndex)
```

Parameter

BarIndex

The index of the bar.

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.GetItemProviderName(1);
```

Show miscellaneous information

Use the AboutBox() method to show the **About** dialog box.

AboutBox() Method

Shows the **About** dialog box.

Syntax

```
Object.AboutBox()
```

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.AboutBox();
```

Error handling when using methods and properties

The Alarm Pareto control handles run-time error messages based upon the **Silent Mode** option. For more information, see Configure the appearance and colors of the Alarm Pareto control.

If **Silent Mode** is selected, the Alarm Pareto control does not show run-time error messages. If it is not selected, the alarm display shows error messages. All Alarm Pareto error messages are sent to the Logger.

Use Alarm Pareto ActiveX events to trigger scripts

You can assign QuickScripts to Alarm Pareto control events, such as a mouse click or double-click. When the event occurs, the QuickScript runs.

The Alarm Pareto control supports the following events:

- Click
- DoubleClick
- ShutDown

- StartUp

The Click event has one parameter called ClicknBarIndex, which identifies the index of the bar that is selected at run time.

The DoubleClick event has one parameter called DoubleClicknBarIndex, which identifies the index of the bar that is double-clicked at run time.

Click and DoubleClick events are zero-based. When Click and/or DoubleClick events are published, the bar count in the display starts with 0.

Note: The Alarm Pareto control ignores the user interface methods when they are called from the StartUp event, because the control is not visible yet. These methods include: AboutBox() and Refresh().

For more information about scripting ActiveX events, see Scripting ActiveX Controls.

Scripts

You can use the InTouch scripting language, QuickScript, to build more robust applications. There are eight types of scripts and many built-in script functions available.

Scripts can be classified by when they are run and whether they run independently of other ongoing application processes. Scripts can generally be run in two different ways:

- Event-based scripts run once when an event occurs. For example, an event-based script can run after an operator presses a key or a tag value changes.
- Time-based scripts run periodically while a condition is fulfilled. For example, a time-based script can run while a window is open or a button is kept pressed.

You can configure multiple event-based and time-based scripts to run with the same trigger. For example, you can configure a script to run once when a key is pressed and another script to run periodically every 5 seconds while the same key remains pressed.

You can use conditional statements, loops, and local variables in the scripting language to create complex effects in your application.

For condition scripts you can either run a script synchronously or asynchronously.

- When a synchronous script runs, all InTouch animation and tag value updating stops. Then, animation and tag value updating resumes after the script stops.
- When an asynchronous script runs, all InTouch animation and tag value updating continues during the period when the script runs.

The built-in script functions include mathematical functions, trigonometric functions, string functions, and others. Using these functions saves you time in developing your application.

InTouch scripts can include Object Linking and Embedding (OLE) objects and ActiveX controls.

Basic scripting concepts

Before you start scripting, you should understand:

- A script is a set of instructions that direct an application to do something.

- QuickScript is the InTouch HMI scripting language.
- A function is a script that can be called by another script. The InTouch HMI comes with a set of predefined functions for your use.
- QuickFunctions are reusable functions written in QuickScript and stored in the QuickFunction library. To create a QuickFunction, you simply create a QuickScript and name it. A QuickFunction can be called by another script or from animation link expressions.

Types of scripts

In InTouch, scripts are categorized based on what causes the script to run. For example, you would create a "key script" if you want a script to execute when the operator presses a certain key on the keyboard.

After you have chosen the script type, you can then further define the criteria, or conditions, that make the script execute. For example, you might want the script to execute when the key is released, not when the key is pressed.

The script types are:

- **Application** scripts execute either continuously while WindowViewer is running or one time when WindowViewer is started or shut down.
- **Window** scripts execute periodically when an InTouch window is open or one time when an InTouch window is opened or closed.
- **Key** scripts execute one time or periodically when a certain key or key combination is pressed or released.
- **Condition** scripts execute one time or periodically when a certain condition is fulfilled or not fulfilled.
- **Data change** scripts execute one time when the value of a certain tag or expression changes.
- **Action** scripts execute one time or periodically when an operator selects an InTouch HMI graphic object. Action scripts often are used for pushbuttons.
- **ActiveX event** scripts execute one time when an ActiveX event occurs, such selecting the ActiveX control.

Advanced scripting concepts

Some advanced scripting capabilities allow you to achieve sophisticated functions beyond those of the basic InTouch HMI.

OLE objects and ActiveX controls allow you to access your native computer system functions and interact with other programs such as the Manufacturing Engineering Module.

OLE objects

In your custom scripts, you can call OLE objects. OLE objects allow you to access your native computer system functions and to interact with other programs such as the Manufacturing Engineering Module.

For example, using OLE, you can:

- Produce random numbers.
- Create user interface dialog boxes.
- Open the Windows date and time properties panel.

- Read and write to the registry.
- Minimize windows.

Script with ActiveX controls

Several ActiveX controls are provided with the InTouch HMI in the Wizards menu. Because the InTouch HMI is based on the Windows operating environment, you can use nearly any ActiveX control with the InTouch HMI.

Create and edit scripts

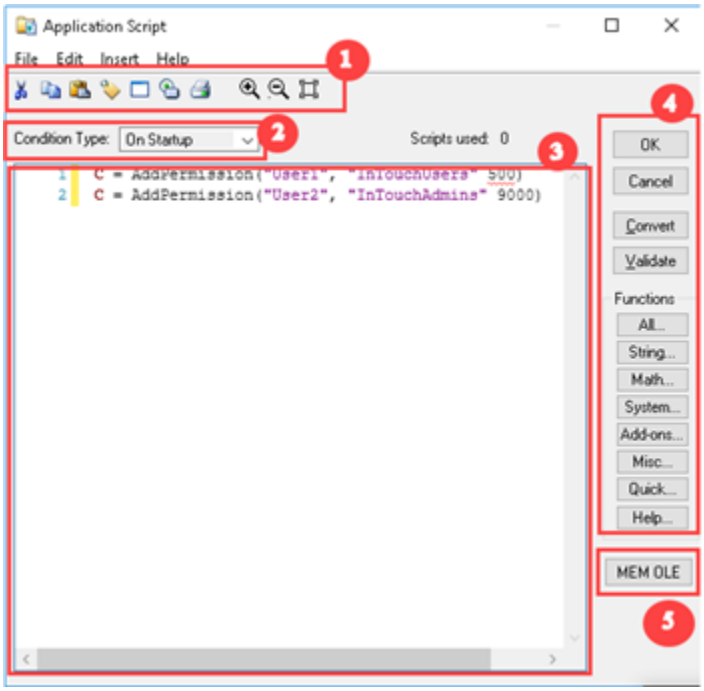
The steps to create a new script vary according to the script type. In general, you open the Script Editor, select a condition type, enter statements, and then save the script.

For detailed information on creating scripts of each type, see the following sections:

- Configure application scripts.
- Configure window scripts.
- Configure key scripts.
- Configure condition scripts.
- Configure data change scripts.
- Configure action scripts.
- Configure ActiveX event scripts.

Work with the InTouch Script Editor

Use the InTouch HMI Script Editor to create and edit scripts within InTouch WindowMaker.



Area	Description
1	Toolbars
2	Condition definition area The **Condition Type** box provides the available execution conditions for the type of script you are writing.
3	Script text window
4	Command buttons
5	Built-in script function buttons
6	MEM OLE button The **MEM OLE** button in the lower right corner only appears if the Manufacturing Engineering Module (MEM) is installed with the InTouch HMI installation. Selecting this button allows you to script with MEM.

This example is for an application script. Each type of script has its own version of the script dialog box, with options and selections that are unique to that type of script.

The title bar of the editor identifies which type of script you are working with. For information about types of scripts, see Types of scripts.

There are text, equivalency and mathematical operator buttons available in a context-sensitive pop-up window that you can select to insert that keyword, function, or symbol into your script at the cursor location.

Color indicators for InTouch script elements

The InTouch Script Editor uses different text colors to identify different script elements. The following table shows the text colors associated with script elements.

Element	**Color**
Keywords	Blue Syntax highlighted while typing.
Comments (both single line and multi-line)	Green Syntax highlighted while typing.
Strings	Purple Syntax highlighted while typing.
Function names, numeric constants, operators, semicolons, dim variables, alias variables, and so on	Black See descriptions for Attribute names and Reserved words.

Element	Color
Attributes, InTouch Tags, Reference Strings	Maroon, bold face
Reserved words	Red, non-bold face
.NET type names	Teal, non-bold face

InTouch Script Editor autocomplete features

These features enable automatic word completion of tags, dotfields, methods, programmatic constructs, and other script elements. The Script Editor autocomplete feature: The InTouch Script Editor autocomplete incorporates several features for use while authoring InTouch scripts. These features enable automatic word completion of tags, dotfields, methods, programmatic constructs, and other script elements. The Script Editor autocomplete feature:

- Provides an autocomplete tag reference in a selectable list box.
- Provides method parameter help in an autocomplete list box including context-specific suggestions covering definitions, keywords, script elements, object and attribute names, and programmatic constructs.

These features serve as convenient documentation of method parameters and scripting syntax as well as an enhanced input method.

Press **Ctrl+Space** to display all available autocomplete options and variables for the selected location in the script. You can identify the context from the icons displayed at the bottom of the autocomplete pop-up window.

Icon	Represents
	Method
	Keywords
	Operators
	Variables
	Tag
	Windows
	ActiveX Instances

Accept InTouch Script Editor autocomplete suggestions

Insert an item at the editor caret from the autocomplete list box—without an end line or tab appended—by doing one of the following:

- Double-click the item
- Highlight (select) the item and press the **Enter** key or the **Tab** key.

Type a space, period, comma, open or closed parenthesis, or other punctuation used in the programming language (: ; [ ] = < > - + / *), and the item highlighted in the autocomplete list box will be inserted at the editor caret with the additional character appended.

Disable script autocomplete

If you do not want to view the autocomplete suggestions while authoring scripts, you can disable the script autocomplete function.

Disable script autocomplete from the WindowMaker configuration screen

1. On the **File** menu, choose **Configure**, and then select **WindowMaker**.

   The WindowMaker configuration screen appears.
2. Select the **Disable script autocomplete** checkbox.

   The autocomplete suggestions will not appear in the script editor.

Disable script autocomplete from the script window

1. Launch the script window.
2. On the **Edit** menu, select **Disable script autocomplete**.

   The autocomplete suggestions will not appear in the script editor.

Multi-level undo and redo in InTouch scripts

You can selectively undo a sequence of changes made to your script. The number of changes that can be undone is limited only by the amount of available memory.

- Use main menu options **Edit**, then **Undo** or **Ctrl+Z** to undo edits. You can also use context menu options to undo and redo.
- Use main menu options **Edit**, then **Redo** or **Ctrl+Y** to redo edits. You can also use context menu options to undo and redo.

An undone change can be redone. Redo mirrors undo changes.

A single undo typically is comprised of sequences of typing or deleting, which can be interrupted by interaction with an autocomplete list or by moving the cursor with the mouse, or by selecting elsewhere in the script.

All pending undo and redo actions are lost if you close the Script Editor or switch to another script within the Script Editor.

Visual indication of InTouch script errors

Errors in InTouch script text are marked with a red "squiggly" underline.

Hovering over the error with the mouse cursor will display the error message as a tooltip. The tooltip error message provides the same information as the message shown when selecting the script verification button.

In some cases, more than one error will be underlined. This is not always possible because some errors prevent the compiler from continuing past the error.

When the "\" character is used to concatenate two strings to form a path, the script editor displays a red underline under "\". This is an exception and can be ignored. The path is constructed correctly in runtime.

```
1  TagComment = InfoIntouchAppDir() + "\";
```

Manage InTouch script edits

The InTouch Script Editor offers tools and visual references to find and manage edits.

Script Line Numbers

The Script Editor displays line numbers at the left margin.

- Line numbers up to four digits appear when the Script Editor is not zoomed.
- Use the right-click context menu **Go To** option to go to a specific line in a script.
- Delete the current line of text by pressing **Ctrl+L**.
- You can select and drag a line of text to a different line.

Change Bars

As a visual reference for script changes in-progress, yellow change bars on the left margin of the script text window indicate additions and line insertions and edits.

Tag Definition

You can enter a new tagname in the InTouch Script Editor, then press **Ctrl+T** or select **Edit** on the main menu to define a tag. The **Tagname Dictionary** appears so you can complete the tagname definition.

Find and Replace

The InTouch Script Editor provides customizable find and replace functionality. For information on how to use the find and replace feature, see Search within a script.

Open a script for editing

The steps to open an existing script vary slightly depending on the script type.

Open an application script

1. In the **Scripts** pane, double-click **Application**.

   The Application Script dialog appears.
2. In the **Condition Type** list, select the type of script to edit.

Open a window script

1. In the **Windows** pane, right-click the window name, and then select **Window Scripts**.

   Alternatively, open the window that the script is associated with. Right-click on a blank area in the window, and then select **Window Scripts**.
2. In the **Condition Type** list, select the condition to cause the script to run.

Open an ActiveX event script

- Do any of the following:
  - In the **Scripts** pane, expand **ActiveX Event**, and then double-click the script name.
  - Double-click the ActiveX control instance that the script is associated with. Select the **Events** tab, and then double-click the cell that contains the script name.

Open an action script

1. Open the window that contains the graphic element that the action script is associated with. A typical use for an action script would be to script an action on a button.
2. Double-click the graphic element that the action script is associated with.
3. In the **Touch Pushbuttons** area, select **Action**. The Script Editor appears.
4. In the **Condition Type** list, select the action to cause the script to run.

Open key, condition, or data change scripts

1. In the **Scripts** pane, expand the script category, and then double-click the script name.
   - The Script Editor appears. Choose the **Browse** button, and select the script name.
2. If applicable, in the **Condition Type** list, select the condition to cause the script to run.

Save or discard changes to a script

While working in the Script Editor, or when finished, you can save your script either manually or automatically. Or you can discard it altogether.

The restore option is not available for window and application scripts.

Note: Saving and discarding changes always applies to all condition types for a type of script, not just the condition type that is currently visible.

Save changes and keep the script open

- On the **Script** menu, select **Save**.

Save changes and close the script

- Select **OK**.

Discard changes and keep the script open

- Select **Restore**.

Discard changes and close the script

- Select **Cancel**.

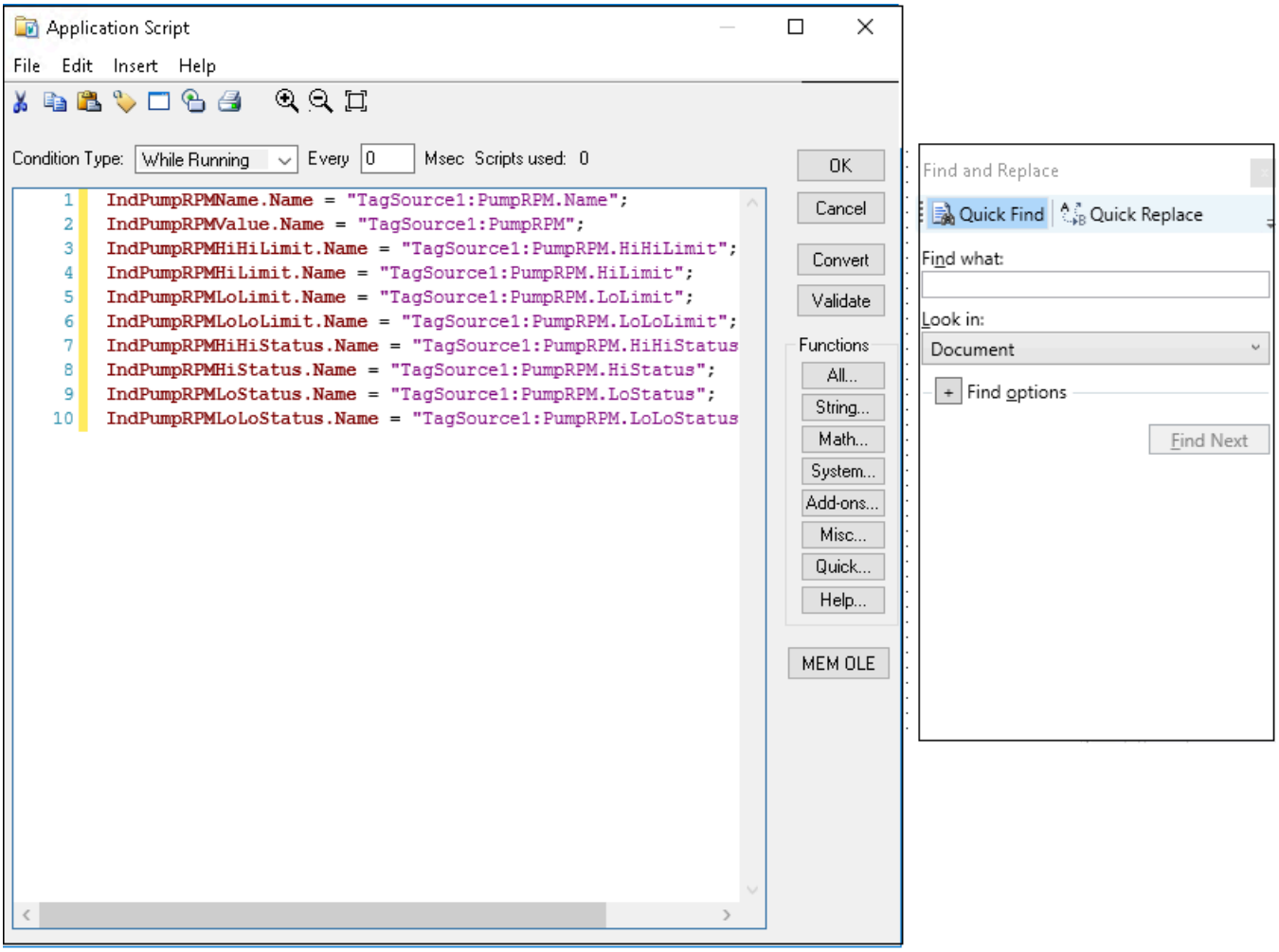
Copy, cut, and paste text

Copying, cutting and pasting text in the Script Editor works the same way as in any other Windows application. Use either the standard keyboard shortcuts Ctrl-C, Ctrl-X, and Ctrl-V, or the toolbar buttons.

You can also select a line and drag the text in that line to a new position in the script.

Search within a script

The InTouch Script Editor provides the functionality to find and replace text within an open script based on a search string and a set of customizable search options. You can access **Find** and **Replace** as **Edit** options on the Script Editor's menu bar.



You can also use keyboard shortcuts **Ctrl+F** (Quick find) and **Ctrl+H** (Quick replace) to display the **Find and Replace** dialog box.

- In a simple search using the default search options, select the **Quick Find** tab and type a word or phrase in the **Find what** field.

  Select **Find Next** to start a search. An amber background identifies a word or phrase in the script that matches the search string

- In a simple replace operation using the default options, select the **Quick Replace** tab, type a word or phrase in the **Find what** field, and also type a replacement word or phrase in the **Replace with** field.

  Select **Find Next** to start a replacement operation. A light blue background identifies a word or phrase in the script that matches the search string entered in the **Find what** field. You have three replacement options after locating a matching word or phrase.

  - Select **Find Next** to ignore the current matching word or phrase and continue searching for the next match in the script.
  - Select **Replace** to replace the current matching word or phrase with the string entered in the **Replace with** field.
  - Select **Replace All** to replace all matching words or phrases with the string entered in the **Replace with** field.

Configure a find or replace

The **Find and Replace** dialog box provides a set of options to configure a set of search options

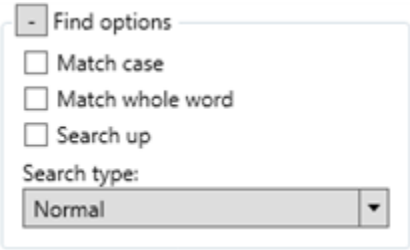
Look in

The **Look in** field includes options to search the entire script (**Document**) or only a selected portion of a script (**Selection**). **Document** is the default.

When you want to search only within a selected portion of a script, select the search area within the script using your mouse before conducting the search. A blue background identifies the selected script lines, which can be searched. The search results only show matching items within the portion of the script you selected.

Find Options

You can expand or collapse the **Find options** section. The following options can be selected or cleared to filter search results more precisely.



- **Match case**

  When selected, the search results only display instances of the **Find what** string that are matched both by content and by case. For example, a search for Triangle4 with **Match case** selected returns Triangle4 but not triangle4.
- **Match whole word**

  When selected, the search results only display instances of the **Find what** string that are matched in complete words. For example, a search for LogicBit will return LogicBit but not LogicBits.
- **Search up**

  When selected, a search is conducted from the current position within a script to the top of the script. By default, a search is conducted from the current position within a script to the bottom.

Search type

The **Search type** field provides options to conduct a script search based on the type of search.

- **Normal**

  The default search type, which requires an exact match between the characters in a search string and text in a script.
- Search by regular expressions

  A regular expression describes one or more strings to match when you search a script. A regular expression consists of ordinary characters that serve as a template for matching a character pattern to the string being

searched.

- Search by wildcard characters

  A wildcard search uses keyboard characters like an asterisk (*) or a question mark (?) to represent one or more characters when searching within a script.

- Search by acronym

  An acronym search matches a character at the start of a word, then every capital letter or character following an underscore.

- Search by shorthand

  A shorthand search extends the 'Search by Acronym' option by allowing non-whitespace characters between the search pattern characters.

You must type a search string that complies with the syntax and supported characters of the selected search type.

Search by wildcard characters

A wildcard search uses a single keyboard character, which can be interpreted as a number of literal characters or an empty string when you are searching for a string in a script.

Wildcard characters are often used in place of one or more characters when you do not know what the real character is or you do not want to type the entire search string.

Wildcard character	Uses
Asterisk (*)	An asterisk in a search string matches any sequence of characters. Use the asterisk as a substitute for zero or more characters. **Examples** • Logic* Finds Logic1, LogicTest, but not ALogicTest • *Test* Finds LogicTest1, PumpTestABC, but not LogicTst1
Question mark (?)	A question mark in a search string matches any character at a single position within a search string. **Examples** • LogicTest? Finds LogicTest1, LogicTestA, but not ALogicTest1 • LogicTest?2 Finds LogicTest12, but not LogicTest13

Search by regular expressions

A regular expression describes one or more strings to match when you search a script using alphanumeric characters and special characters known as metacharacters. The regular expression serves as character pattern to compare with the script text being searched.

Regular expressions are constructed much like arithmetic expressions are created. Small expressions are combined by using a variety of metacharacters and operators to create larger expressions.

The components of a regular expression can be individual characters, sets of characters, ranges of characters, or choices between characters. Components can also be any combination of these components.

Script Regular Expressions

Regular Expression	Purpose	Example
.	Match any single character (except a line break)	s.e matches "ste" in "step" and "sfe" in "transfer" but not "acro" in "across".
*	Match zero or more occurrences of the preceding expression (match as many characters as possible)	a*r matches "r" in "rack", "ar" in "ark", and "aar" in "aardvark"
.*	Match any character zero or more times (Wildcard *)	c.*e matches "cke" in "racket", "comme" in "comment", and "code" in "code"
+	Match one or more occurrences of the preceding expression (match as many characters as possible)	e.+e matches "eede" in "feeder" but not "ee".
.+	Match any character one or more times (Wildcard ?)	e.+e matches "eede" in "feeder" but not "ee".
*?	Match zero or more occurrences of the preceding expression (match as few characters as possible)	e.*?e matches "ee" in "feeder" but not "eede".
+?	Match one or more occurrences of the preceding expression (match as few characters as possible)	e.+?e matches "ente" and "erprise" in "enterprise", but not the whole word "enterprise".

^	Anchor the match string to the beginning of a line or string	^car matches the word "car" only when it appears at the beginning of a line.
\r?$	Anchor the match string to the end of a line	End\r?$ matches "end" only when it appears at the end of a line.
[abc]	Match any single character in a set	b[abc] matches "ba", "bb", and "bc".
[a-f]	Match any character in a range of characters	be[n-t] matches "bet" in "between", "ben" in "beneath", and "bes" in "beside", but not "below".
()	Capture and implicitly number the expression contained within parenthesis	([a-z])X\1 matches "aXa"and "bXb", but not "aXb". ". "\1" refers to the first expression group "[a-z]".
(?!abc)	Invalidate a match	real (?!ity) matches "real" in "realty" and "really" but not in "reality." It also finds the second "real" (but not the first "real") in "realityreal".
[^abc]	Match any character that is not in a given set of characters	be[^n-t] matches "bef" in "before", "beh" in "behind", and "bel" in "below", but not "beneath".
|	Match either the expression before or the one after the symbol.	(sponge|mud)bath matches "spongebath" and "mudbath."
|\^	Escape the character following the backslash	
{x},	Specify the number of occurrences of the preceding character or group	x(ab){2}x matches "xababx", and x(ab){2,3}xmatches "xababx" and "xababababx" but not "xabababababx".
\p{X}	Match text in a Unicode character class, where "X" is the	\p{Lu} matches "T" and "D" in "Thomas Doe".

	Unicode number.	
\b	Match a word boundary	\bin matches "in" in "inside" but not "pinto".
\r?\n	Match a line break (ie a carriage return followed by a new line).	End\r?\nBegin matches "End" and "Begin" only when "End" is the last string in a line and "Begin" is the first string in the next line.
\w	Match any alphanumeric character	a\wd matches "add" and "a1d" but not "a d".
(?[^\r\n])\s	Match any whitespace character.	Public\sInterface matches the phrase "Public Interface".
\d	Match any numeric character	\d matches and "3" in "3456", "2" in 23", and "1" in "1".
\uXXXX where XXXX specifies the Unicode character value	Match a Unicode character	\u0065 matches the character "e".
\b(\w+|[\w-[0-9\]]\w*)\b	Match an identifier	Matches "type1" but not &type1" or "#define".
((\".+?\")|('.+?'))	Match a string inside quotes	Matches any string inside single or double quotes.
\b0[xX]([0-9a-fA-F])\b	Match a hexadecimal number	Matches "0xc67f" but not "0xc67fc67f".
\b[0-9]\.\[0-9]+\b	Match integers and d	Matches "1.333".

Order of precedence

A regular expression is evaluated from left to right and follows an order of precedence.

The following table contains the order of precedence of regular expression operators, from highest to lowest.

Operator or operators	Description
\	Escape
(), (?:), (?=), []	Parentheses and brackets
*, +, ?, {*n*}, {*n*,}, {*n*,*m*}	Quantifiers

^, $, *anymetacharacter*	Anchors and sequences
|	Alternation

Characters have higher precedence than the alternation operator, which, for example, allows "m|food" to match "m" or "food".

Search by acronym

Acronym searching matches a character at the start of a word, then every capital letter or character following an underscore.

Example: The search term "LB" finds instances of "LogicBits".

Search by shorthand

The shorthand search extends the 'Search by Acronym' option, by allowing any number of non-whitespace characters between the search pattern characters.

Example: The search term "ta" finds instances of "Triangle".

Insert code elements

You can automatically insert various code elements into your script by selecting them from lists. This saves you time and reduces the risk of typing errors. Access the code elements one of two ways: using the main menu **Insert** options, or by using the autocomplete pop-up windows and selecting your element from the list.

Code elements you can insert, both from autocomplete pop-up windows and from the Script Editor menu items include:

- Functions
- Tagnames and Dotfields For a dotfield, type a tagname followed by a period to open an autocomplete pop-up window.
- Variable
- Window name
- ActiveX instances
- Keyword
- Operator

The autocomplete pop-up windows include a row of icons at the bottom of the pop-up for direct access to the available elements.

Access help for script functions

If you are looking for help on a specific script function, you can access it directly from the Script Editor.

View help on a specific script function

1. In the bottom right corner of the Script Editor, select **Help**.

   A list of functions appears.
2. Select the name of the function from the list to show its help.

   The corresponding Help topic appears.

Validate scripts for correct syntax

When you save a script, the Script Editor automatically checks it for correct syntax. If an error is discovered, a message with more information appears. You must fix all syntax errors before you can save the script. You can also start the validation manually while you are editing the script.

Manually validate script syntax

- Select **Validate**.

Highlight script errors

The InTouch Script Editor also highlights script errors. For more information, see Visual Indication of InTouch Script Errors.

Print scripts

You can print scripts individually from the Script Editor, or you can print all scripts of a specific type using the print feature in WindowMaker.

You can print scripts individually from the Script Editor, or you can print all scripts of a specific type using the print feature in WindowMaker.

Print an individual script

1. Open the script in the Script Editor.
2. Select **Print** in the toolbar. The script is printed to the Windows default printer.

Print all scripts of a specific type

1. On the quick access toolbar, select **Print**.

   The **WindowMaker Printout** dialog box appears.
2. To print window scripts, do the following:
   a. Select **Windows**.
   b. Select the windows to print:

      All prints the information for all windows in the application.

      Selected prints only the information for specific windows. The **Windows to Print** dialog box appears. Select the windows in your application you want to print and select **OK**.

      Batch prints only the information for windows specified in a .csv file.

c. Select **Window Scripts** to print the scripts associated with the windows.

3. To print other types of scripts, select the appropriate checkboxes. To print all scripts, select **All Scripts**.
4. Select **Next**. The **Select Output Destination** dialog box appears.
5. Do one of the following
   - Select **Send output to Printer**.
   - Select **Send output to Text File**.
6. Select **Browse** to select a printer or to find a file.
7. Select **Print**.

Print all scripts

1. Select **All Scripts** to print all scripts used in the application.

   You can restrict printing to only selected types of scripts by clearing the **All Scripts** checkbox. Then, select the checkbox for each type of script that you want to print.
2. Select **Next**. The **Select Output Destination** dialog box appears.
3. Select the option to print the contents of the Tagname Dictionary or send the output to a text or .html file.
4. Select **Print**.

Delete scripts

The steps to delete a script vary depending on the script type. See the following sections:

- Configure application scripts.
- Configure window scripts.
- Configure key scripts.
- Configure condition scripts.
- Configure data change scripts.
- Configure action scripts.
- Configure ActiveX event scripts.

Adjust zoom options to view scripts

You can adjust the zoom options to view scripts, such as zoom in, zoom out, or zoom normal.

Zoom in

1. Open Script Editor.
2. On the Script Editor toolbar, select **Zoom in**.

   Alternatively, choose **Edit**, and select **Zoom in**.

Zoom out

1. Open Script Editor.

2. On the Script Editor toolbar, select **Zoom out**.

   Alternatively, choose **Edit**, and select **Zoom out**.

To zoom normal

1. Open Script Editor.
2. On the Script Editor toolbar, select **Zoom normal**.

   Alternatively, select **Edit**, and select **Zoom normal**.

Script triggers

All InTouch HMI scripts are executed by script triggers. Each script type has one or more triggers to launch it.

In the Script Editor, you can select which script trigger you want to use to execute your script. You select a script trigger based on when and how a script is executed.

You can configure various triggers based on user actions, internal states, and changes of tagname values. User actions include pressing keys and selecting graphic elements. Internal state triggers can include starting WindowViewer.

Scripts are triggered by these kinds of actions:

- Starting and shutting down WindowViewer. See Configure application scripts.
- Opening and closing a window. See Configure window scripts.
- Pressing a key or key combination. See Configure key scripts.
- Fulfilling a certain condition, such as tagname or an expression value. See Configure condition scripts.
- Changing tagname values or tagname field values. See Configure data change scripts.
- Selecting a graphic object. See Configure action scripts.
- Events that occur in an ActiveX control, such as selecting the control. See Configure ActiveX event scripts.

Also, you can pause script execution. By default, when WindowViewer is started, logic is running and scripts are executed. You can pause script execution at run time by halting logic. After pausing you can resume script execution. For more information, see Pause script execution at run time.

Types of script triggers

In the InTouch HMI, scripts are divided into seven types. Each type of script has one or more triggers you can select to launch a script.

- An application script has three triggers: on startup, on shutdown, and while running. Each trigger can execute a different script.
- A window script has three triggers: on show, on hide, and while showing. Each trigger can execute a different script.
- A key script has three triggers: on key up, on key down, or while down. Each trigger can execute a different script.
- A condition script has four triggers: on true, while true, on false, and while false. Each trigger can execute a

different script.

- A data change script executes when the value of a certain tag or expression changes.
- An action script executes one time or periodically when an operator selects an InTouch HMI graphic object.
- An ActiveX event script executes one time when a certain ActiveX event occurs, such as selection of the ActiveX control.

Use multiple triggers

For most script types you can use multiple triggers and associate different scripts with each trigger.

For example, you can configure an application script to execute one script when WindowViewer is started, and another script periodically while WindowViewer is running.

Select the trigger in the **Condition Type** list to view the existing script for a trigger.

Periodic script execution

Scripts that execute periodically do not execute immediately after triggering, but after the specified period for the first time.

For example, if you configure a key script to execute every 5000 ms while a specific key is pressed, it executes 5 seconds after the key is pressed and held down and then every 5 seconds afterwards.

Configure application scripts

Application scripts are linked to the entire InTouch HMI application. You can use application scripts to:

- Execute a script one time when WindowViewer is started.
- Execute a script periodically while WindowViewer is running.
- Execute a script one time when WindowViewer is shut down.

Configure an application script

1. In the **Scripts** pane, right-click on **Application** and then select **Open**.

   The **Application Script** dialog box appears.

   Embedded Image (65% Scaling) (LIVE)
2. In the **Condition Type** list, select the condition for the script execution:
   - Select **On Startup** to configure a script to execute one time when WindowViewer is started.
   - Select **While Running** to configure a script to execute periodically while WindowViewer is running.
   - Select **On Shutdown** to configure a script to execute one time when WindowViewer is shut down.
3. If you selected **While Running** in the previous step, type a time interval between 1 and 360000 milliseconds in the **Every** box. The time interval specifies how often the script is executed.
4. Type your script in the window.
5. Select **OK**.

Delete an application script

1. In the **Scripts** pane, right-click on **Application** and then select **Open**.
   The **Application Script** dialog box appears.
2. In the **Condition Type** list, select the condition for the script to delete.
   The script appears in the main section of the **Application Script** dialog box.
3. On the **Edit** menu, select **Clear**.
   The script from the main section clears and the associated script is deleted.

Limitations of application scripts

Application scripts that are executed when WindowViewer starts or shuts down have limitations on their interaction with other objects.

You cannot use On Startup application scripts to:

- Reference ActiveX methods, properties, or events.
- Read from or write to controls and I/O tagnames or remote references.
- Run data change scripts and condition scripts.

You cannot use On Shutdown application scripts to:

- Read from or write to controls and I/O tagnames or remote references.
- Start other applications.

Configure window scripts

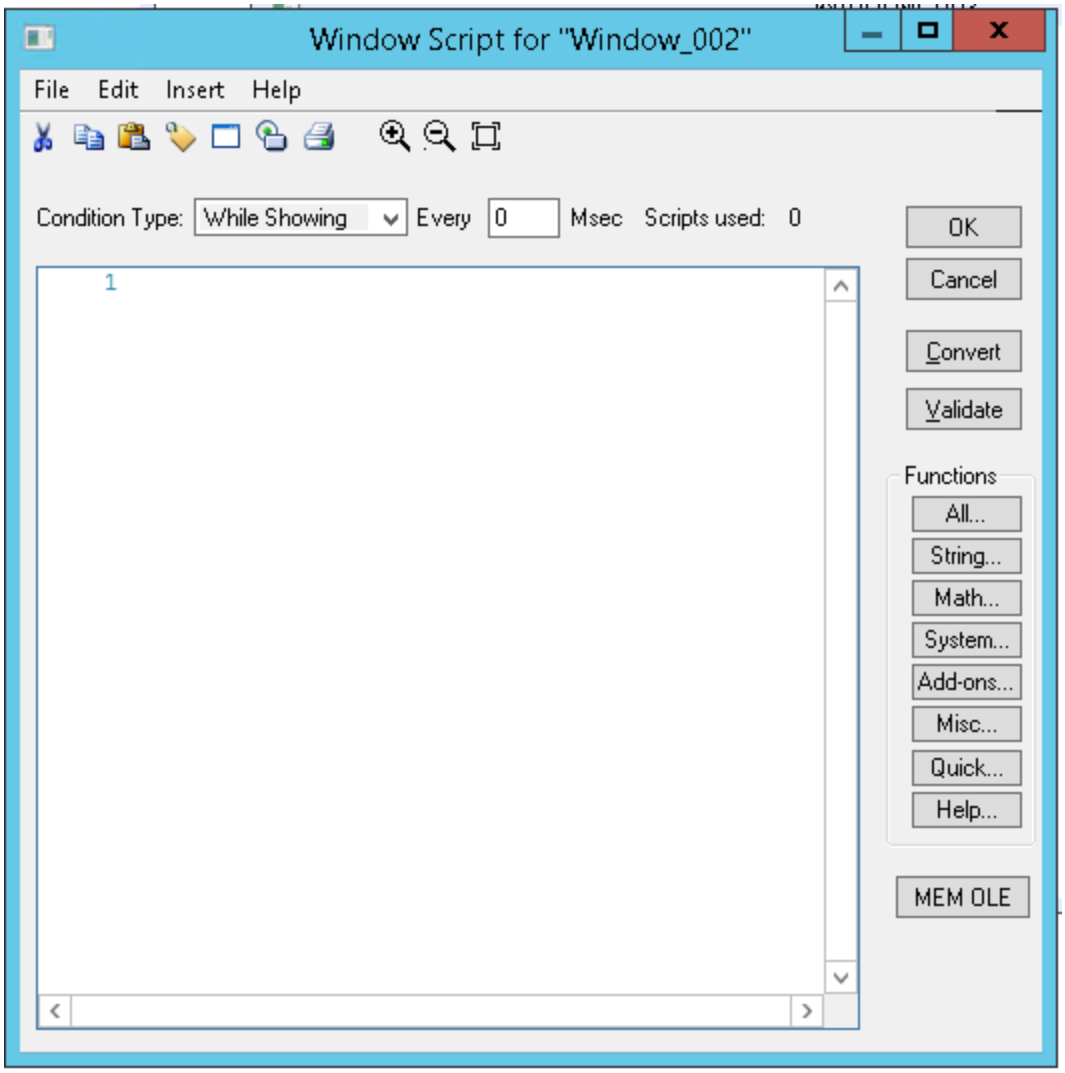
Window scripts are scripts that are linked to specific windows. You can use the **GetWindowName** script function to help the run-time environment reduce scripting necessary to load windows. You can use window scripts to:

- Execute a script one time when an InTouch window is opened.
- Execute a script periodically while an InTouch window is open.
- Execute a script one time when an InTouch window is closed.

Note: Opening an InTouch window is also referred to as "showing an InTouch window." Closing an InTouch window is also referred to as "hiding an InTouch window."

Configure a window script

1. In the **Windows** pane, right-click on a window and then select **Window Scripts**.
   The **Window Script for Window Name** dialog box appears.



2. In the **Condition Type** list, do one of the following:
   - Select **On Show** to configure a script to execute one time when the associated window is started.
   - Select **While Showing** to configure a script to execute periodically while the associated Window is open.
   - Select **On Hide** to configure a script to execute one time when the associated window is closed.
3. If you select **While Showing** in the previous step, type a time interval between 1 and 360000 milliseconds in the **Every** box.
4. Type your script in the window.
5. Select **OK**.

Delete a window script

1. In the **Windows** pane, right-click on a window and select **Window Scripts**. The **Window Script for Window Name** dialog box appears.
2. In the **Condition Type** list, select the script trigger for the script to delete. The script appears in the main section of the **Window Script for Window Name** dialog box.
3. On the **Edit** menu, select **Clear**.

Important: Do not use on hide scripts to read from or write to I/O tagnames. The I/O value update does not necessarily complete before the window is hidden.
To read from or write to I/O tagnames when a window closes, configure a data change script and activate it from an on hide script.

Configure key scripts

Key scripts are scripts that are linked to the pressing of a specific key or key combination. You can use key scripts to:

- Execute a script one time when a key or key combination is pressed.
- Execute a script periodically while a key or key combination is pressed and not released.
- Execute a script one time when a key or key combination is released.

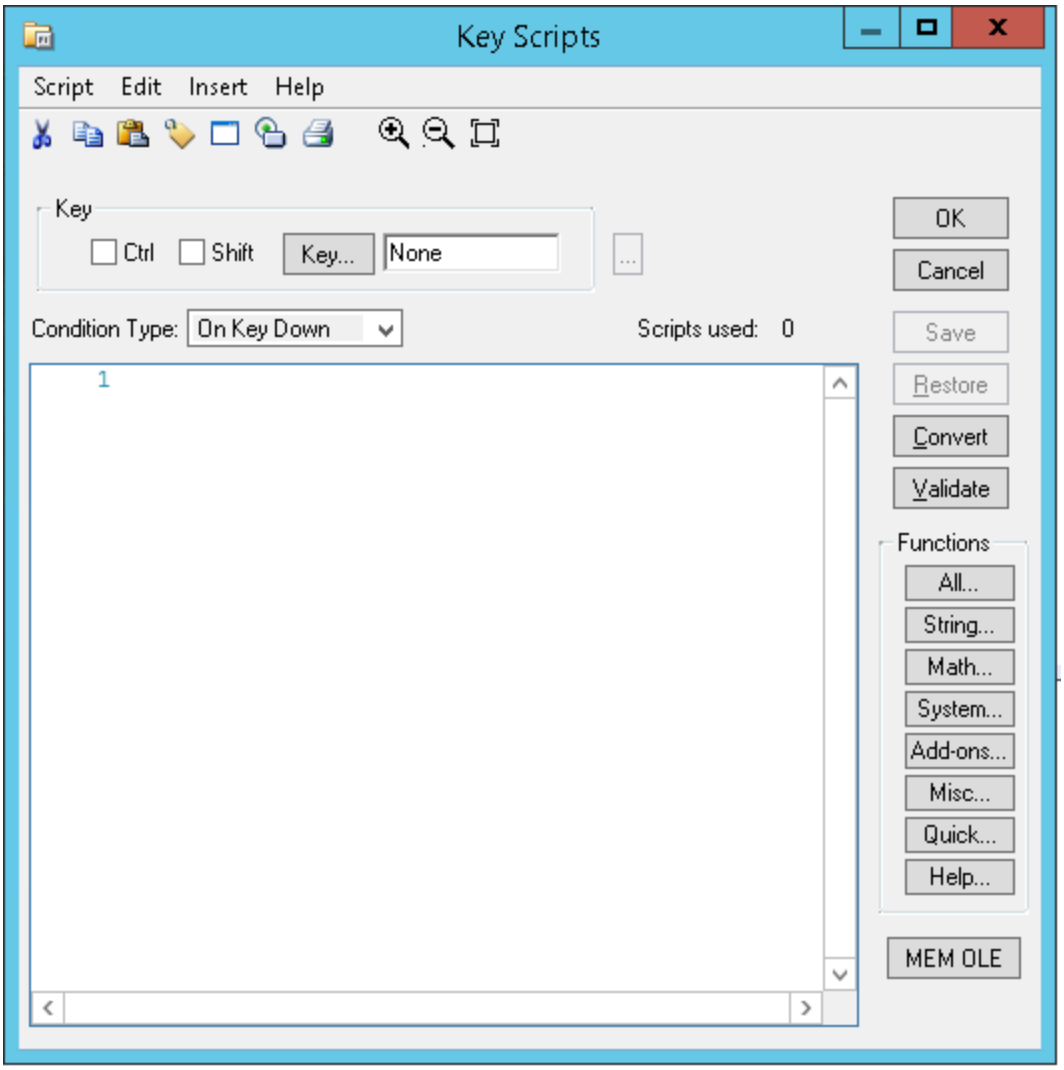
A key script is identified by the name of key that initiates the script. For example: Ctrl+q.

Note: If you have configured an action script that uses the same key or key combination to trigger it, the key script is ignored and instead the action script is executed.

Configure a key script

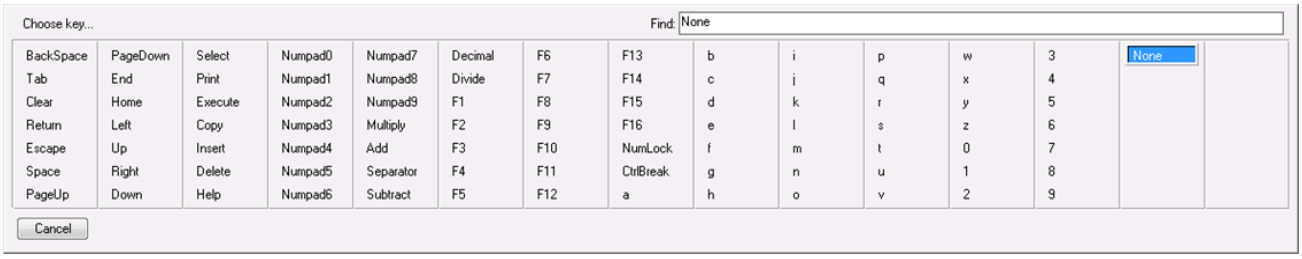
1. In the **Scripts** pane, do one of the following:
   - To configure a new key script, right-click **Key**, and then select **New**.

     The **Key Scripts** dialog box appears.



- To configure an existing key script, expand **Key**, right-click the script name, and then select **Edit**. The **Edit Key Script** dialog box appears.

2. Select **Key** and select a key from the **Choose Key** dialog box.



3. Select the **Ctrl** and/or **Shift** checkboxes to assign a control key and/or shift key combination with your selected key.
4. In the **Condition Type** list, do one of the following:
   - Select **On Key Down** to configure a script to execute one time when the associated key or key combination is pressed.
   - Select **While Down** to configure a script to execute periodically while the associated key or key combination is pressed.

- Select **On Key Up** to configure a script to execute one time when the associated key or key combination is released.

5. If you selected **While Down** in the previous step, type a time interval between 1 and 360000 milliseconds in the **Every** box.
6. Type your script in the window.
7. Select **OK**.

Delete all key scripts associated with a key

- In the **Scripts** pane, expand **Key**, right-click the key script name, and then select **Delete**. When a message appears, select **Yes**.

Delete a key script that is associated with a key

1. Using the **Classic View**, in the **Scripts** pane, expand **Key**, right-click the key script name, and then select **Edit**. The **Edit Key Script** dialog box appears.
2. In the **Condition Type** list, select the script trigger for the script to delete. The script appears in the main section of the **Edit Key Script** dialog box.
3. On the **Edit** menu, select **Clear**. The script from the main section clears and the associated script is deleted.

Configure condition scripts

Condition scripts are triggered depending on when certain logical conditions are fulfilled. Use condition scripts to execute a script:

- One time when a condition is fulfilled.
- One time when a condition is not fulfilled.
- Periodically while a certain condition is fulfilled.
- Periodically while a certain condition is not fulfilled.

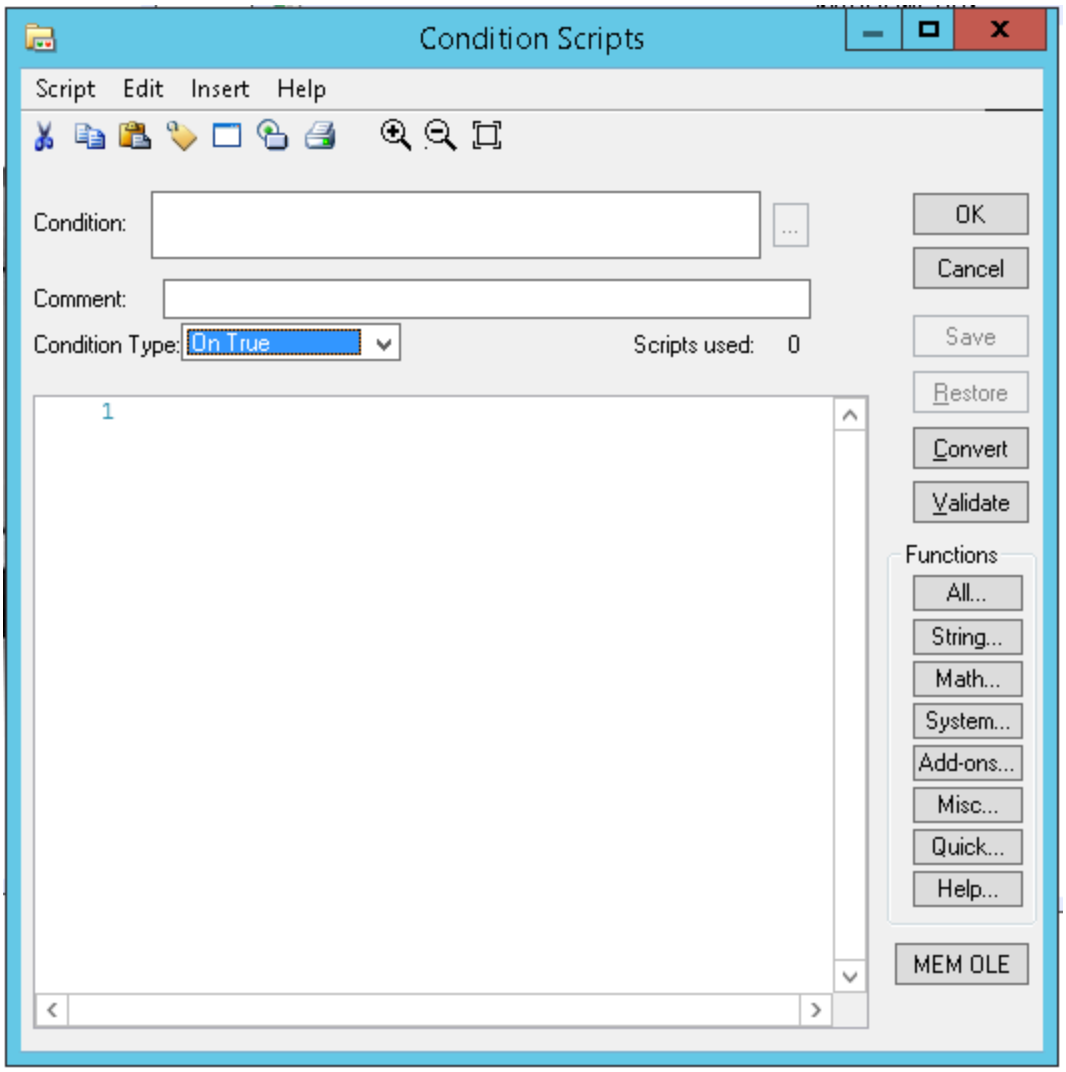
A condition script is identified by the condition syntax that initiates the script. For example: tag1>=13.

Note: A script that is assigned the On True condition type only executes if the condition transitions from False to True. A script that is assigned the On False condition type only executes if the condition transitions from True to False.

Configure a condition script

1. In the **Scripts** pane, right-click **Condition,** and select **New**.

   The **Condition Scripts** dialog box appears.



- To edit an existing condition script, select the plus sign next to **Condition**, right-click the condition script name, and select **Edit**. The **Edit Condition Script** dialog box appears.

2. In the **Condition** box, type the expression that you want to use as the condition.

   You can type the expression to a maximum length of 1024 characters.
3. You can enter a comment in the **Comment** box.
4. In the **Condition Type** list, do one of the following:
   - Select **On False** to configure a script to execute one time when the condition becomes false.
   - Select **While False** to configure a script to execute periodically while the condition is false.
   - Select **On True** to configure a script to execute one time when the condition becomes true.
   - Select **While True** to configure a script to execute periodically while the condition is true.
5. If you selected **While False** or **While True** in the previous step, type a time interval between 1 and 360000 milliseconds in the **Every** box.

   Note: The conditional WindowViewer timers will stop themselves if the condition is no longer true. For example, While Mouse Button Down events will not trigger if the mouse button is no longer down, and key scripts will stop if keys are no longer down.
6. Type your script, or modify the existing script in the window.

7. Select **OK**.

Delete all condition scripts that are associated with a condition

- In the **Scripts** pane, expand **Condition**, right-click the condition script name and select **Delete**. When a message appears, select **Yes**.

Delete individual condition scripts that are associated with a condition

1. In the **Scripts** pane, expand **Condition**, right-click the key script name and select **Edit**. The **Edit Condition Script** dialog box appears.
2. In the **Condition Type** list, select the script trigger for the script to delete. The script appears in the main section of the **Edit Condition Script** dialog box.
3. On the **Edit** menu, select **Clear**. The script from the main section clears and the associated script is deleted.

Configure data change scripts

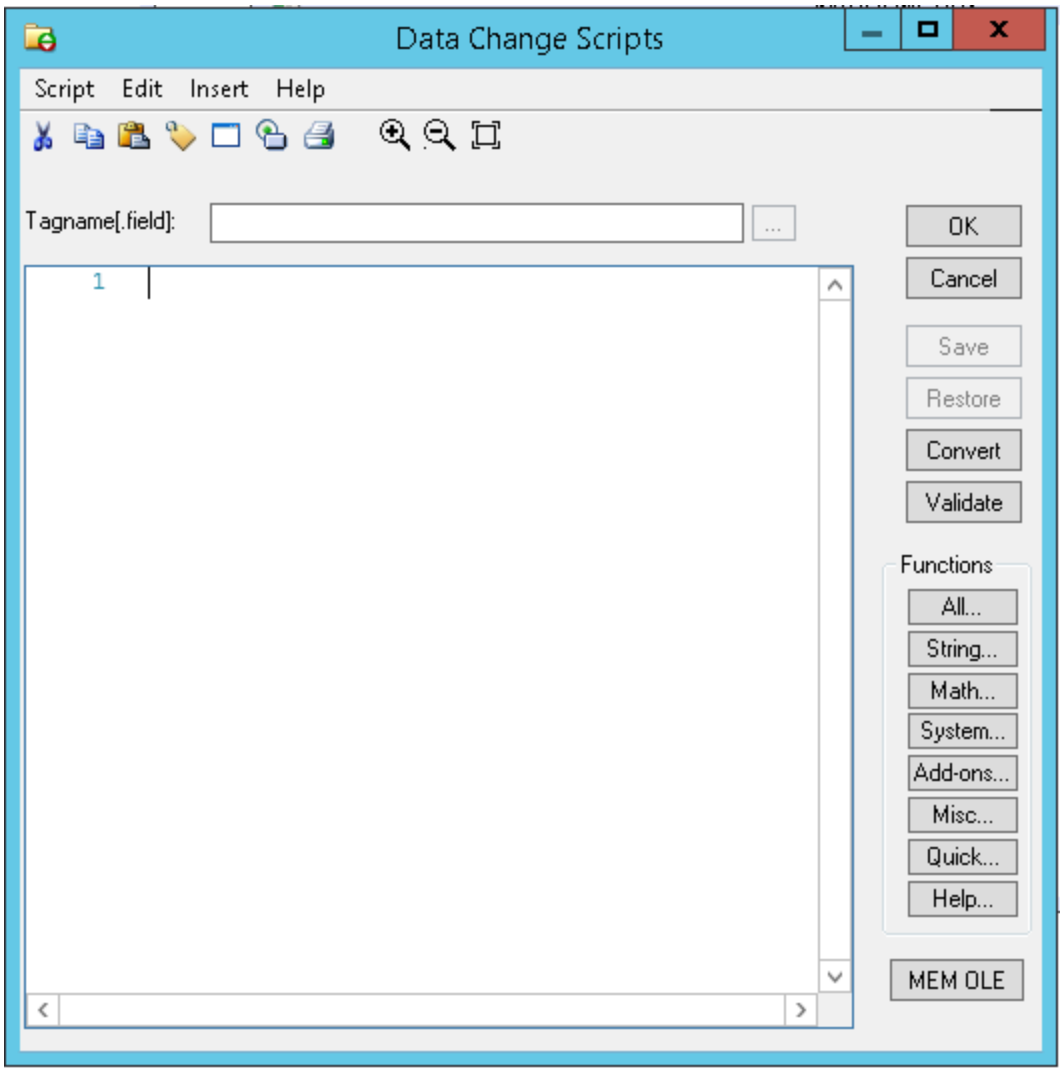
You can use data change scripts to execute a script one time when a certain tagname or dot field changes by more than its defined dead band.

A data change script is identified by the tagname or tagname field that initiates the script. For example: Tag1 or Tag2.HiHiLimit.

Configure a data change script

1. In the **Scripts** pane, right-click **Data Change** and select **New**.

   The **Data Change Scripts** dialog box appears.



2. To create a new script, in the **Tagname[.field]** box, enter a tagname or tagname field.

   To edit an existing script, select the ellipsis button to the right of the **Tagname[.field]** box and select the script from the list that appears.
3. Type your script in the window.
4. Select **OK**.

Delete a data change script

- In the **Scripts** pane, expand **Data Change**, right-click the data change script name and select **Delete**. When a message appears, select **Yes**.

Configure action scripts

Use action scripts to associate operator actions with graphic objects. You can configure one or more of the following events with a graphic object:

- Selecting the left, center, or right mouse button.

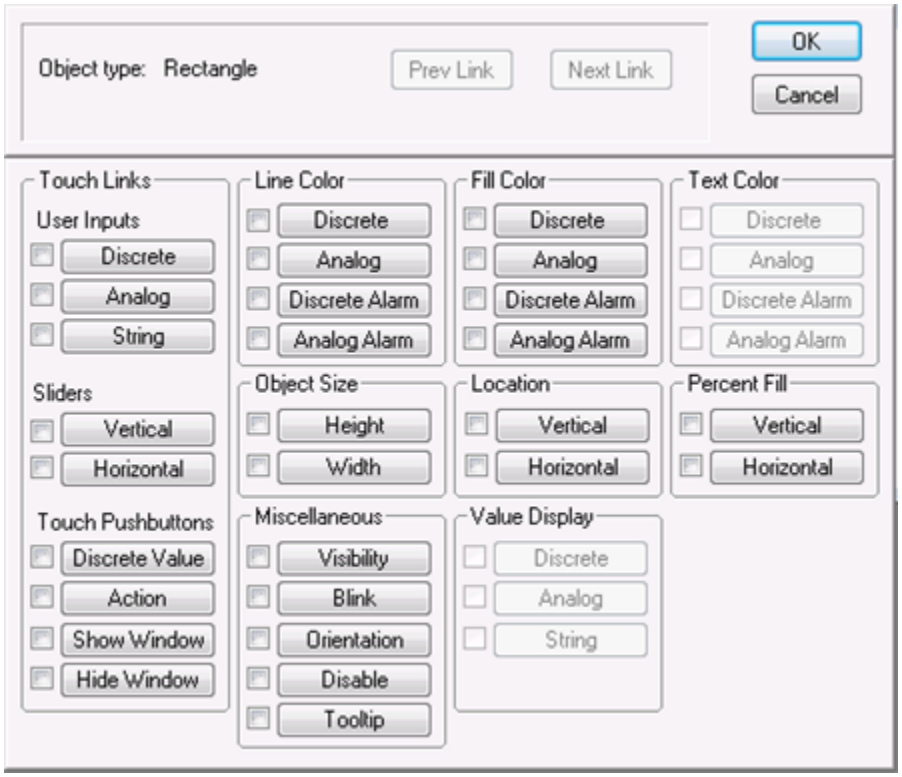
- Selecting and holding the left, center, or right mouse button.
- Releasing the left, center, or right mouse button.
- Double-clicking the left, center, or right mouse button.
- Pressing a key or key combination.
- Pressing and holding a key or key combination.
- Releasing a key or key combination.
- Moving a mouse pointer over an object.

An action script can only be configured in the **Animation Link Selection** panel of the object itself.

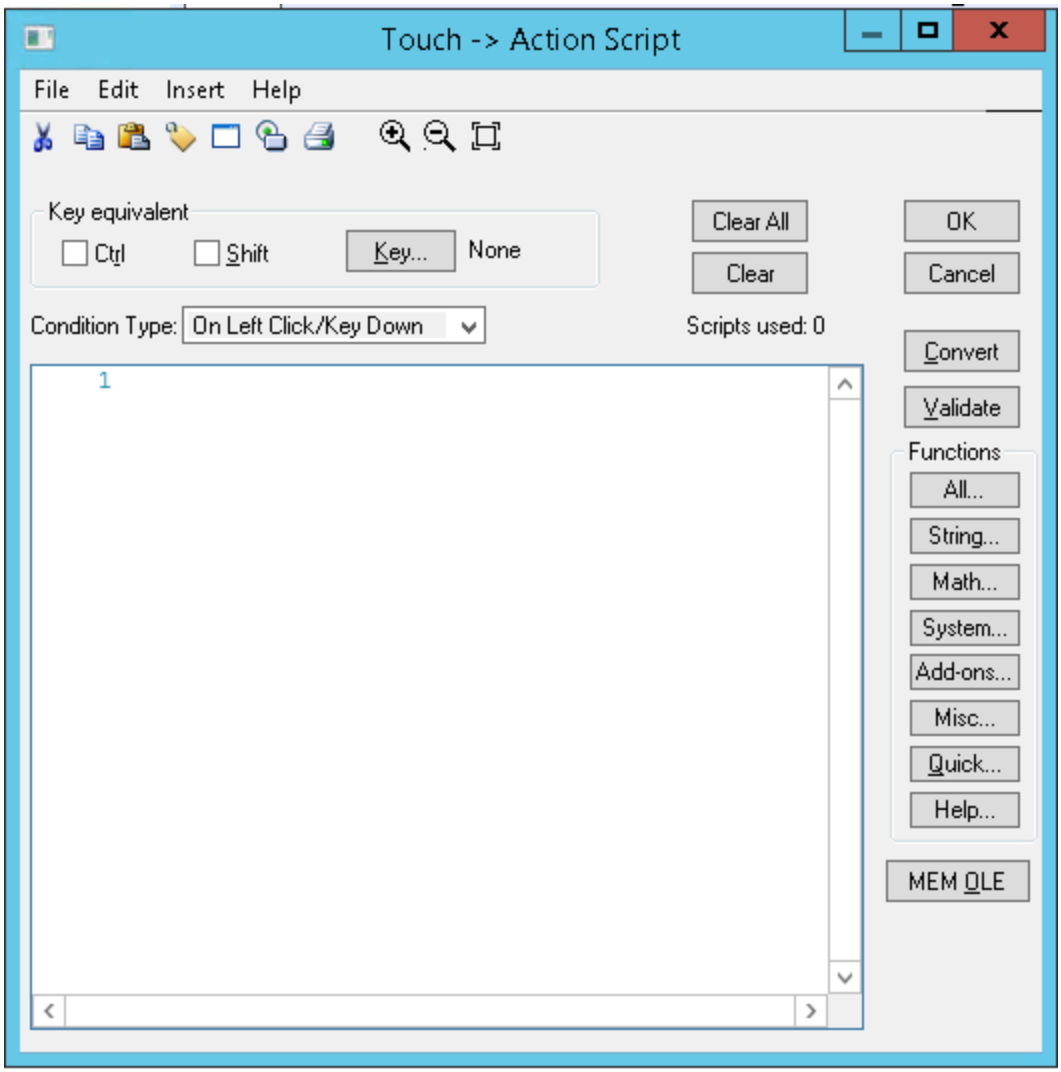
Important: If a key script exists that is triggered by the same key or key combination as the action script, the action script is executed and the key script is ignored.

Configure an action script

1. Double-click the graphic object. The **Animation Links Selection** panel appears.



2. Select **Action**. The **Touch -> Action Script** dialog box appears.



3. In the **Condition Type** list, select one of the following:

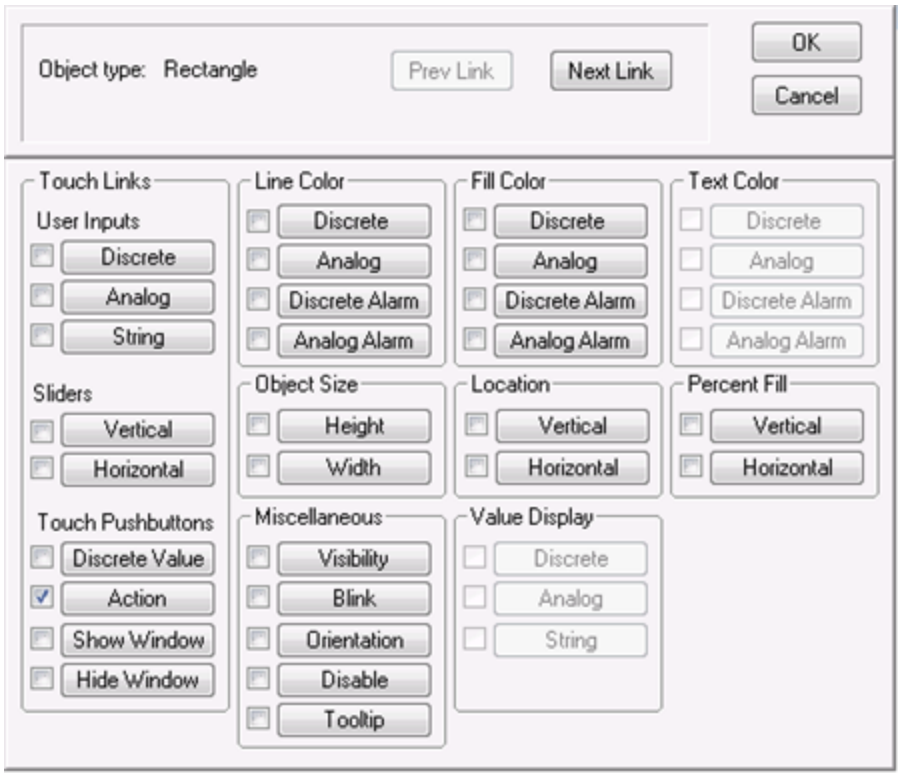
To configure a script that executes on this condition	Select
One time when the left mouse button or a certain key or key combination is pressed	**On Left Click/Key Down**
Periodically while the left mouse button or a certain key or key combination is pressed	**While Left/Key Down**
One time when the left mouse button or a certain key or key combination is released	**On Left/Key Up**
One time when the left mouse button is double-clicked	**On Left Double Click**
One time when the right mouse button is pressed	**On Right Click**
Periodically while the right mouse button is pressed	**While Right Down**

To configure a script that executes on this condition	Select
One time when the right mouse button is released	**On Right Up**
One time when the right mouse button is double-clicked	**On Right Double Click**
One time when the center mouse button is pressed	**On Center Click**
Periodically while the center mouse button is pressed	**While Center Down**
One time when the center mouse button is released	**On Center Up**
One time when the center mouse button is double-clicked	**On Center Double Click**
One time when the mouse moves over the object	**On Mouse Over**

4. If you select **On Left Click/Key Down**, **While Left/Key Down**, or **On Left/Key Up**:
   a. Select **Key**. The **Choose Key** dialog box appears.
   b. Select a key.
   c. Select the **Ctrl** and/or **Shift** checkboxes to assign a control key and/or shift key combination to your selected key.
5. If you select **While Left/Key Down** or **While Right Down**, type a time interval between 1 and 360000 milliseconds in the **Every** box.
6. If you select **On Mouse Over**, in the **After** box, type the number of milliseconds between 1 and 360000 to pass after the mouse has moved over the object before the script is executed.
7. Type your script in the window.
8. Select **OK**.

Delete all action scripts associated with an InTouch graphic object

1. Double-click the graphic object. The object properties panel appears.



2. Select to clear the **Action** checkbox. The action scripts will not be executed during run time. If you select the **Action** button, the editor opens with the last action script that you saved for any object.

Delete an individual action script

1. Double-click the graphic object that has the action script to delete. The object properties panel appears.
2. Select the **Action** button. The **Touch -> Action Script** dialog box appears.
3. In the **Condition Type** list, select the script trigger.
4. On the **Edit** menu, select **Clear**. The script from the main section clears and the associated script is deleted.

Configure ActiveX event scripts

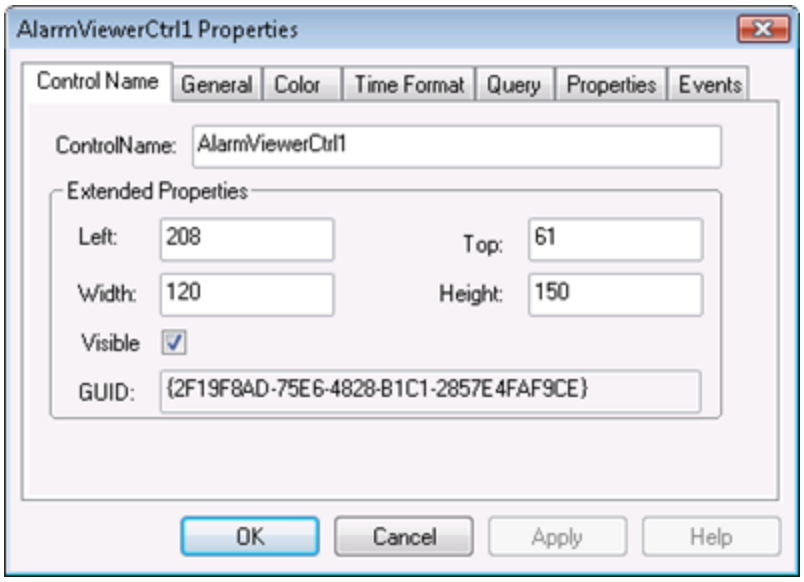
Use ActiveX event scripts to run a script when an ActiveX event occurs. Depending on the ActiveX control, such events can include:

- ActiveX control is started: Startup
- ActiveX control is closed: Shutdown
- User selects ActiveX control: Click
- User double-clicks on ActiveX control: Doubleclick

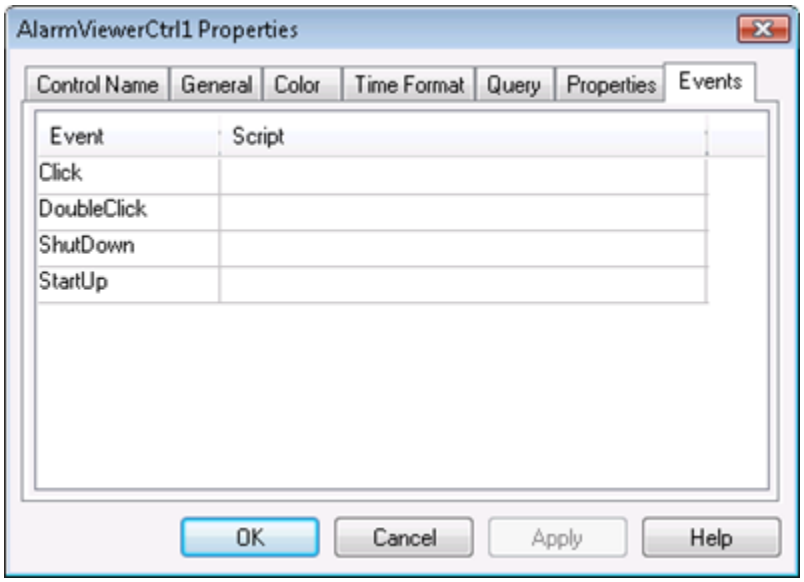
An ActiveX event script is identified by a name. By default, the InTouch HMI automatically adds the control name and event that the script is associated with. For example: MyActiveXScript (AlarmViewerCtrl1::Click).

Configure a new ActiveX event script

1. Double-click on the ActiveX control to configure. The ActiveX control properties dialog box appears.

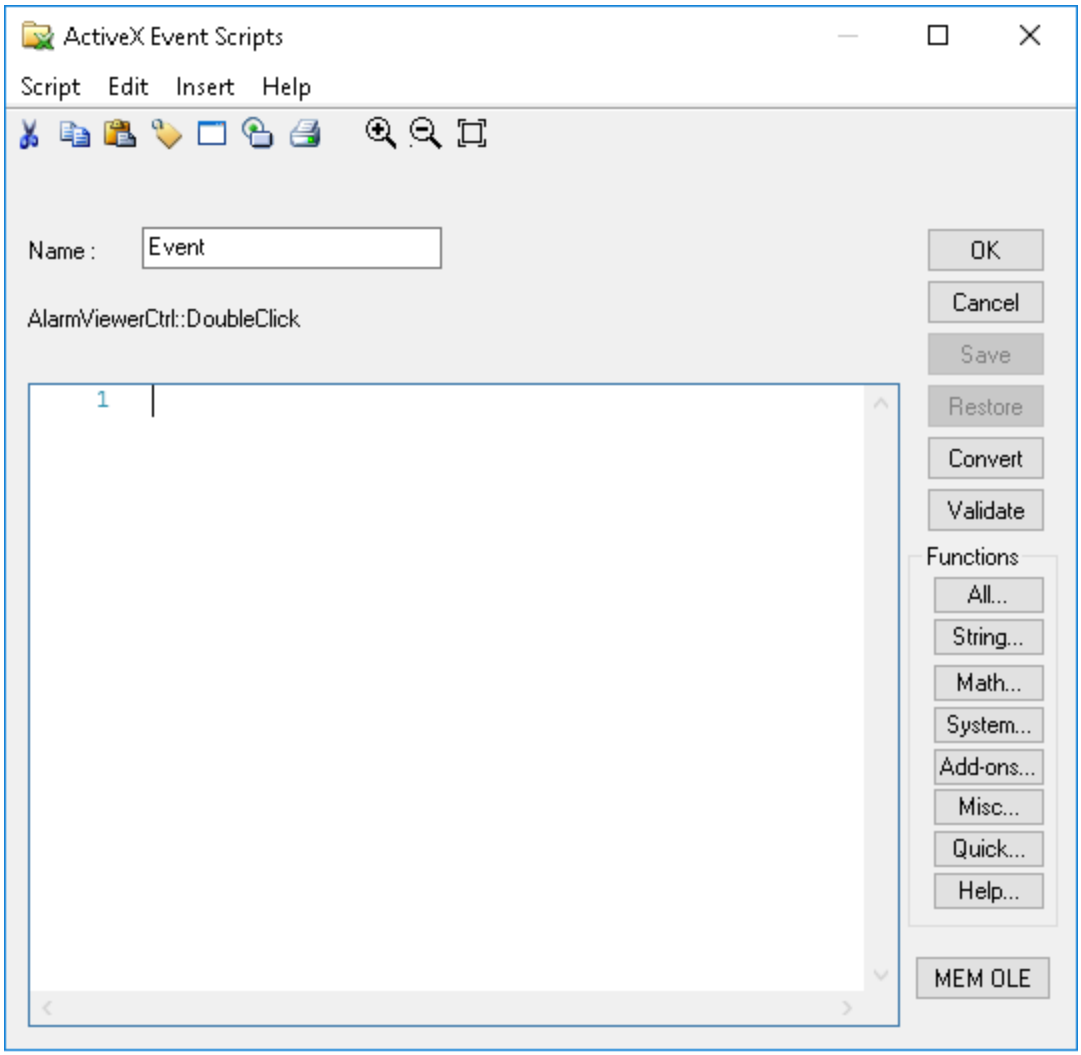


2. Select the **Events** tab.



3. Select an event such as click, double-click, shut down, or start up.
4. Select the **Scripts** cell for that event. Square brackets appear.
5. Type in a new name for an event script and select **OK**. When a message appears, select **OK** to create a new script.

   The **ActiveX Event Scripts** dialog box appears.



6. In the **Name** box, you can make changes to the ActiveX event script name.
7. Type your script in the window.
8. Select **OK**.

Edit an existing ActiveX event script

1. In the **Scripts** pane, expand **ActiveX Event**, right-click the ActiveX script name and select **Edit**. The **ActiveX Event Scripts** dialog box appears.
2. Make any necessary changes to the script and select **OK**.

Delete an existing ActiveX event script

1. Make sure that no ActiveX controls are using the ActiveX event script to delete. If there are ActiveX controls using the script, do the following first:
   a. Remove the ActiveX event script references in the **Events** panel of every ActiveX control that may be using it.
   b. Close all windows and update the use counts.

2. In the **Scripts** pane, expand **ActiveX Event**, right-click the ActiveX script name and select **Delete**. When a message appears, select **Yes**. The ActiveX event script is deleted.

Pause script execution at run time

By default, when WindowViewer is started, logic is running and synchronous scripts are executed. You can pause script execution at run time by halting logic. After pausing you can resume script execution.

Pause script execution at run time from the menu

- On the **Logic** menu, select **Halt Logic**. The synchronous scripts stop running. Asynchronous scripts continue running but no new asynchronous scripts are started.

Pause script execution at run time with scripting

- Write the value 0 to the discrete system tag $LogicRunning. The synchronous scripts stop running. Asynchronous scripts continue running but no new asynchronous scripts are started.

Resume script execution at run time

- On the **Logic** menu, select **Start Logic**. The script execution is resumed.

Resume script execution at run time with scripting

- Write the value 1 to the discrete system tag $LogicRunning. The $LogicRunning system tag must be contained in an asynchronous script that is executing at the time the logic is paused.

$LogicRunning System Tag

This system tag monitors and/or controls the running of scripts.

Usage

$LogicRunning

Remarks

Setting the value to 1 starts the script running. Setting the value to 0 stops the script running.

This system tag is equal to selecting **Start Logic** or **Halt Logic** on the **Logic** menu in WindowViewer.

You cannot stop asynchronous scripts that are currently running. However, you can prevent new scripts from running.

Data Type

Discrete (read / write)

The script language

Use these concepts, techniques, and syntax rules for writing scripts using the InTouch HMI script language.

- Basic syntax rules. See Basic syntax rules.
- Calling pre-defined or custom functions. See Call standard functions and Call custom functions (QuickFunctions).
- Using value assignments and the various operators. See Value assignments and operators.
- Using conditional statements. See Use conditional program branching structures.
- Using loops. See Use program loops.
- Using local variables. See Use local variables.

For more information on the general operation of the script editor, see Create and edit scripts.

For more information on the various types of script triggers, see Script triggers.

For a reference of standard script functions, see Built-In functions.

Basic syntax rules

Basic syntax rules cover these aspects of the InTouch HMI script language:

- Subroutines
- Statements
- Indentation
- Comments
- Tag references
- Literal data values
- Value expressions
- Syntax validation

Subroutines

There is no concept of separate subroutines within the same script, such as "Sub" procedures in Visual Basic. To structure a script into multiple subroutines, you must create a custom QuickFunction for each subroutine. See Custom script functions.

Statements

- A statement can be a value assignment, a function call, or a control structure.
- Each statement in a script must end with a semicolon (;).
- You can have multiple statements in the same line, as long as each statement ends with a semicolon.
- You can spread a statement across multiple lines by using line breaks (pressing Enter).

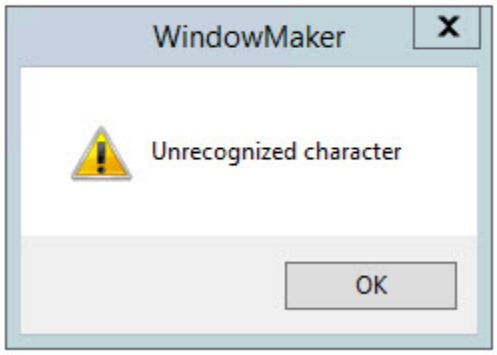
Indentation

You can indent your script code in any manner. Indents have no functional relevance.

Comments

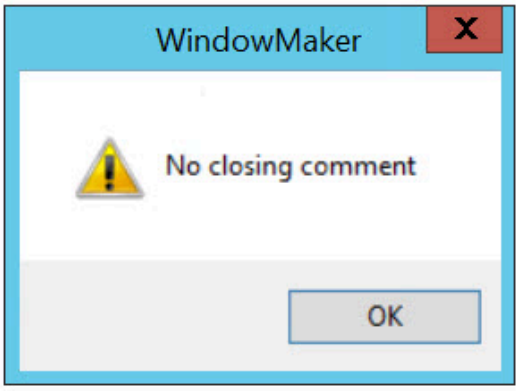
To mark text as a comment, enclose it in braces { }. Comments can span multiple lines. Nested comments are not supported. For example in the following example the first } brace closes the comment, and displays an error for the second } brace.

```
1  DIM msg AS Message;
2  {
3
4  Comments across multiple lines
5
6  {
7  Nested comments
8  }
9  }
10 msg = $DateString + ":" + $Operator;
```



If the } brace is not provided, an error is displayed during validation.

```
1  DIM msg AS Message;
2  {
3
4  Comments across multiple lines
5
6
7  Nested comments
8
9  msg = $DateString + ":" + $Operator;
```



Tag references

There are several ways to make tag references.

- To refer to a tag that is defined in the local Tagname Dictionary, simply use the tagname.
- To refer to a specific dot field, use the regular reference format (`Tagname.Dotfield`).

- To refer to a data item on a remote node, use a regular remote tag reference (`AccessName:Item`).
- You can also define local variables whose scope is limited to the current script. See Use local variables.

Literal data values

- You can specify integer values in decimal or hexadecimal notation. For example, `255` or `0xFF`.
- You can specify floating-point values in decimal or scientific notation. For example, `0.001` or `1E-3`.
- To specify a Boolean value, use the numerical values 0 for FALSE and 1 for TRUE.
- To specify a string value, enclose it in double quotation marks. For example: `"This is a string."`

Value expressions

Value expressions can include literal values, tag references and function calls, all linked together by suitable operators. See Value assignments and operators.

Syntax validation

When you save a script, the Script Editor automatically checks it for correct syntax. You can also start this validation manually by selecting the **Validate** button. See Validate scripts for correct syntax.

Call standard functions

Standard functions come predefined with the InTouch HMI. See Call custom functions (QuickFunctions).

Syntax for calling standard functions

The syntax to call a predefined script function depends on whether and how the function returns a result.

Some functions do not return any result; some functions return an optional result that can be assigned to a tag or used in an expression; some functions return a result that must be assigned to a tag or used in an expression.

To determine the function type, look at the function description. Each function description has a syntax listing that shows whether the function returns a result and whether that result is optional.

Call a function that does not return a result

- Use only the function name (and parameters, if any) in a statement. For example:

```
FunctionName(Parameters);
```

Call a function that requires its result to be assigned

- Use the function name (and parameters, if any) anywhere in a script where you could use a literal value or a tagname of the relevant data type. For example, in a value assignment:

```
ResultsTagname = FunctionName(Parameters);
```

Or in a nested function call, using it as a parameter for another standard function:

```
OtherStandardFunction(FunctionName(Parameters));
```

Call a function that returns an optional result

- Use either of the preceding procedures.

Pass parameters to a function

Parameters to standard predefined functions are usually passed by *value*. This means that you can pass any valid expression as a parameter, as long as the expression evaluates to the data type that is required for the parameter. Such expressions can include literal values, tag references, and function calls, all linked together by suitable operators. For more information on expressions and operators, see Value assignments and operators.

When the script calls the function, the expression is evaluated and the resulting value passed to the function.

However, there are some functions that require a tag *reference* as a parameter. For example:

```
RecipeSelectRecipe(Filename, RecipeName, Number);
```

In this example, the RecipeName parameter must be a tag reference (that is, you must use a literal tagname for the RecipeName parameter). You cannot pass a string expression instead, even if that expression evaluates to a valid tagname.

Note: Some legacy predefined functions with only one parameter (for example, the Ack() function) do not follow the standard syntax of passing parameters in parentheses. Instead, the parameter is separated from the function name by a space. Check the syntax description in the function documentation if you are in doubt about a particular function.

Call custom functions (QuickFunctions)

Calling a custom QuickFunction differs slightly from calling a predefined standard function:

- The keyword `CALL` must precede the QuickFunction name.
- Results returned by QuickFunctions are always optional; you can use them, but you do not have to.

Call a QuickFunction that does not return a result

- Use the function name (and parameters, if any) preceded by the keyword `CALL` in a statement. For example:

```
CALL QuickFunctionName(Parameters);
```

Call a QuickFunction that returns a result

- Do either of the following:
  - Call the QuickFunction as if it did not return a result (see the preceding procedure).
  - Use the function name (and parameters, if any) preceded by the keyword `CALL` anywhere in a script where you could use a literal value or a tagname of the relevant data type. For example, in a value assignment:

    ```
    ResultsTagname = CALL QuickFunctionName(Parameters);
    ```

    Or in a nested function call, using it as a parameter for a standard function:

    ```
    OtherStandardFunction(CALL FunctionName(Parameters));
    ```

Note: You cannot nest QuickFunction calls so that a QuickFunction is used as a parameter for another

QuickFunction. For example, `Call QF1(Call QF2());` is not a valid statement.

Pass parameters to a QuickFunction

Parameters to QuickFunctions are always passed by value. You cannot pass parameters to QuickFunctions by reference.

You can pass any valid expression as a parameter, as long as the expression evaluates to the data type that is required for the parameter. Such expressions can include literal values, tag references, and function calls, all linked together by suitable operators. For more information on expressions and operators, see Value assignments and operators. When the script calls the function, the expression is evaluated and the resulting value passed to the function.

Note: You cannot nest QuickFunction calls so that a QuickFunction is used as a parameter for another QuickFunction. For example, `CALL QF1(CALL QF2());` is not a valid statement.

Value assignments and operators

In a script, you use value assignments to write values to a tag. The syntax for a value assignment is as follows:

```
Tagname = ValueExpression;
```

When this statement is executed, `ValueExpression` is written to the tag referred to by `Tagname`. `ValueExpression` can be any valid expression whose data type matches the tag data type. Value expressions can include literal values, tag references, and function calls, all linked together by suitable operators.

See Supported operators.

See Set the evaluation order of operators.

See Examples of valid and invalid expressions.

Supported operators

The following table lists all supported operators. For information on the use of a specific operator, see the relevant section.

Operator	More information
+	Addition or concatenation: +
-	Subtraction: -
*	Multiplication: *
/	Division: /
**	Power: **
MOD	Modulo: MOD
~	Complement: ~
SHL	Shift Left: SHL and Shift Right: SHR

Operator	More information
SHR	Shift Left: SHL and Shift Right: SHR
&	Bitwise AND: &
|	Bitwise OR: |
^	Bitwise XOR: ^
AND	Logical conjunction: AND
OR	Logical disjunction: OR
NOT	Logical negation: NOT
<	Comparisons: <, >, <=, >=, ==, <>
>	Comparisons: <, >, <=, >=, ==, <>
<=	Comparisons: <, >, <=, >=, ==, <>
>=	Comparisons: <, >, <=, >=, ==, <>
==	Comparisons: <, >, <=, >=, ==, <>
<>	Comparisons: <, >, <=, >=, ==, <>

Note: For numeric calculations, always select the operands so that the result of the calculation is still within the value range of a Real number. Otherwise, the result will not be correct.

Addition or concatenation: +

Adds two numeric operands or concatenates two string operands.

Valid operands

For addition: Any Integer or Real value

For concatenation: Any Message value

Data type of return value

For addition: Integer or Real

For concatenation: Message

Example

```
MessageTag = "Setpoint value: " + Text(SetpointTag, "#.##");
```

Subtraction: -

When used with two operands, performs a regular numeric subtraction.

Valid operands

Any Integer or Real value

Data type of return value

Integer or Real

Example

In this example, if TemperatureSetpoint is 70, after the script executes TemperatureSetpoint is 65.

```
TemperatureSetpoint = TemperatureSetpoint – 5;
```

Multiplication: *

Regular numeric multiplication.

Valid operands

Any Integer or Real value

Data type of return value

Integer or Real

Division: /

Regular numeric division. If you try to divide by 0 at run time, 0 is returned as the result.

Valid operands

Any Integer or Real value

Data type of return value

Integer or Real

Power: **

Raises the left operand (the base) to the power of the right operand (the power).

Valid operands

Integer or Real values. It is not possible to combine a base of 0 with a negative power, or a negative base with a fractional power. In these cases, 0 is returned as the result.

Data type of return value

Integer or Real

Example

`8 ** (1/3)` returns 2 (the cubic root of 8)

Modulo: MOD

Returns the remainder of the division of two integer values.

Valid operands

Any Integer value.

Data type of return value

Integer

Example

`37 MOD 4` returns 1

Complement: ~

Returns the one's complement of an integer value. That is, converts each zero-bit to a one-bit and vice versa.

Valid operands

Any Integer value.

Data type of return value

Integer

Shift Left: SHL and Shift Right: SHR

Shifts the binary representation of an integer value to the right or left by a specified number of bit positions. The left operand is the value to be shifted, the right operand is the number of bit positions. Bits shifted out of the

word are lost. Bit positions vacated by the shift are set to 0.

Valid operands

Any Integer value.

Data type of return value

Integer

Example

`IntTag = IntTag SHL 1;` has the following results when executed repeatedly for an initial tag value of 5:

Iteration	Binary pattern	Tag value
Initial value	`0[...]00000101`	5
Execution 1	`0[...]00001010`	10
Execution 2	`0[...]00010100`	20

Bitwise AND: &

Compares the binary representations of two integer numbers, bit for bit, and returns a result according to the following table:

Bit in first operand	Bit in second operand	Bit in result
0	0	0
0	1	0
1	0	0
1	1	1

You can use this operator to quickly "mask out" (set to 0) certain parts of a bit pattern. For example, the following statement masks out the upper 24 bits of the `IntTag` tag:

```
IntTag = IntTag & 255;
```

As shown in the table, the result bit is always 0 if one of the operand bits is 0. In the binary representation of 255, only the lower 8 bits are 1, so the 24 remaining 0-bits cause all the corresponding bits in the result to be set to 0.

Valid operands

Any Integer value.

Data type of return value

Integer

Bitwise OR: |

Compares the binary representations of two integer numbers, bit for bit, and returns a result according to the following table:

Bit in first operand	Bit in second operand	Bit in result
0	0	0
0	1	1
1	0	1
1	1	1

This operation is also called "inclusive OR."

Valid operands

Any Integer value.

Data type of return value

Integer

Bitwise XOR: ^

Compares the binary representations of two integer numbers, bit for bit, and returns a result according to the following table:

Bit in first operand	Bit in second operand	Bit in result
0	0	0
0	1	1
1	0	1
1	1	0

This operation is also called "exclusive OR."

Valid operands

Any Integer value.

Data type of return value

Integer

Logical conjunction: AND

Returns TRUE if both discrete operands are TRUE; otherwise, returns FALSE. The truth table for this operator is as follows:

p	q	p AND q
F	F	F
F	T	F
T	F	F
T	T	T

Valid operands

Any Discrete value.

Data type of return value

Discrete

Logical disjunction: OR

Returns TRUE if at least one of the discrete operands is TRUE; otherwise, returns FALSE. The truth table for this operator is as follows:

p	q	p OR q
F	F	F
F	T	T
T	F	T
T	T	T

Valid operands

Any Discrete value.

Data type of return value

Discrete

Logical negation: NOT

Returns TRUE if the discrete operand is FALSE, and vice versa. The truth table for this operator is as follows:

p	NOT p
F	T
T	F

Valid operands

Any Discrete value.

Data type of return value

Discrete

Comparisons: <, >, <=, >=, ==, <>

These operators compare two values and return TRUE if the condition specified by the operator is met. The operands can be of any data type. For string operands, the comparison is based on alphabetical, non-case-sensitive ordering, with b being greater than a, c greater than b, and so on. For discrete operands, TRUE is considered greater than FALSE. The following table lists all comparison operators along with their conditions:

Operation	Example	Condition
Less than	a < b	a is less than b
Greater than	a > b	a is greater than b
Less than or equal	a <= b	a is less than or equal to b
Greater than or equal	a >= b	a is greater than or equal to b
Equal	a == b	a is equal to b
Not equal	a <> b	a is not equal to b

Valid operands

Values of any data type (both values must be of the same data type).

Data type of return value

Discrete

Set the evaluation order of operators

In any expression, you can use parentheses to force operators to be evaluated in a certain order. This works the same way as in any mathematical expression. If you do not use parentheses, your expression is evaluated based on the default precedence rules for operators. The operation with the highest precedence level is executed first, followed by the operation with the second-highest precedence level, and so on.

The following table shows the precedence level of each operator. Operators on the same row have the same precedence level.

-, NOT, ~	Highest precedence
**	
*, /, MOD	
+, -	
SHL, SHR	
<, >, <=, >=	
==, <>	
&	
^	
|	
AND	
OR	
=	Lowest precedence

Implicit data type conversion

The InTouch HMI scripting language provides implicit value conversion in assignments between certain data types. However, this can lead to unexpected results, so you should only use this feature with caution.

The following table shows what happens when you assign a value of a certain type to a tag of a different type.

Expected data type	Used data type	Remarks
Discrete	Integer	A value of 0 is interpreted as FALSE. Any other value is interpreted as TRUE.
Discrete	Real	A value of 0 is interpreted as FALSE. Any other value is interpreted as TRUE.
Integer	Discrete	A value of FALSE is converted to 0. A value of TRUE is converted to 1.
Integer	Real	Only the value before the decimal separator is used. All decimal places are discarded.
Real	Discrete	A value of FALSE is converted to 0. A value of TRUE is converted to 1.
Real	Integer	The value is preserved without changes.

For information on using script functions to convert between other data types, see Convert data types.

Examples of valid and invalid expressions

The following table shows some valid expressions, along with the expression's result and the result's data type.

Expression	Data type of result	Result
`37 MOD 4`	Integer	1
`37 MOD 4 == 1`	Discrete	TRUE
`NOT (37 MOD 4 == 1)`	Discrete	FALSE
`InfoAppActive(InfoAppTitle("xyz")) == 1`	Discrete	TRUE if a process called "xyz" is running
`"Batch " + Text(IntTag, "000")`	Message	"Batch 010" if `IntTag` has a value of 10

The following table shows some invalid expressions, along with the reason why they are invalid.

Expression	Problem
`NOT (37 MOD 4)`	NOT requires a discrete operand.
`NOT 37 MOD 4 == 1`	NOT has a higher precedence than the other operators, so the InTouch HMI tries to apply NOT to the integer value of 37 instead of the discrete result of the comparison.
`"Batch " + IntTag`	When using the + operator to concatenate strings, both operands must be strings.

Use conditional program branching structures

You can dynamically control the execution path of a script based on certain conditions being met. The InTouch HMI supports `IF-THEN-ELSE` control structures for this purpose.

The basic syntax for an `IF-THEN-ELSE` control structure is as follows:

Syntax

```
IF Condition THEN
   ... statements and/or another IF-THEN-ELSE structure
[ELSE
   ... statements and/or another IF-THEN-ELSE structure]
ENDIF;
```

Remember the following rules when working with `IF-THEN-ELSE` structures:

- `IF-THEN-ELSE` structures can be nested, both in the `THEN` section and in the `ELSE` section.
- For every `IF` statement, there must be a closing `ENDIF` statement. An `ENDIF` statement always applies to the nearest prior `IF` statement on the same nesting level.
- `Condition` must be a valid discrete expression. The `THEN` section is executed if `Condition` is TRUE. The `ELSE` section is executed if `Condition` is FALSE.
- The `ELSE` section is optional.
- Some other programming languages allow you to check multiple conditions on the same hierarchy level of an `IF-THEN-ELSE` structure and have one general `ELSE` section that is executed if all of the conditions evaluate to FALSE. (The `If-ElseIf-Else` structure in Visual Basic is an example of this.) This is not possible in the InTouch HMI. For every condition to check, you must open a new `IF-THEN-ELSE` structure. Therefore, to have a single section of code to act as the `ELSE` code for all conditions, you must place it in the `ELSE` section of the `IF-THEN-ELSE` structure at the last nesting level.

Simple conditional structure

The following script shows a simple conditional structure. If `SuccessTag` is TRUE, the "Success" window opens, otherwise the "Failure" window opens.

```
IF SuccessTag == 1 THEN
   Show "Success";
```

```
ELSE
   Show "Failure";
ENDIF;
```

Nested conditional structure

The following script shows how to check for multiple conditions and have one general ELSE section with code that is executed if none of the conditions are met.

```
IF ChoiceTag == 1 THEN
   Show "Procedure 1";
ELSE
   IF ChoiceTag == 2 THEN
      Show "Procedure 2";
   ELSE
      IF ChoiceTag == 3 THEN
         Show "Procedure 3";
      ELSE
         Show "Default Procedure";
      ENDIF;
   ENDIF;
ENDIF;
```

Invalid scripting example (missing ENDIF)

If you are familiar with Visual Basic, you might try to write a simple IF statement like this:

```
IF OpenThisWindow == 1 THEN Show "This Window";
```

This does not work in the InTouch HMI. For every IF statement, there must be a closing ENDIF statement.

Invalid scripting example (incorrect nesting)

If you are familiar with a language like Visual Basic, you might want to write a conditional structure with multiple conditions and a default condition like this:

```
IF ChoiceTag == 1 THEN
   Show "Procedure 1";
ELSE IF ChoiceTag == 2 THEN
   Show "Procedure 2";
ELSE IF ChoiceTag == 3 THEN
   Show "Procedure 3";
ELSE
   Show "Default Procedure";
ENDIF;
```

This does not work in the InTouch HMI. Each IF opens a new nesting level and must have a corresponding ENDIF statement. For a correct version of this example, see Nested conditional structure.

Use program loops

Loops allow you to execute a section of code repeatedly. The InTouch HMI only supports FOR loops. A FOR loop works by monitoring the value of a numeric loop variable that is incremented or decremented with each loop iteration. The loop is executed until the value of the loop variable reaches a fixed limit.

Syntax

```
FOR LoopTag = StartExpression TO EndExpression [STEP ChangeExpression]
... statements or another FOR loop ...
NEXT;
```

- `StartExpression`, `EndExpression` and `ChangeExpression` together define the number of iterations.
- `StartExpression` sets the start value of the loop range. `EndExpression` sets the end value of the loop range.
- `STEP ChangeExpression` optionally sets the value by which the loop tag is incremented or decremented during each loop iteration; if you do not specify this, a default of 1 is used.

When you execute a `FOR` loop, the InTouch HMI:

1. Sets `LoopTag` to the value of `StartExpression`.
2. Tests whether `LoopTag` is greater than `EndExpression`. If so, the InTouch HMI exits the loop. (If `ChangeExpression` is negative, the InTouch HMI tests whether `LoopTag` is less than `EndExpression`.)
3. Executes the statements within the loop.
4. Increments `LoopTag` by the value of `ChangeExpression` (1 unless otherwise specified).
5. Repeats steps 2 through 4.

Remember the following rules when working with `FOR` loops:

- `FOR` loops can be nested. The maximum number of nesting levels depends on the available memory and system resources.
- For every `FOR` statement, there must be a closing `NEXT` statement. A `NEXT` statement always applies to the nearest prior `FOR` statement on the same nesting level.
- `LoopTag` must be a numeric tag (or local variable).
- `StartExpression`, `EndExpression` and `ChangeExpression` must be valid expressions that evaluate to a numeric result.
- If `ChangeExpression` is positive, `EndExpression` must be greater than `StartExpression`; if `ChangeExpression` is negative, `StartExpression` must be greater than `EndExpression`. Otherwise, the loop does not start.
- To exit a loop, use the `EXIT FOR` statement. For more information, see Force the end of a loop.
- There is a time limit for loops. See Time limit for loop execution.

Caution: Loop execution affects other run-time processes. For more information, see Effect of loops on other run-time processes.

Force the end of a loop

You can exit a loop at any time by calling the following statement:

```
EXIT FOR;
```

This statement causes script execution to continue at the statement immediately following the loop `NEXT` statement.

Example

The following code fragment uses a loop to insert a large number of dummy records into a database table. If there is an error inserting a record, the loop is aborted to prevent creating more errors.

```
FOR Counter = 1 TO 1000
   ResultCode = SQLInsert(ConnectionID, "BatchDetails", "BindList1");
   IF ResultCode <> 0 THEN
      LogMessage("Error creating records! Aborting...");
      EXIT FOR;
   ENDIF;
NEXT;
```

Effect of loops on other run-time processes

While a FOR loop is executing, all other run-time processes in WindowViewer are paused. This includes the following areas:

- Screen updates (animation links, value displays, trends, etc.). This means that you cannot use FOR loops to animate objects, because no movement will occur until after the loop has completed.
- I/O communications. For example, if you modify the value of an I/O tag in a FOR loop, only the value after the final iteration is written to the I/O device.
- Other scripts, including asynchronous QuickFunctions.

You can avoid pausing other run-time processes by placing the FOR loop in an asynchronous QuickFunction.

Time limit for loop execution

To avoid infinite loops, there is a time limit during which FOR loops must complete execution. If a loop does not complete execution after this time span, WindowViewer automatically terminates it and writes a message about the termination to the Log Viewer.

The default time limit is 5 seconds. You can customize it by adding the following line to the intouch.ini file in your application directory:

```
LoopTimeout=x
```

Replace x with the time limit in seconds.

Note: The time limit is checked only at the NEXT statement of the loop. Therefore, the first iteration of the loop is always executed, even if it takes longer than the time limit.

Examples of loops

The following script uses a simple loop and an indirect tag to re initialize 100 tags (Tag001 to Tag100) with a value of 0.

```
DIM Counter AS INTEGER;
FOR Counter = 1 TO 100
   IndirectInteger.Name = "Tag" + Text(Counter, "000");
   IndirectInteger.Value = 0;
NEXT;
```

The following script uses two nested loops and an indirect tag to reinitialize 1000 tags (Line01_Tag001 to

Line10_Tag100) with a value of 0.

```
DIM LineCounter AS INTEGER;
DIM TagCounter AS INTEGER;
FOR LineCounter = 1 TO 10
   FOR TagCounter = 1 TO 100
      IndirectInteger.Name = "Line" + Text(LineCounter, "00") + "_Tag" + Text(TagCounter,
      "000");
      IndirectInteger.Value = 0;
   NEXT;
NEXT;
```

Use local variables

You can declare local variables in a script to store temporary or intermediate results. This increases performance and helps to keep your tag count low. You can use local variables just like tagnames in your script. However, there are certain differences:

- Local variables only exist within the scope of the script in which they are declared. They lose their value when script execution finishes. They cannot be referenced by any other scripts in your application.
- Local variables do not have dotfields.
- Local variables do not count towards the tag count.

Before you can use a local variable in a script, you must declare it; otherwise, the reference is considered a tagname. See Declare a local variable.

You can declare local variables that have the same names as tags. See Naming conflicts between local variables and tags.

Declare a local variable

You can declare local variables anywhere in your script, as long as you declare them before their first use. To declare a local variable, use the following statement:

```
DIM LocVarName [AS DataType];
```

`LocVarName` is the name of the local variable. The name must follow the naming conventions for tagnames. For more information, see Tag name conventions.

`DataType` is the data type of the local variable. Valid values are `Discrete`, `Integer`, `Real`, and `Message`. If you do not specify this option, `Integer` is used as the default.

You must use a separate `DIM` statement for each local variable to declare.

You can declare any number of local variables. The number is only limited by the available memory.

Examples

To declare an Integer variable:

```
DIM MyLocalIntVar AS Integer;
```

To declare multiple Real variables:

```
DIM MyLocalRealVar1 AS Real;
DIM MyLocalRealVar2 AS Real;
```

The following statement is *not* valid:

```
DIM MyLocalRealVar1, MyLocalRealVar2 AS Real;
```

Naming conflicts between local variables and tags

You can declare a local variable with the same name as an existing tag. However, when you refer to that name in a script, the local variable always takes precedence over the tag. For example, assume you have an existing Integer tag called "iTag," and you run the following script:

```
DIM iTag as Integer;
iTag = 20;
```

In this scenario, the value assignment writes a value to the local variable only. The value of the tag with the same name remains unchanged.

Custom script functions

InTouch HMI QuickFunctions are scripts that in other environments might be known as macros, subroutines, or procedures.

About QuickFunctions

QuickFunctions are scripts that you can call from other scripts and animation links. The main advantage of QuickFunctions is a reduction in duplicate code.

You can pass values to QuickFunctions, which can use the values and return results.

QuickFunctions can run asynchronously. Unlike other scripts, they can run in the background without disrupting the main program flow. A QuickFunction running asynchronously can be used for time-consuming operations, such as SQL database calls.

Note: Plan QuickFunctions and their arguments carefully, because if you want to modify the arguments in a QuickFunction, you must first delete all calls to that QuickFunction from every script that uses the QuickFunction. After the change is made, you must then add the QuickFunction call back to the scripts. See the note in Configure QuickFunctions.

There are three basic parts of a QuickFunction:

- Name
- Arguments (optional)
- Script body with optional return values

QuickFunctions are executed by using the CALL function in either an animation link or another script. See Call QuickFunctions.

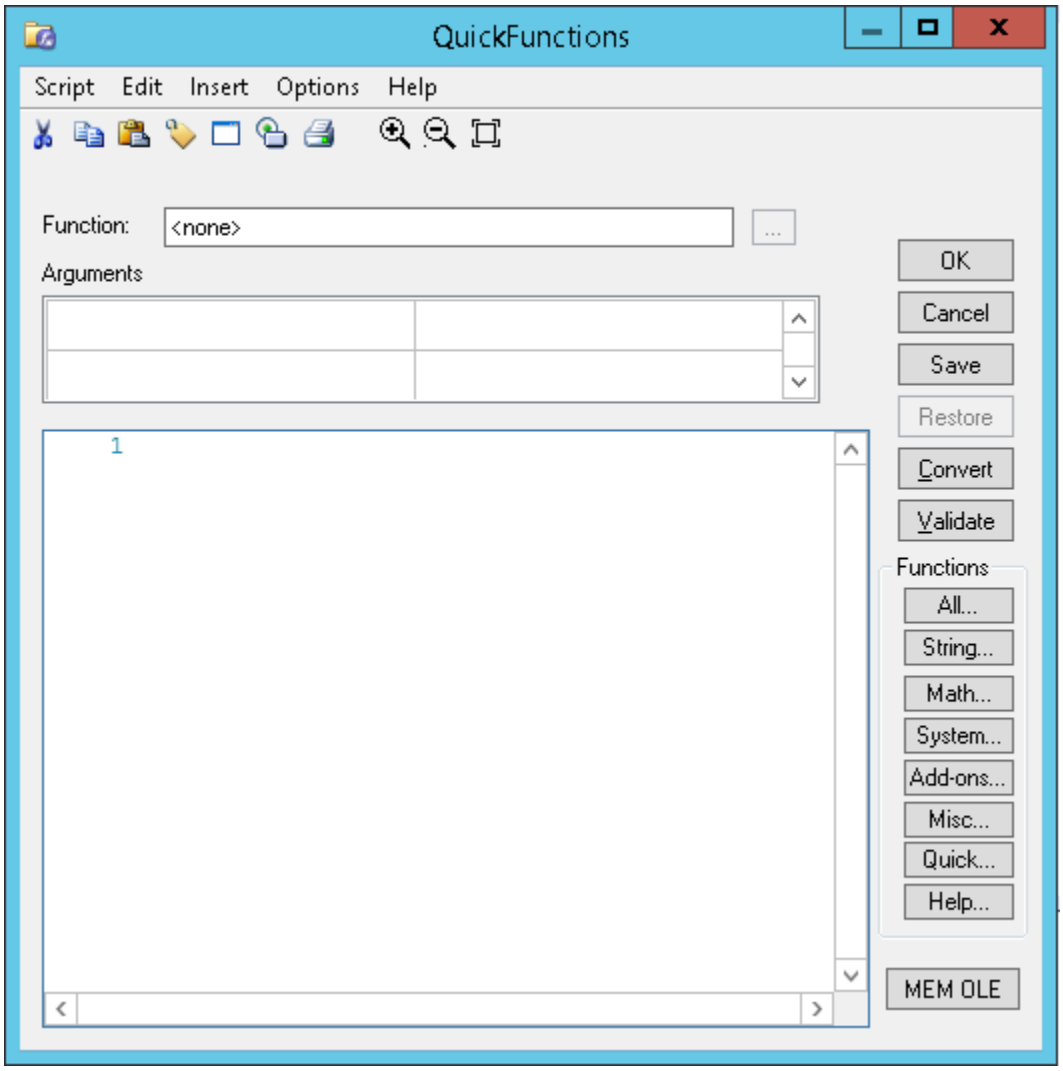
Configure QuickFunctions

You can create, modify, or delete QuickFunctions.

Create a QuickFunction

1. In the **Scripts** pane, right-click **QuickFunctions,** and then select **New**.

   The **QuickFunctions** dialog box appears.



2. In the **Function** box, enter a name for the QuickFunction.
3. In the **Arguments** area, for each argument, enter a name on the left and a data type on the right.

   Arguments are local variables that exist only within the QuickFunction in which they are defined. You can have up to 16 arguments per QuickFunction. Argument names can have 31 characters but no spaces. The argument names must begin with an alpha character. Argument names must be unique.
4. Type your script in the window.
5. To cause the QuickFunction to return a result, add to your script: RETURN value

   Value can be a literal value, a local variable, or global tagname or calculated expression. The script terminates at the RETURN command and continues at the calling function.
6. Select **OK**.

Modify a QuickFunction

1. In the **Scripts** pane, expand **QuickFunctions**, right-click the QuickFunction to modify and select **Open**. The **QuickFunctions** dialog box appears.
2. Make modifications to the script body and select **OK**.

Note: You cannot make modifications to the argument list if there are calls to the QuickFunction in the InTouch application. You must delete those calls first, close all InTouch windows, and update the use counts.

Delete a QuickFunction

1. Delete all calls to the QuickFunction, close all InTouch windows, and update the use counts.
2. In the **Scripts** pane, expand **QuickFunctions**, right-click the QuickFunction to delete and select **Delete**. When a message appears, select **Yes**.

Call QuickFunctions

You can configure scripts and animation links to call QuickFunctions and to process or show a possible return value.

A QuickFunction is not called if the parameter values have not changed. You can use $second as a parameter to insure a QuickFunction is executed at least every second.

For more information, see Call custom functions (QuickFunctions).

Create asynchronous QuickFunctions

You can define QuickFunctions to run asynchronously (that is, parallel) to the main program flow.

Create an asynchronous QuickFunction

1. In the Script Editor, create a QuickFunction.
2. On the **Options** menu, select **Asynchronous**.

Limitations of Asynchronous QuickFunctions

You cannot:

- Return a value from an asynchronous QuickFunction.
- Run more than one instance of the same QuickFunction at the same time.
- Stop asynchronous QuickFunctions after they start executing.

You should not:

- Run more than three different asynchronous QuickFunctions at the same time. Running more than three QuickFunctions at the same time reduces system performance significantly.
- Use asynchronous functions as part of expressions for animation links, e.g. Tool Tips.

Check if any asynchronous QuickFunctions are running

You can check if any asynchronous QuickFunctions are running with the IsAnyAsyncFunctionBusy() function. You can use this function to make the QuickScript that calls an asynchronous QuickFunction wait for all other asynchronous QuickFunctions to complete processing.

IsAnyAsyncFunctionBusy() Function

Returns a discrete value indicating if any asynchronous QuickFunctions are running.

Syntax

```
result = IsAnyAsyncFunctionBusy(timeout)
```

Arguments

result

The discrete value that indicates if asynchronous QuickFunctions are running with following meaning:

- 0 = No asynchronous QuickFunctions are running.
- 1 = Asynchronous QuickFunctions are running.

timeout

The number of seconds to wait before checking if any asynchronous QuickFunctions are running. A literal integer value, integer tagname or integer expression.

Example(s)

Assume you want to connect to several SQL databases using asynchronous QuickFunctions, and you know that it takes 2 minutes to make those connections.

First, execute the asynchronous QuickFunctions to connect to the SQL databases.

Next, use the IsAnyAsyncFunctionBusy(120) function in a QuickScript to allow enough time for SQL to make the connections before completing the QuickFunction.

If after 2 minutes the connections have not been made and the asynchronous QuickFunctions are still busy trying to make the connections, a value of 1 (true) is returned by the IsAnyAsyncFunctionBusy() function.

You can now show an error message telling the operator that the SQL connections were unsuccessful.

The following script implements the scenario:

```
IF IsAnyAsyncFunctionBusy(120) == 1 THEN
   SHOW "SQL Connection Error Dialog";
ENDIF;
```

Stop asynchronous QuickFunctions from running

You cannot stop asynchronous QuickFunctions after they are started, but you can stop further asynchronous QuickFunctions from being started by stopping the script logic. This affects all QuickScripts in your InTouch

application.

For more information on stopping script execution, see Pause script execution at run time.

Built-In functions

InTouch QuickScript functions allow you to execute commands and logical operations based on specified criteria being met. You can use QuickScript functions by themselves and have them executed whenever a certain condition is met, or use them in animation display links. Predefined functions are organized by functional groups. After you select a group, then you select the predefined function to insert in your script. You use the **Choose function** dialog box that appears after selecting the predefined math functions group. The predefined function is placed in your script at the current position of your pointer when you select the function.

Important: This chapter includes legacy InTouch QuickScript functions designed to work only on 32-bit versions of the Windows operating system. These functions should not be included in any InTouch QuickScript designed to run on a 64-bit version of Windows. Notes within this chapter identify these legacy 32-bit only functions.

Force updates in animation display links

If you use QuickScripts in animation links, the animation links are only updated if a tag is associated with them. This tag acts as a trigger whenever its value changes. A good choice is to use the $Second or $Minute system tag to update animation links.

Force an update in an animation display link

1. Open the animation link in the object property window.
2. Add a trigger tag (for example $Second) to the calculation. For example:
   - If the animation link is real or integer, you can multiply the expression with $Second/$Second.
   - If the animation link is string, you can add StringMid($TimeString, 0, 0) to the expression.
   - If the animation link is discrete you can add ($second.00 - $second.00) to the expression.

Mathematical calculations

The InTouch HMI supports basic mathematical functions that you can use in scripts and in animation links, such as functions to:

- Round and truncate numbers.
- Calculate sine and cosine.
- Calculate logarithms and exponentials.
- Calculate the square root.

Round, truncate, and determine sign

In a script, you can use the following functions to round numbers, truncate numbers, and determine the sign of numbers:

Use	To
Abs()	Calculate the absolute of a value or expression.
Int()	Calculate the integer of a value or expression.
Round()	Round a value or expression.
Sgn()	Determine the sign (minus, plus, zero) of a value or expression.
Trunc()	Return the part of a value or expression before the decimal point.

Abs() Function

Returns the absolute value of a specified number. You can use this to convert a negative number to a positive number.

Syntax

```
result = Abs (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

Example(s)

`Abs(14)` returns 14.

`Abs(-7.5)` returns 7.5.

Int() Function

Returns the integer less than (or equal to) a specified number.

Syntax

```
result = Int (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

Example(s)

`Int(4.7)` returns 4.

`Int(-4.7)` returns -5.

Note: For negative real numbers, this function returns an integer that is smaller than the specified number. For example, Int(-4.7) is not -4, but -5. To have the integer part returned, use the Trunc() function. See Trunc() Function.

Round() Function

Rounds a number to a specified precision. The result is a real number.

Syntax

```
result = Round (number, precision)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

precision

The precision to which the number is rounded. Can be a literal number, analog tagname, or numeric expression.

Example(s)

`Round(4.3, 1)` returns 4.

`Round(4.3, 0.01)` returns 4.30.

`Round(4.5, 1)` returns 5.

`Round(-4.5, 1)` returns -4.

`Round(106, 5)` returns 105.

`Round(43.7, 0.5)` returns 43.5.

Sgn() Function

Returns the sign of a number. Use it to determine if a number, tagname, or expression is negative, positive, or zero.

Syntax

```
result = Sgn (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

Example(s)

`Sgn(425)` returns 1.

`Sgn(0)` returns 0.

`Sgn(-37.3)` returns -1.

Trunc() Function

Returns the truncated value of a number. The truncated value is the part before a decimal point. Use it to work with the integer part of a real number.

Syntax

```
result = Trunc (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

Example(s)

`Trunc(4.3)` returns 4.

`Trunc(-4.3)` returns -4.

Note: You can also use this function to work with the fractional part of a number. To return the fractional part of a specified number use the Trunc() function as follows:
result = number - trunc(number);

Use trigonometric functions

In a script, you can use the following functions to do trigonometric calculations.

Use	To
Sin()	Calculate the sine of an angle.
ArcSin()	Calculate the arcus sine of a value or expression.
Cos()	Calculate the cosine of an angle.

Use	To
ArcCos()	Calculate the arcus cosine of a value or expression.
Tan()	Calculate the tangent of an angle.
ArcTan()	Calculate the arcus tangent of a value or expression.

Note: Trigonometric QuickScript functions in the InTouch HMI use angles in degrees (0 - 360). To work with radians instead you must perform the corresponding calculation before passing the parameter to the function or after retrieving the result from the function.

Sin() Function

Returns the sine of a number. For trigonometric functions the number is the angle in degrees.

Syntax

```
result = Sin (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

Example(s)

`Sin(90)` returns 1.

`Sin(0)` returns 0.

`Sin(30)` returns 0.5.

`100 * Sin (6 * $second)` returns a sine wave with an amplitude of 100 and a period of one minute.

ArcSin() Function

Returns the arc sine of a number. It is the reciprocal function to the Sin() function. Use the ArcSin() function to calculate the angle from -90 to 90 degrees whose sine is equal to that number.

Syntax

```
result = ArcSin (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression in the range of -1 to 1.

Example(s)

`ArcSin(1)` returns 90.

`ArcSin(0)` returns 0.

`ArcSin(0.5)` returns 30.

Cos() Function

Returns the cosine of a number. For trigonometric functions the number is the angle in degrees.

Syntax

```
result = Cos (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

Example(s)

`Cos(90)` returns 0.

`Cos(0)` returns 1.

`Cos(60)` returns 0.5.

`20 + 50 * Cos(6 * $second)` produces a sine wave oscillating around 20 with an amplitude of 50 and a period of one minute.

ArcCos() Function

Returns the arcus cosine of a number. It is the reciprocal function to the Cos() function. Use the ArcCos() function to calculate the angle from 0 to 180 degrees whose cosine is equal to that number.

Syntax

```
result = ArcCos (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression in the range of -1 to 1.

Example(s)

`ArcCos(1)` returns 0.

`ArcCos(-0.5)` returns 120.

Tan() Function

Returns the tangent of a specified number. For trigonometric functions the number is the angle in degrees.

Syntax

```
result = Tan (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

Example(s)

`Tan(45)` returns 1.

`Tan(0)` returns 0.

ArcTan() Function

Returns the arcus tangent of a number. It is the reciprocal function to the Tan() function. Use the ArcTan() function to calculate the angle whose tangent is equal to that number.

Syntax

```
result = ArcTan (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression.

Example(s)

`ArcTan(1)` returns 45.

`ArcTan(0)` returns 0.

Return the value of Pi

In a script, you can use the Pi() function to use the constant Pi in mathematical calculations. The Pi() function is exact to 7 digits after the decimal point.

Syntax

```
result = Pi ()
```

Example(s)

`Pi()` returns 3.1415927.

Calculate logarithms

In a script, you can use the following functions to run calculations with logarithms and exponential functions.

Use	To
Log()	Calculate the natural logarithm of a value or expression.
Exp()	Calculate the exponential of a value or expression.
LogN()	Calculate the logarithm of a value or expression to the base of another value or expression.

Log() Function

Returns the natural logarithm of a specified positive number. This is the reciprocal function to the Exp() function.

Note: The natural logarithm of 0 and negative numbers is undefined. If you pass 0 or a negative number to the Log() function, it returns a result of -99.0000000.

Syntax

```
result = Log (number)
```

Parameters

number

A positive literal number, analog tagname, or numeric expression.

Example(s)

`Log(100)` returns 4.6051702.

`Log(1)` returns 0.

Exp() Function

Returns the exponential of a specified number. This is the reciprocal function to the Log() function and is

equivalent to e raised to a power.

Note: If you pass values outside the range of -88.72 to 88.72 to the Exp() function, it returns a result of -99.0000000.

Syntax

```
result = Exp (number)
```

Parameters

number

A literal number, analog tagname, or numeric expression in the range of -88.72 to 88.72.

Example(s)

`Exp(1)` returns 2.7182818.

`Exp(0)` returns 1.

LogN() Function

Returns the logarithm of a positive number to a specified base. This is the reciprocal function to the base to the power of the logarithm.

Example(s)

Syntax

```
result = LogN (number, base)
```

Parameters

number

A positive literal number, analog tagname, or numeric expression.

base

A positive literal number, analog tagname, or expression unequal to 1.

Example(s)

`LogN(8,2)` returns 3.

`LogN(num,btag)` returns the logarithm of num to the base btag.

Note: If you pass invalid parameters to the LogN() function, it returns a result of -99.0000000.

Calculate the square root

In a script, you can use the Sqrt() function to calculate the square root of a specified non-negative number.

Note: If you pass a negative value to the Sqrt() function, it returns a result of -99.0000000.

Syntax

```
result = Sqrt (number)
```

Parameters

number
A non-negative literal number, analog tagname, or numeric expression

Example(s)

`Sqrt(36)` returns 6.
`Sqrt(perftag)` returns the square root of the value held by the tagname perftag.

String operations

You can use many basic string functions in scripts and animation links. You can use these functions to:

- Return parts of strings.
- Change the case of strings.
- Remove and add spaces to strings.
- Handle ASCII values in strings.
- Search and replace in strings.
- Compare strings with each other.
- Return other information about strings, such as their length.

Return parts of strings

In a script, you can use the StringLeft(), StringMid() and StringRight() functions to return parts of strings.

StringLeft() Function

Returns a specified number of characters from the beginning of a string.

Syntax

```
result = StringLeft (string, length)
```

Parameters

string

A literal text, message tagname, or string expression.

length

The numbers of characters to return. A literal number, analog tagname, or numeric expression.

Example(s)

`StringLeft("Hello World",5)` returns "Hello".

`StringLeft("Hello World",20)` returns "Hello World".

`StringLeft("Hello World",0)` returns "Hello World".

Note: If you pass 0 as length to the StringLeft() function, it returns the entire string.

StringRight() Function

Returns a specified number of characters from the end of a string.

Syntax

```
result = StringRight (string, length)
```

Parameters

string

A literal text, message tagname, or string expression.

length

The number of characters to return. A literal number, analog tagname, or numeric expression.

Example(s)

`StringRight("Hello World",5)` returns "World".

`StringRight("Hello World",20)` returns "Hello World".

`StringRight("Hello World",0)` returns "Hello World".

Note: If you pass 0 as length to the StringRight() function, it returns the entire string.

StringMid() Function

Returns a part of a string. You can specify the starting point and how many characters to return.

Syntax

```
result = StringMid (string, startpos, length)
```

Parameters

string

A literal text, message tagname, or string expression.

startpos

The starting position in the string. A literal number, analog tagname, or numeric expression.

length

The number of characters to return. A literal number, analog tagname, or numeric expression.

Example(s)

`StringMid("Hello World",5,4)` returns "o Wo".

`StringMid("Hello World",7,50)` returns "World".

`StringMid("Hello World",4,0)` returns "lo World".

Note: If you pass 0 as length to the StringMid() function, it returns the entire string after the starting position.

Change case of strings

In a script, you can use the StringLower() and StringUpper() functions to return a specified string in lowercase and uppercase. You can assign the result to the specified string to perform a conversion from upper to lowercase or vice versa.

StringLower() Function

Returns the lowercase equivalent of a string.

Syntax

```
result = StringLower (string)
```

Parameters

string

A literal text, message tagname, or string expression.

Example(s)

`StringLower("TURBINE")` returns "turbine".

`StringLower("The Value Is 22.2")` returns "the value is 22.2".

`mtag = StringLower(mtag)` converts the message value of mtag to lowercase.

StringUpper() Function

Returns the uppercase equivalent of a string.

Syntax

```
result = StringUpper (string)
```

Parameters

string

A literal text, message tagname, or string expression.

Example(s)

`StringUpper("abcd")` returns "ABCD".

`StringUpper("The Value Is 22.2")` returns "THE VALUE IS 22.2".

`mtag = StringUpper(mtag)` converts the message value of mtag to uppercase.

Remove spaces from strings

In a script, you can trim leading and trailing spaces (blanks) from strings by using the StringTrim() function. You can use this to remove unwanted spaces from a string, for example after a user input.

StringTrim() Function

Trim leading and trailing spaces (blanks) from strings. You can use this to remove unwanted spaces from a string, for example after a user input.

Syntax

```
result = StringTrim (string, trimtype)
```

Parameters

string

A literal text, message tagname, or string expression.

trimtype

A literal value, analog tagname, or numeric expression that determines which spaces to remove:

- 1 = Leading spaces.
- 2 = Trailing spaces.
- 3 = Leading and trailing spaces.

Remarks

This function removes all leading and trailing white spaces from a string. White spaces are spaces (ASCII 0x20) and control characters in the range from ASCII 0x09 to 0x0D.

Example(s)

To remove all spaces in a message tag, mtag, with an action script, use the following script:

```
DIM i AS INTEGER;
DIM tmp AS MESSAGE;
mtag = StringTrim(mtag,3); {mtag is trimmed}
FOR i = 1 TO StringLen(mtag) {run variable i over the characters of mtag}
IF StringMid(mtag, i, 1)<>" " THEN {i-th character is not space} tmp = tmp +
StringMid(mtag, i, 1); {
add that character to tmp}
ENDIF;
NEXT;
mtag = tmp; {pass tmp back to mtag}.
```

Other examples:

`StringTrim(" Joe ",1)` returns "Joe ".

`StringTrim(" Joe ",2)` returns " Joe".

This script removes all spaces from the left and the right of the mtag value:

```
mtag = StringTrim(mtag,3)
```

Format strings with spaces

In a script, you can use the StringSpace() function to add spaces (blanks) to strings.

Syntax

```
result = StringSpace (number)
```

Parameters

number

A literal number, numeric tagname, or numeric expression.

Example(s)

`StringSpace(4)` returns a string consisting of 4 blanks.

`"Pump"+StringSpace(1)+"Station"` returns "Pump Station".

Convert between characters and ASCII codes

In a script, you can convert characters of a string to ASCII codes and ASCII codes back to characters by using the StringChar() and StringASCII() functions.

These functions do not support multiple byte character sets. Only characters in the range of 0-255 are supported.

Using ASCII codes is useful if you wish to perform some numeric calculation on a string (for example for encoding a string).

StringChar() Function

Returns a single character corresponding to a specified ASCII code.

Syntax

```
result = StringChar (ASCIICode)
```

Parameters

ASCIICode

A literal number, numeric tagname, or numeric expression in the range of 0 to 255.

Remarks

This function is very useful for passing control characters to external devices (such as printers or modems) or double quotes to SQL queries.

Example(s)

```
StringChar(65) returns "A".
```

This script returns "Hello World" enclosed by double quotes:

```
StringChar(34)+"Hello World"+StringChar(34)
```

This script returns "Hello World" where both words are separated by a carriage return and a line feed:

```
"Hello"+StringChar(13)+StringChar(10)+"World"
```

StringASCII() Function

Returns the ASCII code of the first character of a string.

Syntax

```
result = StringASCII (string)
```

Parameters

string

A literal string, message tagname, or string expression.

Example(s)

```
StringASCII("A") returns 65.
StringASCII("hello world") returns 104.
```

Search and replace text in strings

For languages that use single-byte character sets (such as English) you can use the StringInString() and StringReplace() functions in a script to perform limited search and replace functionality on message tags.

Use	To
StringInString()	Search for a certain string in another string and return the result as a position.
StringReplace()	Replace certain characters or words with other characters or words in a specified string and return the result as a new string.

StringInString() Function

Returns the first position of a specified string in another string.

Syntax

```
result = StringInString (string, searchfor, startpos, casesens)
```

Parameters

string

This is the string to searched. A literal string, message tagname, or string expression.

searchfor

This is the string that is to be searched for. A literal string, message tagname, or string expression.

startpos

This is the starting position in string of the search. A literal value, numeric tagname, or numeric expression.

casesens

Determines whether the search is case sensitive. Can be 0 or 1, discrete tagname, or Boolean expression.

0 - search is not case sensitive (uppercase and lowercase are considered the same).

1 - search is case sensitive (uppercase and lowercase are considered to be different).

Remarks

Use this function to determine if a certain string is contained in a message tag. You can specify the starting position for the search and whether the letter case is to be respected.

Example(s)

This script returns 5—because the first "M" in "MTX" is in the fifth position of the string:

```
StringInString("DBO MTX-010","MTX",1,0)
```

This script returns 3—because the first "M" in "MTX" is in the third position in the string:

```
StringInString("T-MTX 010 MTX","MTX",1,0)
```

This script returns 11—because the first "M" in "MTX" after the 8th position is in the 11th position in the string:

```
StringInString("T-MTX 010 MTX","MTX",8,0)
```

This script returns 11—because the first string that matches MTX in the correct case is in the 11th position:

```
StringInString("t-mtx 030 MTX", "MTX",1,1)
```

This script returns 0—because there is no "Mty" in the string:

```
StringInString("t-mtx 030 MTY-Mtx","Mty",1,1)
```

StringReplace() Function

Searches for a string within another string and, if found, replaces it with yet another string. You can specify:

- Case-sensitivity - This determines if uppercase letters and lowercase letters are to be treated as identical letters or not.
- Number of occurrences to replace - This is useful if more than one occurrence of the search string is found.
- Match whole words - Use this if the search string is a whole word.

Note: This function does not support double byte character sets.

Syntax

```
result = StringReplace (string, searchfor, replacewith, casesens, numtoreplace,
matchwholewords)
```

Parameters

string

The string to search within. A literal string, message tagname, or string expression.

searchfor

The string that is to be searched for. A literal string, message tagname, or string expression.

replacewith

The string that is used as replacement. A literal string, message tagname, or string expression.

casesens

Determines whether the search is case sensitive. Can be 0 or 1, discrete tagname or Boolean expression.

0 - search is not case sensitive (uppercase and lowercase are considered the same)

1 - search is case sensitive (uppercase and lowercase are considered to be different)

numtoreplace

The number of replacements to make. Set it to -1 to replace all occurrences of the found search string. A literal integer value, integer tagname, or integer expression.

matchwholewords

Determines whether only whole words are matched. Can be 0 or 1, discrete tagname, or Boolean expression.

0 - the function looks for the search string characters anywhere in the string

1 - only whole words are matched

Example(s)

This statement replaces only the first occurrence and returns "MTY 030 MTX".

```
StringReplace("MTX 030 MTX","MTX","MTY",0,1,0)
```

This statement replaces all occurrences and returns "MTY 030 MTY".

```
StringReplace("MTX 030 MTX","MTX","MTY",0,-1,0)
```

This statement replaces all occurrences that match the case and returns "MTY 030 mtx".

```
StringReplace("MTX 030 mtx","MTX","MTY",1,-1,0)
```

This statement replaces all occurrences that are whole words and returns "MTY 030 QMTX".

```
StringReplace("MTX 030 QMTX","MTX","MTY",0,-1,1)
```

Return information about strings

In a script, you can use the StringLen() and StringTest() functions to return the length of a specified string and to test whether a character is in a certain group of characters.

StringLen() Function

Returns the length of a specified string, including non-visible characters.

Syntax

```
result = StringLen (string)
```

Parameters

string

A literal string, message tagname, or string expression.

Example(s)

```
StringLen("Twelve percent") returns 14.
StringLen("12%") returns 3.
StringLen("The end." + StringChar(13)) returns 9.
```

StringTest() Function

Tests whether the first character of a string is in a certain group of characters.

Syntax

```
result = StringTest (string, group)
```

Parameters

string

A literal string, message tagname, or string expression.

group

The number of the group to test the character against. A literal value, integer tagname, or integer expression in the range of 1 to 11.

1 - alphanumeric characters (A-Z, a-z, 0-9)

2 - numeric characters (0-9)

3 - alphabetic characters (A-Z, a-z)

4 - uppercase characters (A-Z)

5 - lowercase characters (a-z)

6 - punctuation characters (ASCII 0x21 - 0x2F), for example !,@,#,$,%,^,&,* and so on

7 - ASCII characters (ASCII 0x00 - 0x7F)

8 - Hexadecimal characters (0-9, A-F, a-f)

9 - Printable characters (ASCII 0x20 - 0x7E)

10 - Control characters (ASCII 0x00 - 0x1F and 0x7F)

11 - White space characters (ASCII 0x09 - 0x0D and 0x20)

Example(s)

This string returns a 1—because "A" is an alphanumeric character:

```
StringTest("ACB123",1)
```

This string returns a 0—because "A" is not a lowercase character:

```
StringTest("ABC123",5)
```

Compare strings

In a script, you can use the StringCompare(), StringCompareNoCase() and StringCompareEncrypted() functions to compare two strings.

Use	To
StringCompare()	Make a case-sensitive comparison.
StringCompareNoCase()	Make a case-insensitive comparison.
StringCompareEncrypted()	Compare an encrypted string with an unencrypted string.

StringCompare() Function

Compares two strings with each other and returns a Boolean result (0 = strings are equal). The case of each letter is respected so that, for example, 'A' is considered not equal to 'a'.

Syntax

```
result = StringCompare (string1, string2)
```

Parameters

string1

A literal string, message tagname, or string expression.

string2

A literal string, message tagname, or string expression.

Example(s)

```
StringCompare ("Apple","Apple") returns 0.
StringCompare ("Apple","apple") returns 1.
```

This string compares the two message tags and returns a discrete result (0 or 1):

```
StringCompare (mtag1, mtag2)
```

StringCompareNoCase() Function

Compares two strings with each other and returns an integer result. The case of each letter is not respected so that, for example, 'A' is considered equal to 'a'.

The integer result returns:

- 0 if both strings are identical (ignoring case).
- Non-zero otherwise. The result is the difference of ASCII values between the differentiating character (ignoring case).

  Note: The result of the StringCompareNoCase() function can be used as a discrete result, as all non-zero values are considered to equal TRUE in InTouch scripting.

Syntax

```
result = StringCompareNoCase (string1, string2)
```

Parameters

string1

A literal string, message tagname, or string expression.

string2

A literal string, message tagname, or string expression.

Example(s)

This string returns 0—because the strings are considered identical:

```
StringCompareNoCase("Apple","apple")
```

This string returns -6—because the strings are considered not identical and the ASCII values of the first differentiating character "p" minus the ASCII value of the corresponding letter "v" equals -6:

```
StringCompareNoCase("Apple","Avocado")
```

StringCompareEncrypted() Function

Compares an encrypted string with an unencrypted string and returns a Boolean result. You can use this function for password verification. For more information on password encryption, see Animate objects.

Syntax

```
result = StringCompareEncrypted (plain, encrypted)
```

Parameters

plain

A literal string, message tagname, or string expression.

encrypted

An encrypted message tagname.

Example(s)

This script returns 1 when the plain text and the encrypted text are identical, otherwise it returns 0. Passwd is a message tag containing a value from an encrypted user input. PlainTxt is a message tag against which the user input is to be compared.

```
StringCompareEncrypted(PlainTxt, Passwd)
```

Convert data types

In a script, you can convert values contained in tagnames to other data types by using conversion QuickScripts. This allows you to manipulate string data with mathematical functions or to log values to the Log Viewer for debugging purposes.

- Text() Function
- StringFromIntg() Function
- StringFromReal() Function
- StringToIntg() Function
- StringToReal() Function

- DText() Function

Text() Function

The Text() function returns the value of a number as a string according to a specified format. You may want to do this to format a value in a certain way or to combine the result with other string values for further processing.

Syntax

```
result = Text (number, format)
```

Parameters

number

A literal numeric value, analog tagname, or numeric expression.

format

Use "#", "0", ".", or ",".
Use "#" to represent a digit, "." to represent the decimal separator, "0" to force a leading zero, and "," to insert a comma.
If you use a zero in the format, it must be followed by zeros. All places to the right of the decimal point must always be zeros. For example, 000.00 is correct, while #0#0.0# is incorrect.
The function rounds the value, if necessary. A literal string, message tagname, or string expression.

Example(s)

`Text(66,"#.00")` returns "66.00".

`Text (1234,"#")` returns "1234".

`Text (123.4, "#,##0.0")` returns "123.4".

`Text (12.3, "0,000.0")` returns "0,012.3".

`Text(3.57,"#.#")` returns "3.6".

This script returns the string "Reactor Pressure is 1690.3 mbar" if the analog tagname "pressure" contains the value 1690.2743.

```
"Reactor Pressure is "+Text(pressure,"#.#")+" mbar"
```

StringFromIntg() Function

In a script, you can convert an integer value to a string value by using the StringFromIntg() function.

This function returns the string value of an integer value and performs a base conversion at the same time. This can be used, for example, to show text together with integer values or for converting integer values to hexadecimal numbers.

Syntax

```
result = StringFromIntg (number, base)
```

Parameters

number

A literal integer value, integer tagname, or integer expression.

base

The base of the conversion. This is used for converting the value to a different base, such as binary (2), decimal (10) or hexadecimal (16). A literal integer value, integer tagname, or integer expression.

Example(s)

`StringFromIntg(26,2)` returns "11010" (binary.

`StringFromIntg(26,8)` returns "32"—because (base 8: 26 = 3*8 + 2)

`StringFromIntg(26,10)` returns "26" (decimal).

`StringFromIntg(26,16)` returns "1A" (hexadecimal).

StringFromReal() Function

In a script, you can convert an real value to a string value by using the StringFromReal() function.

You can also specify to:

- Round the value to a specified precision.
- Pass the value in exponential notation.

This can be used, for example, to show text together with real values or for showing real numbers with exponential notation.

Syntax

```
result = StringFromReal (number, precision, type)
```

Parameters

number

A literal value, analog tagname, or numeric expression.

precision

Specifies how many decimal places are to be used. A literal integer value, integer tagname, or integer expression.

type

Specifies if the exponential notation is to be used. A literal string, message tagname, or string expression.

"f" - Use floating point notation.

"e" - Use exponential notation with lowercase "e".

"E" - Use exponential notation with uppercase "E".

Example(s)

`StringFromReal(263.355, 2,"f")` returns "263.36".

`StringFromReal(263.355, 2,"e")` returns "2.63e2".

`StringFromReal(263.55, 3,"E")` returns "2.636E2".

`StringFromReal(0.5723, 2,"E")` returns "5.72E-1".

StringToIntg() Function

In a script, you can convert a value contained in a string to an integer value by using the StringToIntg() function.

You can use this to read a value contained at the beginning of a string into an integer tag for further mathematical operations.

Syntax

```
result = StringToIntg (string)
```

Parameters

string

A literal string, message tagname, or string expression.

Remarks

The function checks the first character of the string. If it is a number, it attempts to read this and the following characters as an integer number until a non-numeric character is met. The function ignores leading spaces in the string.

Example(s)

`StringToIntg("ABCD")` returns 0.

`StringToIntg("13.4 mbar")` returns 13.

`StringToIntg("Pressure is 13.4")` returns 0.

To extract the first integer from a string (mtag) that is not at the beginning and to store it in the integer tag itag, use the following action script:

```
DIM i AS INTEGER;
DIM tmp AS INTEGER;
FOR i = 1 TO StringLen(mtag) {run variable i over the characters of mtag}
tmp = StringASCII(StringMid(mtag, i, 1)) - 48; {detect ASCII value}
IF (tmp>=0 AND tmp<10) THEN {if ASCII value represented "0" - "9"}
itag = StringToIntg(StringMid(mtag, i, 0)); {set itag to value from that position and exit
loop}
EXIT FOR;
ENDIF;
NEXT;
```

StringToReal() Function

In a script, you can convert a value contained in a string to a real value by using the StringToReal() function.

You can use this to read a value contained at the beginning of a string into a real tag for further mathematical operations.

Note: This function also supports the exponential notation and converts a string expression 1e+6 correctly to 1000000.

Syntax

```
result = StringToReal (string)
```

Parameters

string

A literal string, message tagname, or string expression.

Remarks

The function checks the first character of the string. If it is a number, it attempts to read this and the following characters as a real number until a non-numeric character is met. The function ignores leading spaces in the string.

To extract the first real number from a string (message tag mtag) that is not at the beginning and store it in the real tag rtag1, use the following script:

```
DIM i AS INTEGER;
DIM tmp AS INTEGER;
FOR i = 1 TO StringLen(mtag) {run variable i over the characters of mtag}
tmp = StringASCII(StringMid(mtag, i, 1)) - 48; {detect ASCII value}
IF (tmp>=0 AND tmp<10) THEN {if ASCII value represented "0" - "9"}
rtag = StringToReal(StringMid(mtag, i, 0)); {set rtag to value from that position and exit loop}
EXIT FOR;
ENDIF;
NEXT;
```

Example(s)

```
StringToReal("ABCD") returns 0.
StringToReal("13.4 mbar") returns 13.4.
StringToReal("Pressure is 13.4") returns 0.
```

DText() Function

In a script, you can convert a Boolean value to a string value by using the DText() function. You can use this function to use customized message display animation links.

This function returns different string values depending on the value of a Boolean value.

Syntax

```
result = Dtext (Boolean, stringtrue, stringfalse)
```

Parameters

Boolean

A literal Boolean value, discrete tagname, or Boolean expression.

stringtrue

The string to be returned if Boolean is true. A literal string value, message tagname, or string expression.

stringfalse

The string to be returned if Boolean is false. A literal string value, message tagname, or string expression.

Example(s)

This script returns "Running" if the discrete tagname switch is TRUE, otherwise it returns "Stopped".

```
DText(switch,"Running","Stopped")
```

This script returns the On and Off Messages of another discrete tag switch2 depending on the value of the discrete tag switch1.

```
DText(switch1,switch2.OnMsg,switch2.OffMsg)
```

Work with InTouch windows at run time

In a script, you can control the behavior and appearance of InTouch windows. You can also write a script using QuickScripts to print individual InTouch windows or the entire screen.

Expose window name property

You can use **GetWindowName** script function to help the run-time environment reduce scripting necessary to load windows with the current implementation. It enables you to retrieve the name of the window under which the function has been called.

GetWindowName() Function

Retrieves the name of the window under which the function has been called.

Syntax

The syntax of the script function is as follows:

```
resultcode = GetWindowName(tagname);
```

Resultcode indicates the success or failure of the script function. The resultcode can be a Discrete/Integer/Real data type. Resultcode will be 1 or 0, based on the success or failure of the script function:

- Resultcode is 1 when the script function is called from window context.

- Resultcode is 0 when the script function is called from non-window context.

Note: Configuration of the return value for the script function is optional. This is similar to the existing script functions in InTouch.

Parameter

TagName

The tagname is the out parameter for this function. This will be a message tag to retrieve the window name.

The parameter for the function can be any of the following:

- InTouch tag
- Dot field
- Local variable in the script
- Remote tag reference
- Galaxy reference

The default text loaded by the script browser is:

GetWindowName(*TagName*);

Note: "TagName" is a message tag or Remote Tag Reference (RTR) which is of message type.

Return Value

The **GetWindowName** script function returns the window name with the return value of 1 in the following scenarios:

- Window scripts (All Condition Types)
- Push button action scripts (All Condition Types)

The **GetWindowName** script function returns an empty string with return value of 0 in the following scenarios:

- Application scripts
- Key scripts
- Condition scripts
- Data change scripts
- ActiveX event scripts
- Quick script functions (Synchronous and Asynchronous)

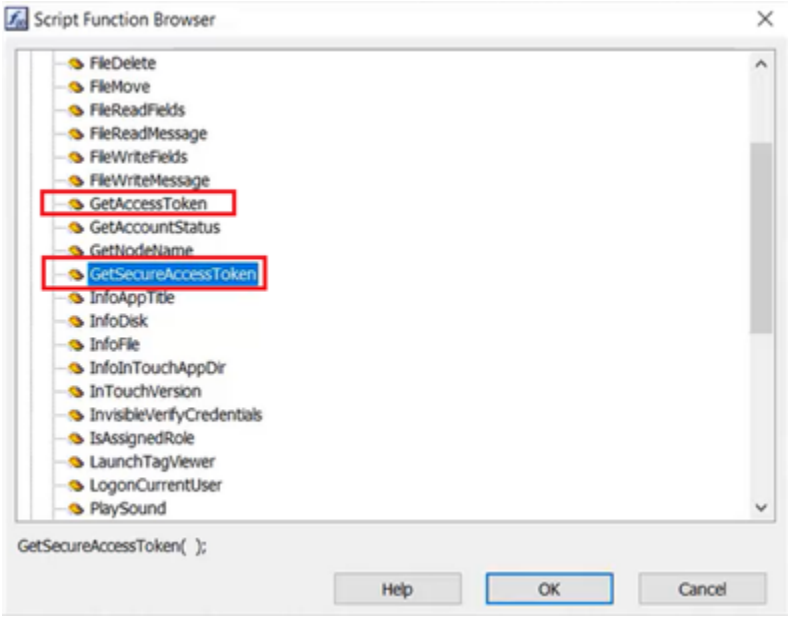
Expose the latest access token

By using the scripts **GetAccessToken** and **GetSecureAccessToken**, the CONNECT access token can be exposed to InTouch and can be used by controls and widgets for Single Sign-On in connected experience at runtime.

You can use the **GetAccessToken** script function to return the latest token value and you can use the **GetSecureAccessToken** script function to return the latest token value in a secure or encrypted way. As the access tokens are time bound and the values keep changing periodically, this script function helps you to get the

latest access token value.

In the **Script Function Browser**, you can see the built-in scripts **GetAccessToken** and **GetSecureAccessToken**. It will log the value of the access token is in the logger.



GetAccessToken() Function

GetAccessToken () provides authentication token, which will be accepted by controls like Trend Control, Alarm Client Control and other InTouch controls, to support Single Sign-On functionality.

Syntax

The syntax of the script function is as follows:

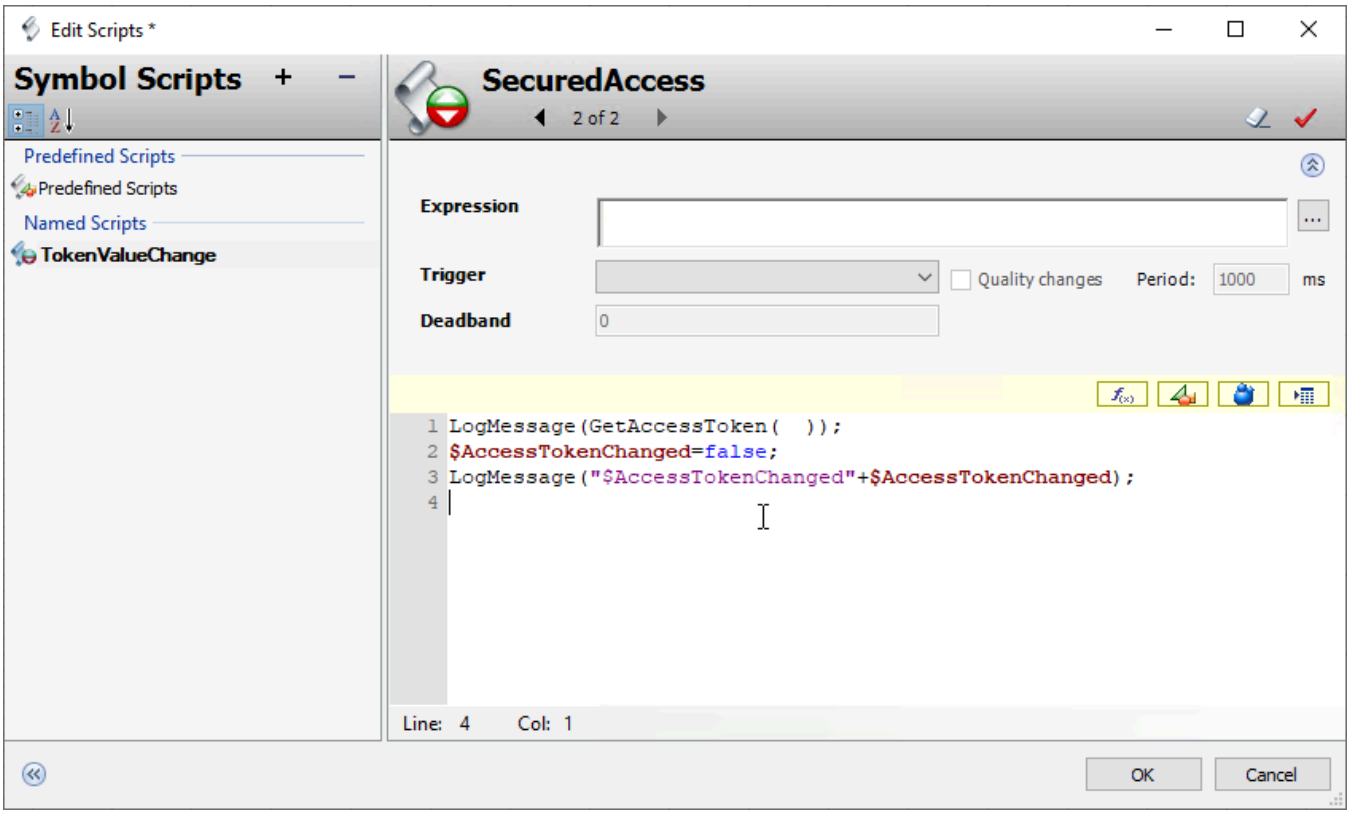
```
resultCode = GetAccessToken();
```

Resultcode indicates the latest access token.

Example:

To get the token value renewed every time it expires, you can use the script as follows:

1. In the Symbol Script window, select **Add Script**.
2. Provide a name to the script.
3. Select the **Display Script Function Browser** icon.
4. In the **Script Browser** screen, under **InTouch,** select **GetAccessToken** to insert the script function or you can manually type.
5. Select **OK**.



Return value

The GetAccessToken script function returns the access token with the a token value, if user is authenticated using CONNECT.

The GetAccessToken script function returns an empty string, if user is not authenticated using CONNECT.

GetSecureAccessToken() Function

GetSecureAccessToken () provides secure authentication token, which will be accepted by controls like Trend Control, Alarm Client Control and other InTouch controls, to support Single Sign-On functionality.

Syntax

The syntax of the script function is as follows:

```
resultCode = GetSecureAccessToken();
```

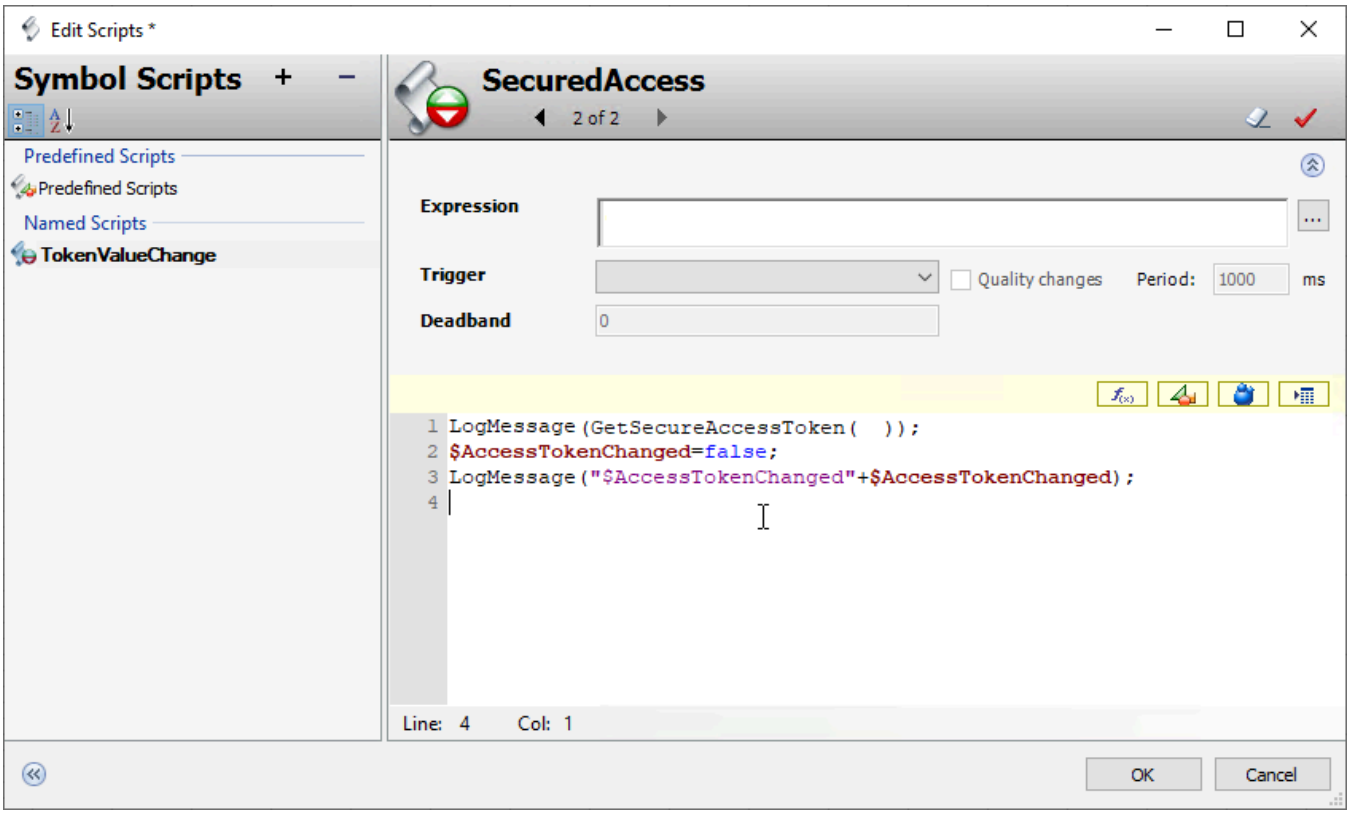
Resultcode indicates the latest access token.

Example:

To get the secured token value renewed every time it expires, you can use the script as follows:

1. In the Symbol Script window, select **Add Script**.
2. Provide a name to the script.
3. Select the **Display Script Function Browser** icon.

4. In the **Script Browser** screen, under **InTouch,** select **GetSecureAccessToken** to insert the script function or you can manually type.
5. Select **OK**.



Return Value

The GetSecureAccessToken script function returns the access token with the a token value, if user is authenticated using CONNECT.

The GetSecureAccessToken script function returns an empty string, if user is not authenticated using CONNECT.

Retrieve the connection status

You can use **GetTokenConnectionStatus** script function to retrieve connection status in AVEVA Operations Control connected experience.

GetTokenConnectionStatus() function

Retrieves status of the connection to the AVEVA Identity Manager and CONNECT in AVEVA Operations Control connected experience.

Syntax

The syntax of the script function is as follows:

```
resultcode = GetTokenConnectionStatus();
```

Resultcode indicates the token for connection status.

Return value

- Resultcode is 0 when the connection status is Fully Connected. That is the application is connected to both AVEVA Identity Manager and CONNECT.
- Resultcode is 1 when the connection status is Partially Connected. That is the application is connected to AVEVA Identity Manager and disconnected from CONNECT.
- Resultcode is 2 when the connection status is Fully Disconnected. That is the application is disconnected from both AVEVA Identity Manager and CONNECT.

Example:

int AccessTokenStatus=GetTokenConnectionStatus()

Show a list of open windows

In a script, you can show a dialog box containing the list of InTouch windows that are currently open using the OpenWindowsList() function.

OpenWindowList() Function

Shows a dialog box containing the list of InTouch windows that are currently open.

You can not use this function in an animation link.

Syntax

```
[result = ]OpenWindowsList();
```

Example(s)

This script opens the **Open Windows List** dialog box and shows all InTouch windows that are currently open.

```
OpenWindowsList()
```

Note: When the **Use In-Memory Window Cache** WindowViewer option is enabled, closed windows may appear in the list created by the OpenWindowList() function.

Check if a window is open, closed, or exists

In a script, you check if an InTouch window is open, is closed, or does not exist by using the WindowState() function.

WindowState() Function

Checks if an InTouch window is open, is closed, or does not exist.

Syntax

```
result = WindowState (windowname)
```

Parameters

windowname

Name of the window. A literal string value, message tagname, or string expression.

Return Value

An integer value with the following meaning:

0 - InTouch window exists and is currently closed

1 - InTouch window exists and is currently open

2 - InTouch window does not exist

Example(s)

This script returns 0, if the InTouch window Main exists, but is not open.

```
WindowState("Main")
```

Open InTouch windows

In a script, you can open an InTouch window by using one of the following QuickScript functions:

Use	To
Show	Open an InTouch window at the position defined in its location settings.
ShowAt()	Open an InTouch window at a specified position. The opened window is centered on the position. This function can also be used to move an opened window.
ShowHome	Open the InTouch window(s) you specified in the **Home Windows** tab in the WindowViewer **Properties** dialog box and closes any other windows.
ShowTopLeftAt()	Open an InTouch window at a specified position. The opened window aligns its top left corner to the position. This function can also be used to move an opened window.

Show() Function

Opens an InTouch window at its default position.

Syntax

```
Show windowname
```

Parameters

windowname

The name of the window to be opened. A literal string value, message tagname, or string expression.

Example(s)

This script opens the window Main.

```
Show "Main";
```

This script opens the window with the name that is stored in the wname message tag.

```
Show wname;
```

ShowAt() Function

Opens an InTouch window at a specified position. It also can move an already open InTouch window to a specified position. The position is the center point of the window.

Note: The window will not be centered if one of its edges is off-screen.

Syntax

```
ShowAt (windowname, xpos, ypos)
```

Parameters

windowname

The name of the window to be opened or moved.

xpos

The horizontal position in pixels that the window center is to be moved to. A literal value, analog tagname, or numeric expression.

ypos

The vertical position in pixels that the window center is to be moved to. A literal value, analog tagname, or numeric expression.

Example(s)

This script opens the window Main so that it is centered at the position x:450, y:130.

```
ShowAt("Main",450,130);
```

This script opens the window called UserDialog and positions it, so that its center is over the center position of the object that called this function (for example a button).

```
ShowAt("UserDialog",$ObjHor,$ObjVer);
```

ShowHome() Function

Opens the InTouch window(s) you specified in the **Home Windows** tab in the WindowViewer **Properties** dialog box and closes any other windows.

Syntax

```
ShowHome;
```

ShowTopLeftAt() Function

Opens an InTouch window at a specified position. Can also be used to move an open window.

Syntax

```
ShowTopLeftAt (windowname, xpos, ypos)
```

Parameters

The name of the window to be opened or moved.

xpos

The horizontal position in pixels that the window left edge is to be moved to. A literal value, analog tagname, or numeric expression.

ypos

The vertical position in pixels that the window top edge is to be moved to. A literal value, analog tagname, or numeric expression.

Example(s)

This script opens the window Main so that its top left corner is positioned at x:450, y:130.

```
ShowTopLeftAt("Main",450,130);
```

Move and resize a window

In a script, you can move and resize an opened InTouch window with the WWMoveWindow() function. The new position and new size apply temporarily while the specified window is open.

WWMoveWindow() Function

Moves and resizes an opened InTouch window to a specified position and specified size. The new position and new size apply temporarily while the specified window is open.

Syntax

```
WWMoveWindow (windowname, xpos, ypos, xsize, ysize)
```

Parameters

windowname

The name of the window to be opened or moved.

xpos

The horizontal position in pixels that the window left edge is to be moved to. A literal value, analog tagname, or numeric expression.

ypos

The vertical position in pixels that the window top edge is to be moved to. A literal value, analog tagname, or numeric expression.

xsize

The horizontal size in pixels for the specified window. A literal value, analog tagname, or numeric expression.

ysize

The vertical size in pixels for the specified window. A literal value, analog tagname, or numeric expression.

Hide InTouch windows

In a script, you can hide InTouch windows by using either of the following functions.

Use	To
Hide	Hide a specified window.
HideSelf	Hide the currently active window.

Hide() Function

Hides (closes) an InTouch window.

Syntax

```
Hide windowname;
```

Parameters

windowname

The name of the window to be hidden. A literal string value, message tagname, or string expression.

Example(s)

This script hides the window called UserConfirmation.

```
Hide "UserConfirmation";
```

HideSelf() Function

Hides (closes) the currently active InTouch window.

Note: This function can only be used in an action QuickScript.

Syntax

```
HideSelf;
```

Example(s)

```
HideSelf;
```

Change the color of a window

In a script, you can change the color of an open InTouch window by using the ChangeWindowColor() function.

ChangeWindowColor() Function

Changes the color of an open InTouch window and returns a result code.

Syntax

```
Result = ChangeWindowColor (windowname, rValue, gValue, bValue)
```

Parameters

windowname

The name of the window for which the color is to be changed. A literal string value, message tagname, or string expression.

rValue

The intensity of the red color. A literal integer value, integer tagname, or integer expression in the range of 0 to 255.

gValue

The intensity of the green color. A literal integer value, integer tagname, or integer expression in the range of 0 to 255.

bValue

The intensity of the blue color. A literal integer value, integer tagname, or integer expression in the range of 0 to 255.

Return Value

A value with the following meaning:

0 - Failure, window is not defined or RGB value is out of range.

1 - Success.

2 - Failure. The window exists, but it is not open.

Print windows at run time

In a script, you can print individual InTouch windows or the entire WindowViewer screen by using the PrintWindow() or PrintScreen() functions. You can also set the printer you want to use with the SetWindowPrinter() function.

SetWindowPrinter() Function

At run time, you can set the printer you want to use with the SetWindowPrinter() function.

Note: The printer set with this function is also the printer that is used with the PrintHT() function.

Syntax

```
SetWindowPrinter (printername)
```

Parameters

printername

The name of the printer, either as network share or as printer name as it appears in its property window. A literal string value, message tagname, or string expression.

Example(s)

In this example, PRTSRV1 is the node name and PRT22SW1 is the share name given to the printer.

```
SetWindowPrinter("\\PRTSRV1\PRT22SW1");
```

In this example, Epson LX-300 is the name of the printer as seen in the Properties window of the printer.

```
SetWindowPrinter("Epson LX-300");
```

In this example, MyPrinter is a message tag containing the name of an installed windows printer or the path to a shared network printer.

```
SetWindowPrinter(MyPrinter);
```

Recommendations for printing

The following list contains some issues to consider when printing. These are applicable to printing a single window or printing the WindowViewer screen.

- Open the window(s) to be printed before printing it. Otherwise Windows and ActiveX Controls may not print

correctly.

- You cannot print to the same printer that is currently printing alarms.
- Avoid overlapping of windows and objects on the window when printing.
- Use True Type fonts whenever possible. The default InTouch font (System) is not a True Type font.
- For faster printing consider using a white background, fewer objects, and text instead of graphics.
- WindowViewer waits a certain amount of time before the window is sent to the printer queue. During this time, WindowViewer updates any I/O values for that window in the background. To change this waiting time, open the intouch.ini file and change or add the following line (in milliseconds): `PrintWindowWait=10000`

PrintWindow() Function

In a script, you can print an InTouch window with the PrintWindow() function.

Note: Scripts containing the PrintWindow() function cannot print the following Industrial graphic controls within an InTouch window: ListBox, DateTimePicker, CalenderControl, EditBox, CheckBox, RadioButtonGroup, ComboBox, AlarmClient or TrendClient.

Syntax

```
[result = ] PrintWindow (windowname, leftmargin, topmargin, width, height, options);
```

Parameters

windowname

The name of the window to be printed. A literal string value, message tagname, or string expression.

leftmargin

Left margin offset (in inches). A literal numeric value, analog tagname, or numeric expression.

topmargin

Top margin offset (in inches). A literal numeric value, analog tagname, or numeric expression.

width

Printout width (in inches). Set this value to 0 for largest aspect ratio. A literal numeric value, analog tagname, or numeric expression.

height

Printout height (in inches). Set this value to 0 for largest aspect ratio. A literal numeric value, analog tagname, or numeric expression.

options

A discrete value, 0 or 1, that is only used if width and height are 0. A literal Boolean value, discrete tagname or Boolean expression. Set to:

1 - The window is printed with the largest aspect ratio that is an integer multiple of the window size.

0 - The window is printed with the largest aspect ratio that fits on the page.

Note: If the window contains a bitmap, set options to 1 to prevent the bitmap from being stretched.

Return Value

0 - Printing job is not queued successfully, or window does not exist

1 - Printing job is queued successfully

PrintScreen() Function

You can write a script to print the entire WindowViewer screen with the PrintScreen() function.

Syntax

```
PrintScreen (ScreenOption, PrintOption)
```

Parameters

ScreenOption

Determines how much of the WindowViewer screen is to be printed. A literal integer value, integer tagname, or integer expression.

1 - Print the client area, no menus (default)

2 - Print the entire window area, including menus

PrintOption

Determines how the printed image is to be stretched to fit on the printout.

- 1 - Best Fit:
  image is stretched so that it fits either horizontally or vertically on the printout without changing the aspect ratio. (default)
- 2 - Vertical Fit:
  image is stretched so that it fits vertically on the printout without changing the aspect ratio. The image may be cut off horizontally.
- 3 - Horizontal Fit:
  image is stretched so that it fits horizontally on the printout without changing the aspect ratio. The image may be cut off vertically.
- 4 - Stretch to Page:
  image is stretched so that it fits horizontally and vertically on the printout. The aspect ratio may change but the image is not truncated.
- Invalid options, including 0, default to Best Fit.

Note: Popup windows that extend beyond the WindowViewer screen area are cut off.

Example(s)

This script sends a printout of the current entire WindowViewer screen area without menus to the printer queue. The printout contains the screen area stretched so that it fills the printout dimensions.

```
PrintScreen(1,4);
```

PrintHT() Function

In a script, you can create a button to print the historical trend by linking it to an action QuickScript that executes the PrintHT QuickScript function.

Use the PrintWindow() function instead of the PrintHT() function when you want to print the entire window instead of just the trend chart.

Note: Printing the Historical Trend using the Print option or the PrintHT() function will not print the x & y values. Use PrintWindow() or PrintScreen() to print the x & y values.

Syntax

```
PrintHT(HistTrendTagname);
```

Parameter

HistTrendTagname

The history trend tag name for the history trend to be printed.

Start tag viewer

Tag Viewer is a run-time application that allows you to watch and monitor tags and to modify tag values. For information about Tag Viewer and its use, see *AVEVA™ InTouch HMI Application Run Time.*

LaunchTagViewer() function

You can start Tag Viewer only when WindowViewer is running, and only after Tag Viewer has been enabled in WindowMaker.

For information about enabling Tag Viewer, see Configure general WindowViewer properties in *AVEVA™ InTouch HMI Creating Standards for InTouch HMI Components*.

Syntax

```
LaunchTagViewer()
```

Remarks

The LaunchTagViewer() function can be executed from any script type except the application scripts OnStartup and OnShutdown.

If Tag Viewer has not been enabled in WindowMaker, calling the function will not start Tag Viewer and a warning message will appear in the logger.

You must have adequate security privileges to start Tag Viewer.

Work with date and time information

In a script, you can use system tags and QuickScript functions to use system time and date settings in calculations. InTouch scripting also supports calculations involving multiple time zones and daylight saving time.

Retrieve numerical date and time information

In a script, you can use a variety of numerical system tags and one script function to retrieve information on the system time and date. These tags and the script function can be used in other mathematical operations. The following system tags and script functions are available:

Use	To
$Year	Return the current year.
$Month	Return the current month of the year.
$Day	Return the current day of the month.
$Hour	Return the current hour of the day.
$Minute	Return the current minute of the hour.
$Second	Return the current second of the minute.
$Msec	Return the current milliseconds.
$Time	Return the time in milliseconds that have passed since midnight in the local time zone.
$Date	Return the number of whole days that have passed since the 1st January 1970 in the local time zone.
$DateTime	Return the number of days (including fractions of a day) that have passed since the January 01, 1970 in the local time zone.
DateTimeGMT()	Return the number of days (including fractions of a day) that have passed since the 1st January 1970 in Coordinated Universal Time (UTC).

$Year System Tag

Returns the current year number.

Syntax

```
$Year
```

Data Type

```
Integer (read only)
```

Example(s)

This script assigns the string "Welcome to xxxx" to the string Welcome where xxxx is the current year.

```
Welcome = "Welcome to " + StringFromIntg($Year,10)
```

$Month System Tag

Returns the current month number.

Syntax

```
$Month
```

Data Type

```
Integer (read only)
```

Example(s)

This script assigns the string "October" to the string MonthName if the current month is 10.

```
IF $Month==10 THEN
   MonthName="October";
ENDIF;
```

$Day System Tag

Returns the current day of the month.

Syntax

```
$Day
```

Data Type

```
Integer (read only)
```

Example(s)

This script assigns the string "It is a leap year!" to the string Msg2User if the current date is the February 29.

```
IF $Day==29 AND $Month==2 THEN
   Msg2Usr="It is a leap year!";
ENDIF;
```

$Hour System Tag

Returns the current hour of the day.

Syntax

```
$Hour
```

Data Type

```
Integer (read only)
```

Example(s)

This script checks if it is 8 PM and the backup has not run yet (expressed by the discrete tag BackupAlreadyRun), and if so, calls a QuickFunction script called RunBackup() and sets the BackupAlreadyRun flag to TRUE.

```
IF $Hour==20 AND BackupAlreadyRun==0 THEN
   CALL RunBackup();
   BackupAlreadyRun=1;
ENDIF;
```

$Minute System Tag

Returns the current minute of the hour.

Syntax

```
$Minute
```

Data Type

```
Integer (read only)
```

Example(s)

This script checks if it is 4:50 PM and if so, shows the window with the name Shift End.

```
IF $Minute==50 AND $Hour==16 THEN
   Show "Shift End";
ENDIF;
```

$Second System Tag

Returns the current second of the minute.

Syntax

```
$Second
```

Data Type

```
Integer (read only)
```

Example(s)

This script generates a sine wave function with an amplitude of 100 and a period of one minute.

```
100*Sin(6*$Second)
```

This script generates a series of 0's and 1's that change every second.

```
$second.00
```

$Msec System Tag

Returns the current milliseconds.

Note: By default the InTouch updates all tags every 1000 milliseconds. Because of this, the $Msec system tag seems not to change. If you increase the rate of update in the WindowViewer properties, you can see the $Msec tag updating.

Syntax

```
$Msec
```

Data Type

```
Integer (read only)
```

$Time System Tag

Returns the number of milliseconds that have passed since midnight in local time.

Syntax

```
$Time
```

Data Type

```
Integer (read only)
```

Example(s)

This script returns the number of seconds that have passed since midnight.

```
$Time/1000
```

$Date System Tag

Returns the number of whole days that have passed since January 01, 1970 in local time.

Syntax

```
$Date
```

Data Type

```
Integer (read only)
```

Example(s)

This script returns the current time.

```
StringFromTime(($Date*86400)+($Time/1000),3);
```

$DateTime System Tag

Returns the number of days (including fractions) that have passed since January 01, 1970 in local time..

Syntax

```
$DateTime
```

Data Type

```
Real (read only)
```

Example(s)

This script returns the current time.

```
StringFromTime($DateTime*86400,3);
```

DateTimeGMT() Function

Returns the number of days (including fractions of a day) that have passed since January 01, 1970 in Coordinated Universal Time (UTC).

Note: This function cannot be used in animation display links.

Syntax

```
result = DateTimeGMT();
```

Return Value

Number of days since January 01, 1970 in UTC. A literal real value.

Example(s)

This script returns the current date/time in UTC.

```
StringFromTime(DateTimeGMT() * 86400.0, 3);
```

Retrieve string date and time information

In a script, you can retrieve date and time information as strings. This is useful for showing date or time on the screen or when calculations on whole time/date strings are required.

You can use the following system tags and the script function.

Use	To
$DateString	Return the system date in short format.
$TimeString	Return the system time.
UTCDateTime	Return the UTC time and/or date and the time zone of the local computer.

$DateString System Tag

Returns the system date in short format as defined in the Regional Settings of the local operating system.

Syntax

```
$DateString
```

Data Type

```
String (read only)
```

Example(s)

This script may return 4/28/2006 depending on the short date format setting in the Regional Settings of the operating system.

```
$DateString
```

$TimeString System Tag

Returns the system time as defined in the Regional Settings of the local operating system.

Syntax

```
$TimeString
```

Data Type

```
String (read only)
```

Example(s)

This script may return 02:40:37 PM depending on the time format setting in the Regional Settings of the operating system.

```
$TimeString
```

UTCDateTime() Function

Returns the UTC time, the UTC date and time, or the local time zone.

Syntax

```
result = UTCDateTime (format)
```

Parameters

format

Determines what content is returned. A literal string value, message tagname, or string expression with the following possible values:

UTC_SHORT - the function returns the UTC time

UTC_LONG - the function returns the UTC date and time

UTC_LOCAL - the function returns the name of the time zone as set in the time zone settings of the local operating system

Any other values return the UTC date and time in default format (ddd mm dd hh:mm:ss yyyy).

Example(s)

At 09:24 AM Monday January 6th 2003 in the Pacific time zone, the UTCDateTime() function returns the following.

This script returns 17:24:05

```
UTCDateTime("UTC_SHORT")
```

This script returns 01/06/2003 17:24:05

```
UTCDateTime("UTC_LONG")
```

This script returns Pacific Standard Time -8:0: 1

```
UTCDateTime("UTC_LOCAL")
```

This script returns Mon Jan 06 17:24:05 2003.

```
UTCDateTime("Invalid")
```

Convert date and time information to strings

In a script, you can convert date and time information to strings for easier interpretation and display requirements. You can use the following functions.

Use	To
StringFromTime()	Convert a UTC timestamp to local time and to return as a time string.
wwStringFromTime()	Convert a local time timestamp to UTC time and returns it as a time string.
StringFromTimeLocal()	Convert a timestamp as a time string.

StringFromTime() Function

Converts a timestamp given in UTC time to local time and returns the result as a string. This function takes Daylight Saving Time into account.

Note: This function is equivalent to the StringFromGMTTimeToLocal() function.

Syntax

```
result = StringFromTime (timestamp, format)
```

Parameters

timestamp

The number of seconds that have passed since midnight of January 1, 1970 in the UTC time zone. A literal integer value, integer tagname, or integer expression.

format

Determines how the string result is shown. A literal integer value, integer tagname, or integer expression in the range from 1 to 5 with following meaning:

1 - Shows the date according to the format set in the Regional Settings of the local operating system

2 - Shows the time according to the format set in the Regional Settings of the local operating system

3 - Shows the date and time as a 24 character string (ddd mmm dd hh:mm:ss yyyy)

4 - Shows the day of the week in short form

5 - Shows the day of the week in long form

Example(s)

This example assumes that the time zone on the local node is Pacific Standard Time (PST, UTC-0800). The UTC time passed to the function is 12:00:00 AM on Friday, January 2, 1970. Since PST is 8 hours behind UTC, the

function returns the following results.

This script returns "1/1/70"

```
StringFromTime(86400,1)
```

This script returns "04:00:00 PM"

```
StringFromTime(86400,2)
```

This script returns "Thu Jan 01 16:00:00 1970"

```
StringFromTime(86400,3)
```

This script returns "Thu"

```
StringFromTime(86400,4)
```

This script returns "Thursday"

```
StringFromTime(86400,5)
```

wwStringFromTime() Function

Converts a timestamp given in local time to UTC time and returns the result as a string. This function takes Daylight Saving Time into account.

Syntax

```
result = wwStringFromTime (timestamp, format)
```

Parameters

timestamp

The number of seconds that have passed since midnight of January 1, 1970 in the local time zone. A literal integer value, integer tagname, or integer expression.

format

Determines how the string result is shown. A literal integer value, integer tagname, or integer expression in the range from 1 to 5 with following meaning:

1 - Shows the date according to the format set in the Regional Settings of the local operating system

2 - Shows the time according to the format set in the Regional Settings of the local operating system

3 - Shows the date and time as a 24 character string (ddd mmm dd hh:mm:ss yyyy)

4 - Shows the day of the week in short form

5 - Shows the day of the week in long form

Example(s)

This example assumes that the time zone on the local node is Pacific Standard Time (PST, UTC-0800). The local time passed to the function is 04:00:00 PM on Thursday, January 1, 1970. Since PST is 8 hours behind UTC, the function returns the following results.

This script returns "1/2/70"

```
wwStringFromTime(57600,1)
```

This script returns "12:00:00 AM"

```
wwStringFromTime(57600,2)
```

This script returns "Fri Jan 02 00:00:00 1970"

```
wwStringFromTime(57600,3)
```

This script returns "Fri"

```
wwStringFromTime(57600,4)
```

This script returns "Friday"

```
wwStringFromTime(57600,5)
```

StringFromTimeLocal() Function

Converts a timestamp to a time and returns the result as a string.

Syntax

```
result = StringFromTimeLocal (timestamp, format)
```

Parameters

timestamp

The number of seconds that have passed since midnight of January 1, 1970. A literal integer value, integer tagname, or integer expression.

format

Determines how the string result is shown. A literal integer value, integer tagname, or integer expression in the range from 1 to 5 with following meaning:

1 - Shows the date according to the format set in the Regional Settings of the local operating system

2 - Shows the time according to the format set in the Regional Settings of the local operating system

3 - Shows the date and time as a 24 character string (ddd mmm dd hh:mm:ss yyyy)

4 - Shows the day of the week in short form

5 - Shows the day of the week in long form

Example(s)

This script returns "1/2/70"

```
StringFromTimeLocal(86400,1)
```

This script returns "12:00:00 AM"

```
StringFromTimeLocal(86400,2)
```

This script returns "Fri Jan 02 00:00:00 1970"

```
StringFromTimeLocal(86400,3)
```

This script returns "Fri"

```
StringFromTimeLocal(86400,4)
```

This script returns "Friday"

```
StringFromTimeLocal(86400,5)
```

Check the daylight saving time status

In a script, you can check if daylight saving time is active by using the wwIsDaylightSaving() function.

wwIsDaylightSaving() Function

Returns whether daylight savings time is currently active.

Syntax

```
result = wwIsDaylightSaving()
```

Return Value

A Boolean value with following meaning:

0 - Daylight savings time is not active.

1 - Daylight savings time is active.

Interact with other applications

In a script, you can interact with other Windows applications by using various QuickScripts. For example, you can:

- Start an application, such as Notepad.
- Check an application title name.
- Check if a certain application is running.
- Activate a running application.
- Simulate keyboard strokes.
- Close, minimize or maximize an application window.
- Execute commands and exchange data with applications that support DDE.

Start a Windows application

In a script, you can start a Windows application using the StartApp command.

Syntax

```
StartApp appname;
```

Parameters

appname

Path and file name of the application you want to start. A literal string value, message tagname, or string expression.

Note: You need to know the path and file name of the application. If the application is in a directory that is part of the Windows PATH environment variable, you only need to pass the file name (without path).

Example(s)

This script starts Microsoft Calculator.

```
StartApp "calc"
```

Retrieve the application title of a running application

In a script, you can find the application title or Windows task list name of a specified running application by using the InfoAppTitle() function. This information is, for example, required by InTouch scripting for checking if the specified application is currently running or for activating it.

Note: This function will not return a value for Internet Explorer. As a workaround, use the Google Chrome browser.

InfoAppTitle() function

Returns the application title or Windows task list name of a specified application that is running.

Syntax

```
result = InfoAppTitle (appname)
```

Parameters

appname

Name of the application without the .exe extension. A literal string value, message tagname, or string expression.

Example(s)

This script returns "Calculator"

```
InfoAppTitle("calc")
```

This script returns "Microsoft Excel"

```
InfoAppTitle("excel")
```

Check if an application is running

In a script, you can check if a specific application is already running by using the InfoAppActive() function. You need to know the application title or Windows task list name first to be able to check if the specific application is running.

InfoAppActive() Function

Returns the running status of an application.

Syntax

```
result = InfoAppActive (apptitle)
```

Parameters

apptitle

The application title or Windows task list of the application for which you want to query the running status. A literal string value, message tagname, or string expression.

Return Value

A Boolean value indicating:

0 - The application is not running

1 - The application is running

Example(s)

This script queries for the application Notepad, and if it is already running, activates it. Otherwise it launches a new instance of Notepad. This way launching Notepad multiple times is avoided.

```
IF InfoAppActive(InfoAppTitle("Notepad"))==1
THEN
   ActivateApp InfoAppTitle( "Notepad" );
ELSE
   StartApp "Notepad";
ENDIF;
```

Activate a running Windows application

In a script, you can activate a running Windows application by using the ActivateApp() function. This brings the specified application to the foreground and gives it focus.

You need to do the following before activating a running Windows application:

- Find the application title or Windows task list name. See Retrieve the application title of a running application.
- Ensure the Windows application is running. See Check if an application is running.

ActivateApp Function

Activates an already running Windows application.

Syntax

```
ActivateApp apptitle;
```

Parameters

apptitle

The application title or Windows task list name of the running application you want to activate.

Example(s)

This script checks if a command prompt window is already open, and if so, activates it. Otherwise it starts the command prompt window.

```
IF InfoAppActive( InfoAppTitle("cmd")) == 1 THEN
   ActivateApp InfoAppTitle("cmd");
ELSE
   StartApp "cmd";
ENDIF;
```

Send simulated key strokes to an application

In a script, you can simulate pressing a sequence of keys on the keyboard. You can use this, for example, to:

- Enter data automatically in an open application.
- Control any application (including the InTouch HMI).

SendKeys Function

Simulates a sequence of key strokes.

Important: The SendKeys() function does not work on 64-bit versions of the Windows operating system.

Syntax

```
SendKeys sequence;
```

Parameters

sequence

The sequence of keys strokes to be simulated. A literal string value, message tagname, or string expression.
In addition to regular characters on the keyboard (such as alphanumeric characters) you can also specify control keys as a code:

```
   {BACKSPACE} - Simulates the Backspace key
   {BREAK} - Simulates the Break key
   {CAPSLOCK} - Simulates the Caps Lock key
   {DELETE} - Simulates the Delete key (or {DEL})
   {DOWN} - Simulates Arrow Down key
   {END} - Simulates the End key
```

```
{ENTER} - Simulates the Enter key (or ~)
{ESCAPE} - Simulates the ESC key (or {ESC})
{F1} .. {F12} - Simulate the F1 .. F12 keys
{HOME} - Simulates the Home key
{INSERT} - Simulates the Insert key
{LEFT} - Simulates the Arrow Left key
{NUMLOCK} - Simulates the Num Lock key
{PGDN} - Simulates the Page Down Key
{PGUP} - Simulates the Page Up key
{PRTSC} - Simulates the Print Screen key
{RIGHT} - Simulates the Arrow Right key
{TAB} - Simulates the Tab key
{UP} - Simulates the Up key
+ - Simulates the Shift key
use with parenthesis surrounding the key(s) you want to press in combination with the
Shift key.
^ - Simulates the Ctrl key
use with parenthesis surrounding the key(s) you want to press in combination with the
Ctrl key.
% - Simulates the Alt key
use with parenthesis surrounding the key(s) you want to press in combination with the
Alt key.
```

Remarks

Use the StartApp and/or ActivateApp() commands to activate another application before sending simulated keys strokes to it.

Example(s)

This script simulates pressing the B key.

```
SendKeys "b";
```

This script simulates pressing the key combination Ctrl and P, which can be used to initiate the Printing dialog box in another application.

```
SendKeys "^(p)";
```

This script simulates pressing F1 (which may open the help function), pressing the Tab key (which may place the cursor in a search field), entering HAL, and pressing the Enter key (which may initiate the search).

```
SendKeys "{F1}{TAB}HAL{ENTER}";
```

This script simulates pressing Ctrl, Shift and the key 1, which is the same as switching to WindowMaker. This powerful combination can be used for developing self-modifying (dynamic) InTouch HMI applications.

```
SendKeys "^(+(1))";
```

Close, minimize, or maximize a Windows application

In a script, you can close, minimize, or maximize another Windows application by using the WWControl() command.

You need to do the following before closing, minimizing or maximizing a Windows application:

- Find its application title or Windows task list name. See Retrieve the application title of a running application.

- Make sure that the Windows application is running. See Check if an application is running.

WWControl() Function

Restores, minimizes, maximizes, or closes a Windows application.

Syntax

```
WWControl (apptitle, control);
```

Parameters

apptitle

The application title or Windows task list name of the running application you want to restore, minimize, maximize or close. A literal string value, message tagname, or string expression.

control

Determines the action you want to take on the specified Windows application. A literal string value, message tagname, or string expression with following values:

Restore - activates and shows the application window

Minimize - activates and minimizes the application window

Maximize - activates and maximizes the application window

Close - closes the application

Example(s)

This script restores the calculator application if it is already running.

```
WWControl ("Calculator","Restore");
```

This script closes the WindowViewer.

```
WWControl (InfoAppTitle("View"),"Close");
```

Execute commands and exchange data using DDE

You can write a script to interact with applications that support DDE.

Use	To
WWExecute()	Send and execute commands.
WWRequest()	Read data from DDE items.
WWPoke()	Write data to DDE items.

WWExecute() Function

Sends a command to an application, executes it, and returns a status result. You can use it to have Excel to run a macro.

Important: The WWExecute() function does not work on 64-bit versions of the Windows operating system.

Syntax

```
Result = WWExecute (appname, topic, command)
```

Parameters

appname

The name of the application the command is sent to. A literal string value, message tagname, or string expression.

topic

The name of the topic within the application that the command is sent to. A literal string value, message tagname, or string expression.

command

The command to be sent. A literal string value, message tagname, or string expression.

Return Value

A value of-1, 0, or 1 indicating the following:

-1 - command not executed successfully. Possible causes are the application not running, the topic does not exist or the command contains an error.
0 - command not executed successfully because the application is busy.
1 - command executed successfully.

Example(s)

This script instructs Microsoft Excel to execute the macro Macro1 by sending the command [Run("Macro1",0)] to Excel.

```
Macro="Macro1";
Command="[Run(" + StringChar(34) + Macro + StringChar(34) + ",0)]";
WWExecute("excel","system",Command);
```

WWRequest() Function

Reads data from an item of an application. You can use it, for example, to read the value of a spreadsheet cell in Microsoft Excel.

Important: The WWRequest() function does not work on 64-bit versions of the Windows operating system.

Syntax

```
Result = WWRequest(appname, topic, item, messagetag)
```

Parameters

appname

The name of the application. A literal string value, message tagname, or string expression.

topic

The name of the topic within the application. A literal string value, message tagname, or string expression.

item

The name of the item belonging to the topic and application. A literal string value, message tagname, or string expression.

messagetag

A message tagname to retrieve the value of the item. The message tagname value can be converted into an integer or real value by using the StringToIntg() or StringToReal() functions.

Return Value

A value of -1, 0, or 1 indicating the following:

-1 - data not read successfully. Possible causes are the application not running or the topic or item do not exist.
0 - data not read successfully because the application is busy.
1 - data read successfully.

Example(s)

This script reads the value contained in Microsoft Excel book Book1.xls, sheet Sheet1 in Row 1, Column 1 to the message tagname MTag and puts the value in the real tagname CellValue.

```
Result = WWRequest("excel","[Book1.xls]sheet1", "r1c1",Mtag);
CellValue=StringToReal(MTag);
```

If you are using a non-English operating system, you may need to use the StringReplace() function to change the contents of *MTag* before converting it to a different data type. For example, for operating systems that use a comma as a decimal separator, you may need to replace all commas with decimal dots in *MTag* before converting it to a real data type.

WWPoke() Function

Writes data to an item of an application. You can use it, for example, to write the value into a spreadsheet cell in Excel.

Important: The WWPoke() function does not work on 64-bit versions of the Windows operating system.

Syntax

```
result = WWPoke (appname, topic, item, string)
```

Parameters

appname

The name of the application. A literal string value, message tagname, or string expression.

topic

The name of the topic within the application. A literal string value, message tagname, or string expression.

item

The item name belonging to the topic and application. A literal string value, message tagname, or string expression.

string

The value to be written. A literal string value, message tagname, or string expression. You can use the StringFromIntg(), StringFromReal() or Text() functions to convert the value of an integer or real tagname to a message tagname.

Return Value

A value of-1, 0, or 1 indicating the following:

-1 - data not written successfully. Possible causes are the application not running or the topic or item do not exist.
0 - data not written successfully because the application is busy.
1 - data written successfully.

Remarks

Do not use the WWPoke() or WWRequest() function to read and write data between InTouch applications on different nodes or sessions. To read and write data between InTouch applications, use Access Names instead. See Set up access names.

Example(s)

This script puts the value of the real tagname CellValue in the message tagname Mtag and writes the value to the spreadsheet cell Row 1, Column 1 of sheet Sheet1 in Microsoft Excel book Book1.xls.

```
MTag = Text(CellValue,"0");
Result = WWPoke("excel","[Book1.xls]sheet1", "r1c1",Mtag);
```

Work with files

You can write a script using various file management and access operations.

Use	To
FileCopy()	Copy files.
FileDelete()	Delete files.

Use	To
FileMove()	Move files.
FileReadFields(), FileWriteFields()	Read/write csv data.
FileReadMessage(), FileWriteMessage()	Read/write text data.

Manage files

In a script, you can copy, delete or move files.

FileCopy() function

Copies a source file to a destination file and returns a status result. This function may take a longer time to execute and is executed in multiple stages:

1. FileCopy() function is called and an immediate result is returned, indicating success or failure of the file copy initialization.
2. FileCopy() function executes the copy procedure in the background, and InTouch scripting continues execution while the file copying is in progress. You can monitor the file copying progress with an integer tag.
3. FileCopy() function returns a file copy result, indicating success or failure of the file copy procedure.

If the destination folder is not available (i.e. another computer on the network), the function waits for up to 10 seconds to time out, and then posts a message in the Logger.

Note: Do not use the FileCopy() function in asynchronous QuickFunctions.

Syntax

```
result = FileCopy (sourcefile, destfile, progresstag)
```

Parameters

sourcefile

Full path and file name of the file to be copied. A literal string value, message tagname, or string expression. You can use the wildcard characters (* and ?) in this parameter to copy just files matching a specified criteria. The path name can also be a UNC path name.

destfile

Full path and file name (or just path name) of the destination. A literal string value, message tagname, or string expression. The path name can also be a UNC path.

progresstag

Name of an integer tag enclosed in double quotes that will contain a value indicating the file copy progress. A literal string value, message tagname (such as a message tag containing the value "IntTag.Name") or string expression. The values have following meaning:
0 - FileCopy() procedure is still in progress.

1 - FileCopy() procedure has completed successfully.
-1 - FileCopy() procedure completed with errors.

Return value

A value of -1, 0, or 1 indicating the following:

1 - FileCopy() function successfully called.
0 - Error when calling the FileCopy() function because another FileCopy() procedure is already in progress.
-1 - Error when calling the FileCopy() function because of a non-existent source file or the destination is read only.

Example(s)

This script copies the file c:\MyData\output.log to the directory d:\archive and renames the file to output.txt. The progress of the file copy is written to the integer tag Monitor.

```
Status=FileCopy("c:\MyData\output.log","d:\archive\output.txt","Monitor");
```

This script copies all files with file ending .txt in the c:\ root directory to the destination directory c:\Backup.

```
Status=FileCopy("c:\*.txt", "c:\Backup", "Monitor");
```

This script copies a file whose full path and file name is contained in the message tag LogFile to the destination directory c:\results\ and renames it to logxxx.txt where xxx is a timestamp.

```
Status=FileCopy(LogFile, "c:\results\log" + $DateString + $TimeString + ".txt", "Monitor");
```

FileDelete() function

Deletes an individual file.

Syntax

```
result = FileDelete (filename)
```

Parameters

filename

The path name and file name of the file to delete. A literal string value, message tagname, or string expression. UNC path names are supported.

Remarks

Do not use the wildcard characters (* and ?) with the FileDelete() function and do not use the FileDelete() function in asynchronous QuickFunctions.

The FileDelete() function does not delete directories.

Return value

A value indicating success or failure of the file deletion:

1 - file is deleted successfully
0 - file is not deleted successfully. Possible causes are attempts to delete a read only or a non-existent file.

Example(s)

This script deletes the file c:\Data.txt and returns 1 if the file was found and deleted successfully.

```
Status=FileDelete("c:\Data.txt");
```

FileMove() function

Moves a source file to a destination file and returns a status result. It can be also used to rename a file. This function may take a longer time to execute and executes in multiple stages:

1. FileMove() function is called and an immediate result is returned, indicating success or failure of the file move initialization.
2. FileMove() function executes the move procedure in the background, InTouch scripting continues execution while the file moving is in progress. You can monitor the file moving progress with an integer tag.
3. FileMove() function returns a file move result, indicating success or failure of the file moving procedure.

Do not use the FileMove() function in asynchronous QuickFunctions.

Syntax

```
result = FileMove (sourcefile, destfile, progresstag)
```

Parameters

sourcefile

Full path and file name of the file to be moved. A literal string value, message tagname, or string expression. You can use the wildcard characters (* and ?) in this parameter to move just files matching a specified criteria. The path name can also be a UNC path name.

destfile

Full path and file name (or just path name) of the destination. A literal string value, message tagname, or string expression. The path name can also be a UNC path.

progresstag

Name of an integer tag enclosed in double quotes that will contain a value indicating the file moving progress. A literal string value, message tagname (such as a message tag containing the value "IntTag") or string expression. The values have following meaning:

0 - FileMove() procedure is still in progress

1 - FileMove() procedure has completed successfully

-1 - FileMove() procedure completed with errors

Return value

A value of-1, 0, or 1 indicating the following:

1 - FileMove() function successfully called
0 - Error when calling the FileMove() function because another FileMove() procedure is already in progress
-1 - Error when calling the FileMove() function. Possible errors are attempts to move a non-existent file.

Example(s)

This script moves the file c:\MyData\output.log to the directory d:\archive and renames the file to output.txt. The progress of the file moving is written to the integer tag Monitor.

```
Status=FileMove("c:\MyData\output.log","d:\archive\output.txt","Monitor");
```

This script moves all files with file ending .txt in the c:\ root directory to the destination directory c:\Backup.

```
Status=FileMove("c:\*.txt", "c:\Backup", "Monitor");
```

This script moves a file whose full path and file name is contained in the message tag LogFile to the destination directory c:\results\ and renames it to logxxx.txt where xxx is a timestamp.

```
Status=FileMove(LogFile, "c:\results\log" + $DateString + $TimeString + ".txt", "Monitor");
```

Read and write CSV data

You can write a script to read and write data contained in a csv (comma separated variable) file from and to a series of tagnames by using the function FileReadFields() and FileWriteFields().

The functions FileReadFields() and FileWriteFields() support only the comma as a delimiter.

FileReadFields() function

Reads the values contained in a csv file into a series of tagnames. You can use this function to load a set of tagname values.

Commas are the only supported delimiter.

This function can only be used for synchronous calls.

Syntax

```
[result = ] FileReadFields (filename, offset, starttag, numberoffields)
```

Parameters

filename

Name of the csv file to read the data from. A literal string value, a message tagname or a string expression.

offset

Location (in bytes) in the file to start reading. A literal integer value, integer tagname, or integer expression.

starttag

Name of the first tagname that receives the first read data item. The tagname must be enclosed with double

quotes and end in a number, such as "MyTag1". A literal string value, message tagname (such as a message tagname containing the value "MyTag1"), or a string expression.

numberoffields

Number of data items to read from the csv file. A literal integer value, integer tagname, or integer expression. The first data item is read into the tagname defined in the starttag parameter, subsequent data items into tagnames with the incremented numeral suffix of the starttag parameter (MyTag1, MyTag2, MyTag3, ...).

Return value

Optional new file offset (in byte) after reading the data. This can be used to read the next set of data.

Example(s)

This script reads the values "Flour" to RecipeTag1, 27.23 to RecipeTag2, 14 to RecipeTag3, and 1 to RecipeTag4, and returns the new file offset—if the csv file c:\set.csv contains the following data: Flour, 27.23,14,1 and if the following tags are defined: RecipeTag1:message, RecipeTag2:real, Recipe3:integer, RecipeTag4:discrete.

```
FileReadFields("c:\set.csv",0,"RecipeTag1",4);
```

FileWriteFields() function

Writes the values contained in a series of tagnames to a csv file. You can use this function to save a set of tagname values.

Commas are the only supported delimiter.

Syntax

```
[result = ] FileWriteFields (filename, offset, starttag, numberoffields)
```

Parameters

filename

Name of the csv file to write the data to. A new file is created if it does not previously exist. A literal string value, a message tagname, or a string expression.

offset

Location (in bytes) in the file to start writing to. Use -1 to write to the end of the file (append). A literal integer value, integer tagname, or integer expression.

starttag

Name of the first tagname that contains the first data item to be written. The tagname must be enclosed with double quotes and end in a number, such as "MyTag1". A literal string value, message tagname (such as a message tagname containing the value "MyTag1") or a string expression.

numberoffields

Number of data items to write to the csv file. A literal integer value, integer tagname, or integer expression. The first data item is written from the tagname defined in the starttag parameter to the file, subsequent data items from tagnames with the incremented numeral suffix of the starttag parameter (MyTag1, MyTag2, MyTag3, ...).

Return value

Optional new file offset (in byte) after writing the data. This can be used to write the next set of data.

Example(s)

A series of InTouch tags is defined as follows:

Tagname	Data Type	Value
RecipeTag1	Message	Flour
RecipeTag2	Real	27.23
RecipeTag3	Integer	14
RecipeTag4	Discrete	1

This script writes the values contained in RecipeTag1 to RecipeTag4 to the csv file c:\set.csv.

```
FileWriteFields("c:\set.csv",0,"RecipeTag1",4);
```

So that the file c:\set.csv will contain the following data:

```
Flour,27.23,14,1
```

Read and write text data

You can write a script to read and write text data to and from a file by using the FileReadMessage() and FileWriteMessage() functions. You can either read/write a specified number of bytes or an entire line of text (demarcated by a line feed character).

FileReadMessage() function

Reads a specified number of bytes (or one line) of string data from a file.

Syntax

```
[result = ] FileReadMessage (filename, offset, messagetag, charstoread)
```

Parameters

filename

Name of the file to read the data from. A literal string value, a message tagname, or a string expression.

offset

Location (in bytes) in the file to start reading from. A literal integer value, integer tagname, or integer expression.

messagetag

Message tagname that receives the first line or number of bytes from the file. Enclose the tagname with double quotes when using the function within the Industrial Graphics Editor Script Editor.

charstoread

Number of bytes to read from the file. Set it to 0 to read until the next line feed (LF) character. A literal integer value, integer tagname, or integer expression.

Return value

Contains the new byte position after the read. You can use this for subsequent reads from the file.

Example(s)

This script reads the first line of data in the file c:\Data\File.txt to the message tagname MsgTag.

```
FileReadMessage ("c:\Data\File.txt",0,MsgTag, 0);
FileReadMessage ("c:\Data\File.txt",0,"InTouch:MsgTag", 0);
```

FileWriteMessage() function

Writes a specified number of bytes (or one line) of string data to a file.

Syntax

```
[result = ] FileWriteMessage (filename, offset, messagetag, linefeed)
```

Parameters

filename

Name of the file to write the data to. A literal string value, a message tagname, or a string expression.

offset

Location (in bytes) in the file to start writing to. Set it to -1 to write data to the end of the file (append). A literal integer value, integer tagname, or integer expression.

messagetag

Message tagname that contains the data to be written to the file.

linefeed

Specifies whether to write a line feed (LF) character after writing the data to the file. Set to 1 to write a line feed character; otherwise, set it to 0. A literal Boolean value, discrete tagname, or Boolean expression.

Return value

Contains the new byte position after the write. You can use this for subsequent writes to the file.

Example(s)

This script writes the value of a message tagname MsgTag to the end of the file c:\Data\File.txt.

```
FileWriteMessage("c:\Data\File.txt",-1,MsgTag,1);
```

Retrieve system-related information

In a script, you can retrieve system-related information using the following QuickFunctions.

Use	To
GetNodeName()	Retrieve the node name of the computer.
InfoDisk()	Retrieve disk space information.
InfoFile()	Retrieve information about a file.

Retrieve the node name of the computer

In a script, you can retrieve the node name of the computer with the GetNodeName() function. This can be used, for example, to keep your InTouch applications dynamic when working with access names.

GetNodeName() function

Returns the node name of the computer.

Syntax

```
GetNodeName (messagetag, nodenum);
```

Parameters

messagetag

Message tagname that will contain the node name. Enclose the tagname with double quotes when using the function within the Industrial Graphics Editor Script Editor.

nodenum

Number of characters to retrieve from the node name. A literal integer value, integer tagname, or integer expression in the range of 0 to 131.

Example(s)

This script retrieves the node name and assigns it to the NodeName message tagname.

```
GetNodeName(NodeName,131);
GetNodeName("InTouch:NodeName",131);
```

Retrieve disk space information

In a script, you can retrieve disk space information by using the InfoDisk() function. You can retrieve:

- The total size of the disk drive (in bytes or kilobytes).

- The available free space on the disk drive (in bytes or kilobytes).

You can also determine when or how often the information updates (in an animation link) by specifying a trigger tag.

InfoDisk() function

Returns either the total or free space on a local or network disk drive.

Syntax

```
result = InfoDisk (drive, infotype, trigger);
```

Parameters

drive

The drive letter for which you want to retrieve information. Only the first character of a string is used. A literal string value, message tagname, string expression.

infotype

Specifies the information type. A literal integer value, integer tagname, or integer expression with following possible values:

1 - function returns total size of disk drive (in bytes)

2 - function returns free space of disk drive (in bytes)

3 - function returns total size of disk drive (in kilobytes)

4 - function returns free space of disk drive (in kilobytes)

trigger

A tagname (or expression) that acts as a trigger to recalculate the disk information. If the trigger value changes the disk information is recalculated. A discrete or analog taname, or a discrete or analog expression.

Remarks

The trigger tag only has meaning when the InfoDisk() function is used in an animation display link. If this function is used in a script, you can specify any literal numeric value, analog tagname, or numeric expression.

Example(s)

Use this script in an animation display link to show the free space of disk drive C and update the information every minute.

```
InfoDisk("C", 4, $Minute)
```

Retrieve information on a file or directory

In a script, you can retrieve information on a specific file or directory by using the InfoFile() function. By using different parameters you can find:

- If the file exists.
- If the specified file name is actually a directory.
- The size (in bytes) of the file.
- The timestamp of the file or directory.
- The number of files that match a wildcard search.

InfoFile() function

Returns various information on a file or directory.

Syntax

```
result = InfoFile (filename, infotype, trigger)
```

Parameters

filename

The full file name or directory name you want to retrieve information about. A literal string value, message tagname, or string expression. Can also include wildcard characters, such as "*" and "?".

infotype

The type of information you want to retrieve about the specified file or directory. A literal integer value, integer tagname, or integer expression with following values and meaning:

1 - Existence. The InfoFile() function returns 1 if the file exists, 2 if the file is a directory and 0 if the file or directory does not exist.

2 - Size. The InfoFile() function returns the file size in bytes.

3 - Creation timestamp. The InfoFile() function returns the time stamp as seconds that have passed since midnight January 1, 1970. Use the StringFromTimeLocal() function to convert this value to a message timestamp.

4 - Wildcard Search Match. The InfoFile() function returns the number of files that match a specified wildcard search.

trigger

A tagname (or expression) that acts as a trigger to recalculate the file information. If the trigger value changes, the file information is recalculated. A discrete or analog taname, or a discrete or analog expression.

Remarks

The trigger tag only has meaning when the InfoFile() function is used in an animation display link. If this function is used in a script, you can specify any literal numeric value, analog tagname, or numeric expression.

Example(s)

This script returns 1 if the file c:\data\log.txt exists.

```
InfoFile("c:\data\log.txt",1,$minute)
```

This script returns 14223 if the file c:\data\log.txt has a file size of 14223 bytes.

```
InfoFile("c:\data\log.txt",2,$minute)
```

This script returns 1138245266 if the file c:\data\log.txt was created on January 26, 2006 at 11:14:26 AM.

```
InfoFile("c:\data\log.txt",3,$minute)
```

This script returns 14 if there are 14 files in the directory c:\data\ that have a txt ending.

```
InfoFile("c:\data\*.txt",4,$minute)
```

Retrieve InTouch-related information

In a script, you can retrieve InTouch-related information using these functions.

Use	To
InfoInTouchAppDir()	Get information on the directory of the InTouch application you are developing.
InTouchVersion()	Get information on the InTouch version.

Retrieve the name of the InTouch application directory

In a script, you can retrieve the name of the directory that your InTouch application is running in with the InfoInTouchAppDir() function. This function is useful to locate any external files that you include to ship with your InTouch application.

InfoInTouchAppDir() function

Returns the current InTouch application directory.

Syntax

```
result = InfoInTouchAppDir();
```

Return value

A message tagname to contain the directory of the currently running InTouch application.

Remarks

The application directory name may be truncated when passed to a message tagname or shown in an animation link due to the 131 characters limitation.

Example(s)

This script may return c:\documents and settings\user1\my documents\my intouch applications\packaging.

```
InfoInTouchAppDir()
```

Retrieve the InTouch version

In a script, you can retrieve the version number of the InTouch application you are currently running by using the InTouchVersion() function.

InTouchVersion() function

Returns the complete InTouch version number or just parts of it.

Syntax

```
result = InTouchVersion (infotype);
```

Parameters

infotype

Specifies how the version information is returned. A literal integer value, integer tagname, or integer expression with the following meaning:

0- function returns the whole version number

1- function returns just the major version number

2- function returns just the minor version number

3- function returns just the patch level

4- function returns just the build level

Example(s)

Function	Possible result
InTouchVersion(0)	10.5.1626.0521.0045.0012
InTouchVersion(1)	10
InTouchVersion(2)	5
InTouchVersion(3)	0
InTouchVersion(4)	1626

Security-related scripting

You can add and manage security within your InTouch application with various QuickScript functions and system tags. For more information about security functions, see Secure InTouch in *AVEVA™ InTouch HMI Management*.

Log on and log off

You can use the following functions and system tags to log on and log off.

Use	To
AttemptInvisibleLogon()	Log on a user by supplying authentication data in the parameters.
LogonCurrentUser()	Log on the currently logged on Windows user (if authentication mode is "OS").
PostLogonDialog()	Show the **Logon** dialog box.
Logoff()	Log off the current user.
$PasswordEntered	Set a password.
$OperatorEntered	Set a valid user name.
$OperatorDomainEntered	Set a valid user domain name (if authentication mode is "OS").

For more information about security functions, see Secure InTouch in *AVEVA™ InTouch HMI Management*.

Change and set password

You can use the following functions and system tags to change password:

Use	To
ChangePassword()	Call the **Change Password** dialog box for the currently logged on user.
$ChangePassword	Call the **Change Password** dialog box for the currently logged on user.

For more information about security functions, see Secure InTouch in *AVEVA™ InTouch HMI Management*.

Specify and configure users

You can use the following system tag to specify and configure users.

Use	To
$ConfigureUsers	Call the **Configure Users** dialog box.

For more information about security functions, see see Secure InTouch in *AVEVA™ InTouch HMI Management*.

Manage security and other information

You can use the following system tags and functions to manage security.

Use	To
$AccessLevel	Retrieve the access level of the currently logged in user.
AddPermission()	Assign access levels to a certain user group (local/ domain).
GetAccountStatus()	Retrieve account information (password expiration, lock out, disable flags).
$InactivityTimeout	Indicate the time that elapses before the user is automatically logged off.
$InactivityWarning	Indicate the time for the time-out warning.
InvisibleVerifyCredentials()	Retrieve InTouch access level information of an OS user.
IsAssignedRole()	See if the currently logged on user has the specified user role.
QueryGroupMembership()	See if the currently logged on user is a member of a specified user role.

For more information about security functions, see Secure InTouch in *AVEVA™ InTouch HMI Management*.

Miscellaneous scripts

InTouch scripting supports sound output so that you can associate human machine interaction with sounds. InTouch scripting also supports getting and setting properties of Wizards.

Play sound files from an InTouch application

In a script, you can associate events and conditions with specific sounds. For example, you could associate a warning dialog box or a critical condition with a warning sound.

PlaySound() Function

Plays a sound from a wave file or a Windows default sound.

Syntax

```
Playsound (soundname, flag)
```

Parameters

soundname

The name of the sound or wave file. A literal string value, message tagname, or string expression. If the sound is defined as a name, it must be defined in the Win.ini file under the [Sounds] section, for example MC="c:\test.wav"

flag

Specifies how the sound is played. A literal integer value, integer tagname, or integer expression with the following meanings:
0 - Play sound one time synchronously (script execution waits until sound has finished playing).
1 - Play sound one time asynchronously (script execution does not wait until sound has finished playing).
9 - Play sound continuously (until the PlaySound() function is called again).

Example(s)

This script plays the sound of the file c:\welcome.wav one time and holds script execution until it has finished playing.

```
PlaySound("c:\welcome.wav",0);
```

This script plays the sound Alert continuously. In the win.ini file [Sounds] section you need to associate the sound name Alert with a sound file, such as:

```
Alert=c:\alert.wav.
PlaySound("Alert",9);
```

Get and set properties of wizards

Some wizards such as the Distributed Alarm Object and Windows Controls contain set or read properties. These properties could be values in a text box or the check status of a checkbox.

In a script, you access these properties through the following functions.

Use	To
SetPropertyD(), GetPropertyD()	Set or read discrete properties.
SetPropertyI(), GetPropertyI()	Set or read integer properties.
SetPropertyM(), GetPropertyM()	Set or read message properties.

See the wizard description for a list of supported properties.

Here is how to set and read these properties in a generic way.

GetPropertyD() Function

Reads a discrete property in a wizard and returns a success code.

Syntax

```
result = GetPropertyD (controlname.property, dtag)
```

Parameters

controlname

The name of a wizard that supports properties. A literal string value, message tagname, or string expression.

property

The discrete property of the wizard that is to be read. Together with controlname can be a literal string value, message tagname, or string expression.

dtag

The discrete tagname that will receive the discrete property value.

Return Value

An integer error code. For more information about the error codes, see Understand windows controls error messages.

Example(s)

With a checkbox wizard Checkbox1 and a discrete tagname dtag you can check the visibility of the checkbox with the following script function:

```
result=GetPropertyD("Checkbox1.visible",dtag);
```

This script sets dtag to 1, if the checkbox wizard is visible; otherwise, it sets dtag to 0.

SetPropertyD() Function

Sets a discrete property in a wizard and returns a success code.

Syntax

```
result = SetPropertyD(controlname.property, Boolean)
```

Parameters

controlname

The name of a wizard that supports properties. A literal string value, message tagname, or string expression.

property

The discrete property of the wizard that is to be set. Together with controlname can be a literal string value,

message tagname, or string expression.

Boolean

The Boolean value to pass to the wizard property. A literal Boolean value, discrete tagname or Boolean expression.

Return Value

An integer error code. For more information about the error codes, see Understand windows controls error messages.

Example(s)

With a checkbox wizard Checkbox1 and a discrete tagname dtag you can control the visibility of the checkbox with the following script function:

```
result=SetPropertyD("Checkbox1.visible",dtag);
```

If you set dtag to 0 and call the script function above, the checkbox wizard becomes invisible.

GetPropertyI() Function

Reads an integer in a wizard and returns a success code.

Syntax

```
result = GetPropertyI (controlname.property, itag)
```

Parameters

controlname

The name of a wizard that supports properties. A literal string value, message tagname, or string expression.

property

The integer property of the wizard that is to be read. Together with controlname can be a literal string value, message tagname, or string expression.

itag

The integer tagname that will receive the integer property value.

Return Value

An integer error code. For more information about the error codes, see Understand windows controls error messages.

Example(s)

With a radio button wizard Radiobutton1 and an integer tagname itag you can check the currently selected item in the radio button group with the following script function:

```
result=GetPropertyI("Radiobutton1.value",itag);
```

This script sets itag to 1 (2, 3, ...) , if the first (second, third, ...) radio button is selected.

SetPropertyI() Function

Sets an integer property in a wizard and returns a success code.

Syntax

```
result = SetPropertyI (controlname.property, integer)
```

Parameters

controlname

The name of a wizard that supports properties. A literal string value, message tagname, or string expression.

property

The integer property of the wizard that is to be set. Together with controlname can be a literal string value, message tagname, or string expression.

integer

The integer value to pass to the wizard property. A literal integer value, integer tagname, or integer expression.

Return Value

An integer error code. For more information about the error codes, see Understand windows controls error messages.

Example(s)

With a radio button wizard Radiobutton1 you can set the 2nd radio button with the following script function:

```
result=SetPropertyI("Radiobutton1.value",2);
```

GetPropertyM() Function

Reads a message property in a wizard and returns a success code.

Syntax

```
result = GetPropertyM (controlname.property, mtag)
```

Parameters

controlname

The name of a wizard that supports properties. A literal string value, message tagname, or string expression.

property

The message property of the wizard that is to be read. Together with controlname can be a literal string value, message tagname, or string expression.

mtag

The message tagname that will receive the message property value.

Return Value

An integer error code. For more information about the error codes, see Understand windows controls error messages.

Example(s)

With a checkbox wizard Checkbox1 and a message tagname mtag you can check the caption of the checkbox with the following script function:

```
result=GetPropertyM("Checkbox1.caption",mtag);
```

This script sets mtag to the caption of the checkbox.

SetPropertyM() Function

Sets a message property in a wizard and returns a success code.

Syntax

```
result = SetPropertyM (controlname.property, message)
```

Parameters

controlname

The name of a wizard that supports properties. A literal string value, message tagname, or string expression.

property

The message property of the wizard that is to be set. Together with controlname can be a literal string value, message tagname, or string expression.

message

The message value to pass to the wizard property. A literal string value, message tagname, or string expression.

Return Value

An integer error code. For more information about the error codes, see Understand windows controls error messages.

Example(s)

With a checkbox wizard Checkbox1 you can set the caption of the checkbox wizard dynamically with the following script function:

```
result=SetPropertyM("Checkbox1.caption","Start Engine 1");
```

This script sets the caption of the checkbox Checkbox1 to "Start Engine 1".

Script with OLE objects

You can use OLE objects to extend the functionality of an InTouch HMI application. With OLE objects, you can:

- Create popup dialog boxes for the operator interface.
- Access operating system functions, such as the Control Panel.
- Make data from the Manufacturing Execution Module available for processing within the InTouch HMI. See the Manufacturing Execution Module documentation.

Create, validate, and release OLE objects

You can create and validate OLE objects for use in InTouch scripts. After using an OLE object you can release it to free up memory.

Use the following functions to create, validate, and release OLE objects.

- OLE_CreateObject() Function
- OLE_IsObjectValid() Function
- OLE_ReleaseObject() Function

OLE_CreateObject() Function

Before you can reference an OLE object in a script, you must create it. When you do this you receive a pointer that references the OLE object.

In a script, you can create an OLE object and assign a pointer by using the OLE_CreateObject() function.

Syntax

```
OLE_CreateObject(%pointer, classname);
```

Parameters

%pointer

The name of your choice for the pointer to the OLE object. It can contain alphanumeric characters (A-Z, 0-9) and underscore. It is case-insensitive.

classname

The name of the OLE class. The class name is case-sensitive. A literal string value, message tagname, or string expression.

Remarks

If you use the same object name to create another object, the object is updated to reference the new OLE class. It is released from the old OLE class.

Example(s)

This script creates an OLE object called %WShell that references the class Wscript.Shell.

```
OLE_CreateObject(%WShell, "Wscript.Shell");
```

OLE_IsObjectValid() Function

In a script, you can verify that an OLE object is valid by using the OLE_IsObjectValid() function. This is not a required step for working with OLE objects, but it is recommended to make sure that you do not come across problems when working with OLE objects.

Syntax

```
result = OLE_IsObjectValid(%pointer)
```

Arguments

%pointer

The pointer referencing an OLE object that is to be tested.

result

A Boolean value indicating the following:

0 - The OLE object the pointer is referencing is invalid.

1 - The OLE object the pointer is referencing is valid.

Example(s)

This script creates an OLE object based on the Wscript.Shell class and creates a pointer %WS to reference it. isvalid is a discrete tag that is TRUE if the OLE object is created successfully. Otherwise it is FALSE.

```
OLE_CreateObject(%WS, "Wscript.Shell");
isvalid = OLE_IsObjectValid(%WS);
```

OLE_ReleaseObject() Function

After you have used an OLE object in a script, you can release it and delete its pointer to free up system resources. After you release an OLE object you cannot use its pointer to access properties and methods of the associated OLE class.

Syntax

```
OLE_ReleaseObject(%pointer);
```

Arguments

%pointer

Name of the pointer that references the OLE Object. It can contain alphanumeric characters (A-Z, 0-9) and underscore. It is case-insensitive.

Example(s)

This script releases the OLE object associated with the pointer %WShell and deletes the pointer %WShell.

```
OLE_ReleaseObject(%WShell);
```

Use OLE object properties and methods

In a script, you can use pointers to read and write values from and to OLE properties. You can also use the pointer to call OLE methods. The properties and methods available depend on the OLE object.

Access the properties of an OLE object

In a script, you can access the properties of an OLE object as you would in most programming languages. Properties are usually identified by using the dot "." operator.

Note: When you use OLE object properties in a script, make sure that their references do not exceed 98 characters, including leading "%". Keep OLE pointer names as short as possible.

Read an OLE object property

In a script, you can read an OLE object property by assigning the property to a tag. You cannot use a direct reference to an OLE object property in an animation display link.

Syntax

```
tagname = %pointer.property;
```

Arguments

%pointer

The pointer that references the OLE object. Must be created with OLE_CreateObject() function or assigned to another pointer before reading a property.

property

The name of the property to be read.

tagname

The tag to write the value to.

Example(s)

This script creates an OLE object based on the System.Random OLE class, creates a pointer %SR to reference it, and assigns the value of the .NextDouble property of the Math.Random OLE object to a real tagname randtag.

At run time the real tagname Randtag receives a random double float value between 0 and 1.

```
OLE_CreateObject(%SR,"System.Random");
randtag = %SR.NextDouble;
```

Write to an OLE object property

In a script, you can write a value to an OLE object property by assigning a value to the property.

Syntax

```
%pointer.property = value;
```

Arguments

%pointer

The pointer that references the OLE object. Must be created with OLE_CreateObject() function or assigned to another pointer before writing to a property.

property

The name of the property to be written to.

value

The value to be written to the property. It can be a literal value, tagname or expression. Writing to an OLE property from an animation input link directly is not supported.

Call methods of an OLE object

In a script, you can call OLE object methods.

Syntax

```
%pointer.method(parameters);
```

Arguments

%pointer

The pointer that references the OLE object. Must be created with OLE_CreateObject() function or assigned to another pointer before calling a method.

method

The name of the method that is part of the OLE object.

parameters

A list of parameters to pass to the method. These parameters must be separated by comma. Literal values, tagnames or expressions.

Example(s)

This script creates an OLE object based on the OLE class Shell.Application, creates a pointer %sa to the OLE object and calls the method .MinimizeAll(). This method minimizes all windows on your desktop.

```
OLE_CreateObject(%SA,"Shell.Application");
%SA.MinimizeAll();
```

Note: Optional parameters are not allowed in OLE InTouch HMI scripting. All parameters must be specified.

Assign multiple pointers to the same OLE object

In a script, you can assign multiple pointers to the same OLE object by using the equals sign.

Syntax

```
%newpointer = %pointer
```

Arguments

%pointer

The name of the pointer that already references a created OLE object.

%newpointer

The name of a new pointer that should reference the same OLE object. It can contain alphanumeric characters (A-Z, 0-9) and underscore. It is case-insensitive.

Example(s)

This script creates an OLE object based on the Wscript.Shell class and creates a pointer %WS to reference it. The pointer %WS2 when set to %WS points to the same OLE object. It can be used to read from or write to properties and call methods of the same OLE object.

```
OLE_CreateObject(%WS,"Wscript.Shell");
%WS2=%WS;
```

Note: You can use message tagnames in connection with pointers. If you assign a message tagname to a pointer, it can get an ID value. You can use it to create more pointers to the same OLE object.

Troubleshoot OLE errors

In a script, you can use OLE functions to troubleshoot OLE errors.

Function	Description
OLE_GetLastObjectError() Function	Get the error number of the last OLE error.
OLE_GetLastObjectErrorMessage() Function	Get information on the last OLE error.
OLE_ResetObjectError() Function	Reset the last error.
OLE_ShowMessageOnObjectError() Function	Show or hide the OLE error message dialog box.
OLE_IncrementOnObjectError() Function	Count the number of OLE errors with an InTouch HMI tagname.

OLE_GetLastObjectError() Function

This function returns the error number of the last OLE error.

Syntax

```
errnum = OLE_GetLastObjectError();
```

Arguments

errnum

The number of the last OLE error.

OLE_GetLastObjectErrorMessage() Function

This function returns the error message of the last OLE error.

Syntax

```
errmsg = OLE_GetLastObjectErrorMessage();
```

Arguments

errmsg

The error message of the last OLE error.

OLE_ResetObjectError() Function

In a script, use the OLE_ResetObjectError() function to reset the last OLE error so that the last OLE error number is set to zero and last OLE error message is set to blank.

This can be used for identifying any errors in a batch of OLE functions.

Syntax

```
OLE_ResetObjectError()
```

OLE_ShowMessageOnObjectError() Function

By default, when an OLE error occurs, an error message dialog box is displayed.

In a script, you can specify whether or not to display the error message dialog box by using the function OLE_ShowMessageOnObjectError().

Syntax

```
OLE_ShowMessageOnObjectError(Boolean)
```

Arguments

Boolean

A value that determines if an OLE error message dialog box is displayed or not. A literal Boolean value, discrete tagname or Boolean expression with following meanings:

0 - no OLE error message dialog box is displayed when an OLE error occurs

1 - an OLE error message dialog box is displayed when an OLE error occurs

Example(s)

This script suppresses all OLE error message dialog boxes. When OLE errors occur, no error message dialog boxes are displayed.

```
OLE_ShowMessageOnObjectError(0);
```

OLE_IncrementOnObjectError() Function

In a script, you can use the OLE_IncrementOnObjectError() function to designate an integer tagname as counter for the number of OLE errors.

Syntax

```
OLE_IncrementOnObjectError(integertag)
```

Parameters

integertag

The tagname that acts as a counter.

Remarks

If OLE error message dialog boxes are displayed, the counter tagname is only incremented after the OLE error message dialog box is closed.

Example(s)

This script designates the integer tagname errorcount as error counter, hides the error message dialog boxes and attempts to create an OLE object based on an invalid OLE class name. This creates an error and the tagname value errorcount is incremented to 1.

```
errorcount = 0;
OLE_IncrementOnObjectError(errorcount);
OLE_ShowMessageOnObjectError(0);
OLE_CreateObject(%WS,"InVaLiD.cLaSs.nAmE");
```

Things you can do with OLE

You can use the following scripts to get an idea of the powerful functionality you can add to an application using OLE objects.

Produce random numbers

In a script, use the following commands to produce a random number between 0 and 255:

```
OLE_CreateObject(%SR,"System.Random");
randtag = (%SR.NextDouble)*255;
```

Create user interface dialog boxes

In a script, use the following commands to produce a user interface dialog box:

```
dim DlgBody as message;
dim DlgTitle as message;
dim Style as integer;
dim Result as integer;
DlgBody = "Do you want to open the valve ’MR-3-FF’?";
DlgTitle = "Confirm Opening Valve MR-3-FF";
Style = 48;
OLE_CreateObject(%WS,"Wscript.Shell");
result = %WS.Popup(DlgBody,1,DlgTitle,Style);
```

This example creates the following user interface dialog box.

The Style tagname determines which icon and which buttons appear on the dialog box. Use the following values:

Icon	Style	Value
(no icon)	no icon	0
	Error icon	16
	Question mark icon	32
	Warning icon	48
	Information icon	64

To use a particular button, add one of the following values to the Style value:

Value	Style
0	Only **OK** button
1	**OK** and **Cancel** buttons
2	**Abort**, **Retry** and **Ignore** buttons
3	**Yes**, **No** and **Cancel** buttons
4	**Yes** and **No** buttons
5	**Retry** and **Cancel** buttons
6	**Cancel**, **Try Again** and **Continue** buttons

The Result tagname contains the button number the user selected. This can be used for conditional branching in your InTouch script. Following result codes are possible:

Result Value	Meaning
1	**OK** button was pressed
2	**Cancel** button was pressed
3	**Abort** button was pressed
4	**Retry** button was pressed
5	**Ignore** button was pressed
6	**Yes** button was pressed
7	**No** button was pressed
10	**Try again** button was pressed
11	**Continue** button was pressed

Open the Windows Date and Time Properties panel

In a script, use the following commands to open the Windows Date/Time Properties panel:

```
OLE_CreateObject(%WP,"Shell.Application");
%WP.SetTime();
```

You can do similar tasks by calling different methods and passing them to the referenced OLE object:

This Method	Opens The Panel
TrayProperties()	**Tray** properties
FileRun()	**File Run** dialog box
FindFiles()	**Find Files** dialog box
FindComputer()	**Find Computer** dialog box
ShutdownWindows()	**Shutdown Windows** panel

Read and write to the registry

In a script, you can use OLE to read from and write to the Windows registry by:

- Creating an OLE object based on the Windows class Wscript.Shell.
- Using the RegRead() and RegWrite() methods of the OLE object.

For example, these commands read the installed version of the InTouch HMI directly from the registry key and store the value in the rkey message tagname:

```
OLE_CreateObject(%WS,"Wscript.Shell");
```

```
rkey = %WS.RegRead("HKLM\SOFTWARE\Wonderware\InTouch\Installation\Version");
```

These commands write the value 1 to the registry key that determines if file extensions are hidden for the currently logged on user:

```
OLE_CreateObject(%WS,"Wscript.Shell");
%WS.RegWrite("HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\
HideFileExt",1,"REG_DWORD");
```

Minimize windows

In a script, you can use the following commands to minimize all windows on your desktop:

```
OLE_CreateObject(%WA,"Shell.Application");
%WA.MinimizeAll();
```

You can do similar tasks by calling these methods:

This Method	Arranges Windows
TileHorizontally()	Tiles all Windows horizontally
TileVertically()	Tiles all Windows vertically
CascadeWindows()	Cascades all Windows
UndoMinimizeALL()	Restores all Windows

Script ActiveX controls

You can use ActiveX controls to read from and write to tagnames and I/O references. In a script, you can reference ActiveX controls.

You can also create scripts that execute when an event occurs for the ActiveX control. These scripts can be reused and imported into other applications.

Call ActiveX control methods

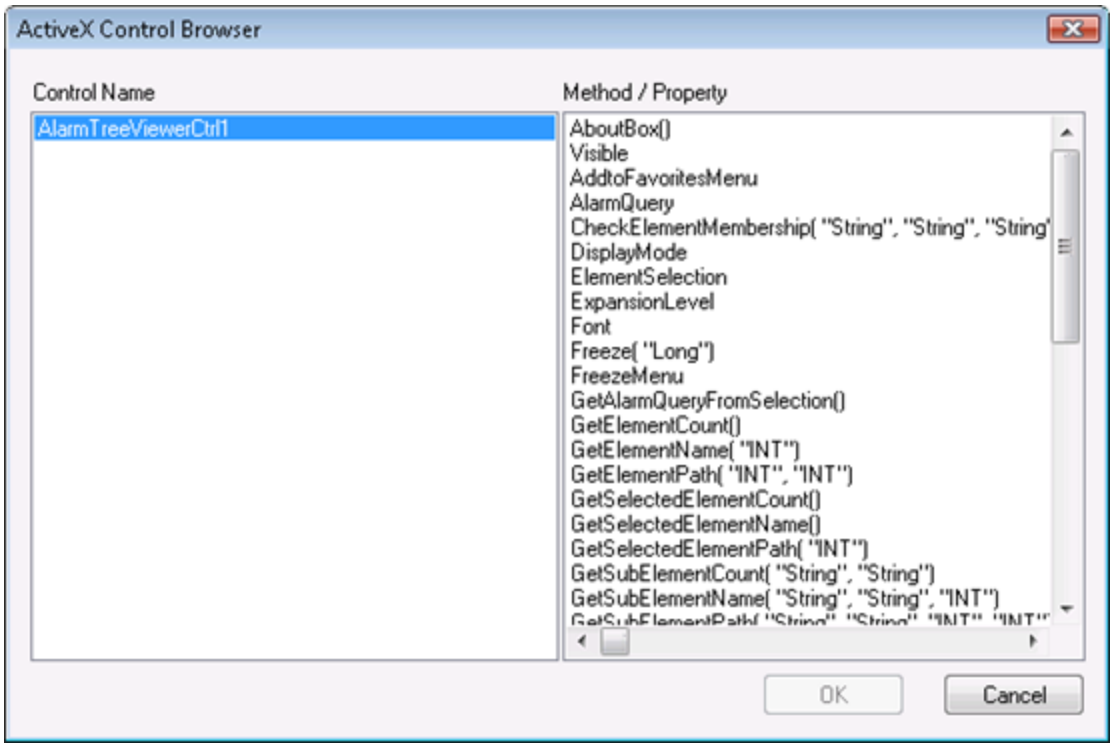
In a script, you can call methods of an ActiveX control to perform actions supported by the ActiveX control. ActiveX methods can be called from any type of InTouch QuickScript or ActiveX Event script.

Note: To call the ActiveX method when an ActiveX event occurs, there are some prerequisite things you need to do. See Configure ActiveX event scripts.

Call an ActiveX control method

1. In a script dialog box, on the **Insert** menu, select **ActiveX**.

   The **ActiveX Control Browser** dialog box appears.



2. Select the name of the ActiveX control from the left pane. The right pane contains the names of properties and methods that are supported by the ActiveX control.
3. Select the name of the method to use from the right pane and then select **OK**. The method name and default parameters are pasted into the script window at the cursor position.
4. Configure the method parameters inside the parentheses, to your specifications.

Access ActiveX control properties from the InTouch HMI

In a script, you can read from and write to ActiveX control properties to exchange data between the ActiveX control and the InTouch tagnames and display links.

Configure ActiveX control properties to read and write data

In a script, you can read data from and write data to an ActiveX control. You use the ActiveX control properties associated with specific ActiveX controls.

There are two ways of doing this:

- Use the ActiveX control property in an InTouch HMI QuickScript or ActiveX event script. The property value is read or written every time the script is executed.
- Link the ActiveX control property directly to an InTouch HMI tag or I/O reference. The property value is read or written at every update interval.

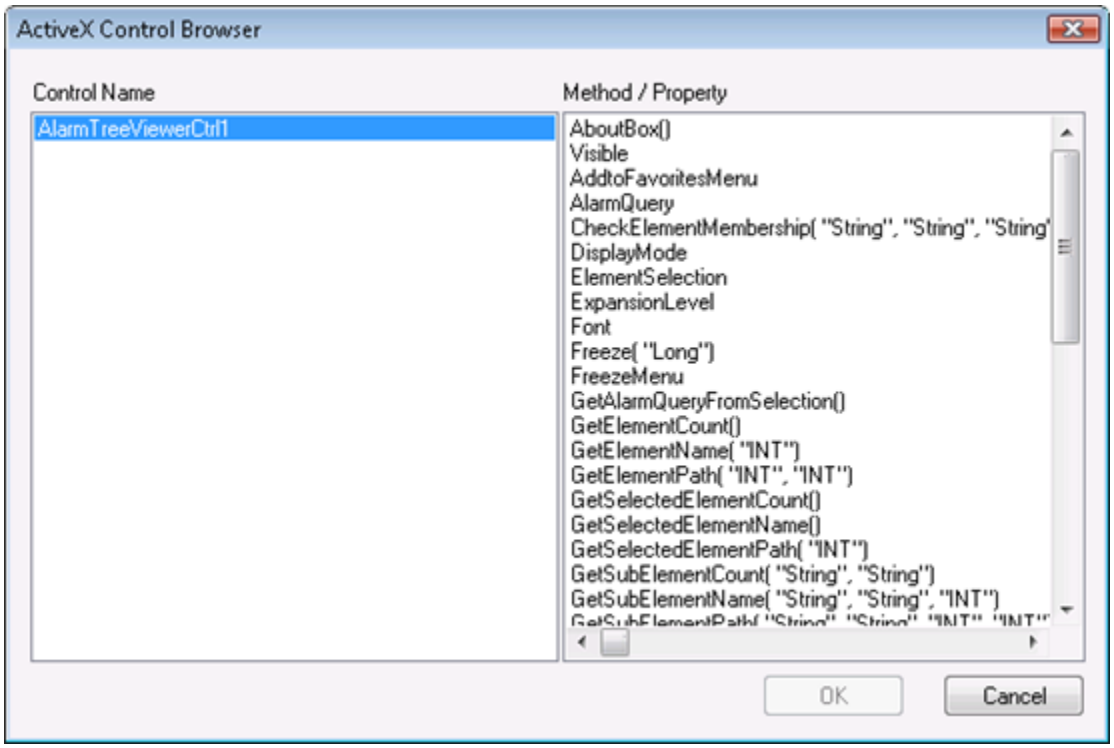
Configure scripts to read and write ActiveX control properties

In a script, you can configure ActiveX control properties to either write values to or read values from InTouch

HMI tagnames or other expressions.

Read data from or write data to an ActiveX control property

1. Open a script window, point to **Insert**, and select **ActiveX**. The **ActiveX Control Browser** dialog box appears.



2. Select the name of the ActiveX control from the left pane. The right pane contains the names of properties and methods of the selected ActiveX control.
3. Select the name of the property to use from the right pane. The property name is inserted into the script window at the cursor position.
4. Assign the property name to a tag or use according to your specifications.
5. Select **OK**.

Example(s)

The following script reads the ToPriority property of the ActiveX control instance AlarmViewerCtrl1 into the integer tagname topri.

```
topri = #AlarmViewerCtrl1.ToPriority;
```

The following script writes the value MS Comic to the Font property of the ActiveX control called AlarmViewerCtrl1. This example changes the display font of the AlarmViewer ActiveX control dynamically.

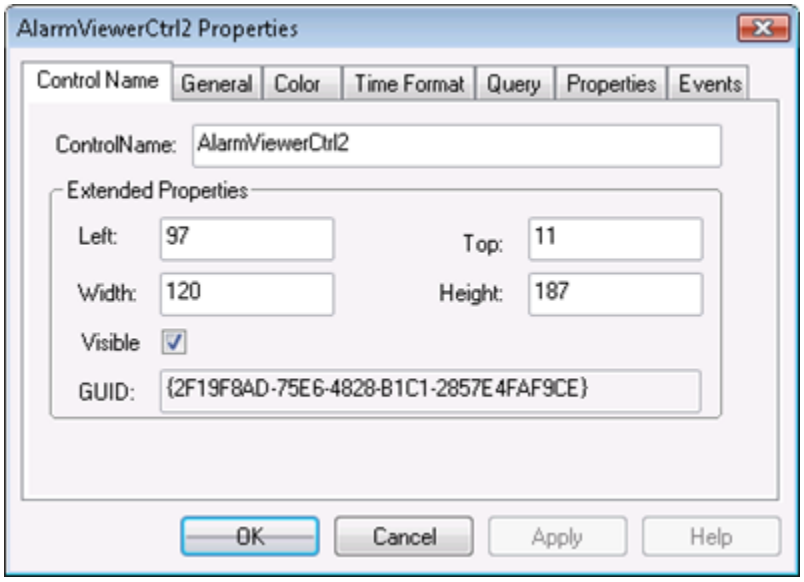
```
#AlarmViewerCtrl1.Font = "MS Comic";
```

Link ActiveX control properties to tag or I/O references

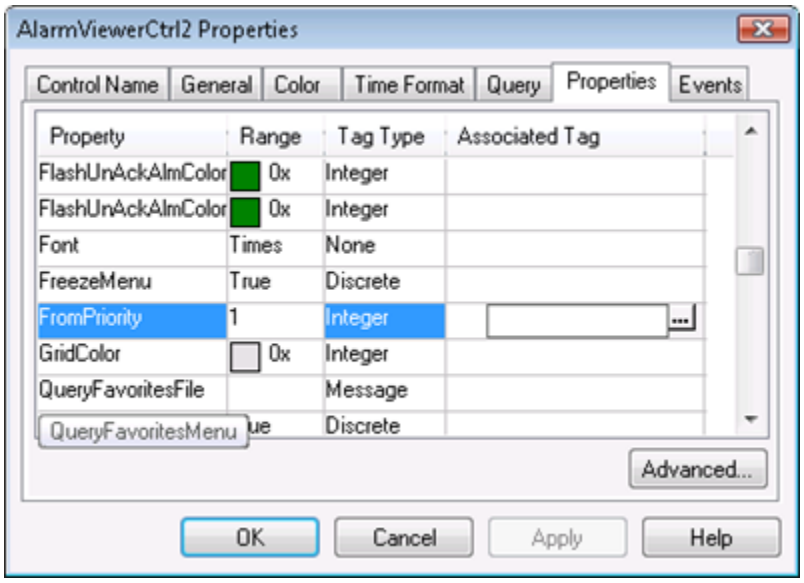
You can link ActiveX control properties to InTouch HMI tags or I/O references.

Link ActiveX control properties to tags or I/O references

1. Double-click the ActiveX control. The properties dialog box of the ActiveX control appears.



2. Select the **Properties** tab and scroll to the right.
3. Select the property in the list.



4. Assign a tag or I/O reference. Do either of the following:
   - Type the tag or I/O reference directly into the **Associated Tag** column.
   - Select the ellipsis button in the **Associated Tag** column between the square parenthesis. The **Select Tag** dialog box appears. Choose a tag and select **OK**.
5. Select **OK**.

Create and reuse ActiveX event scripts

An ActiveX control can support events, such as single-clicking on the control, that you can use to associate

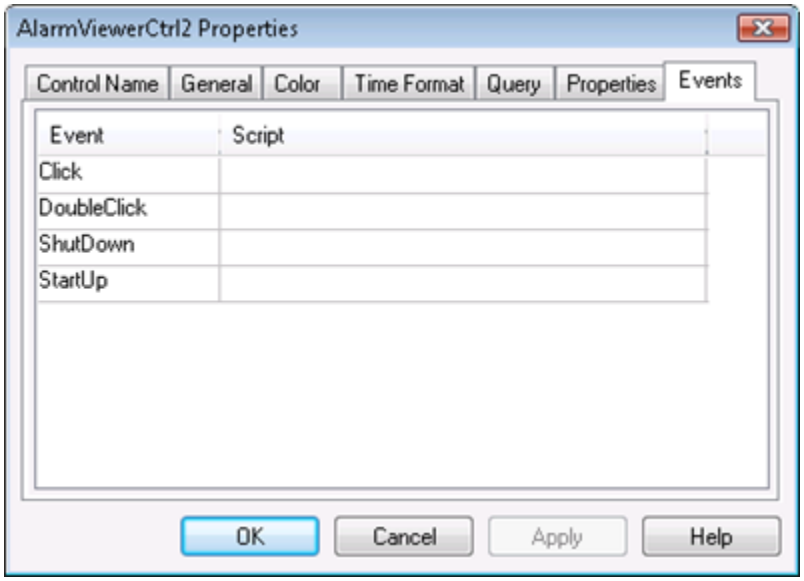
certain actions with. These actions are stored in ActiveX event scripts.

Create ActiveX event scripts

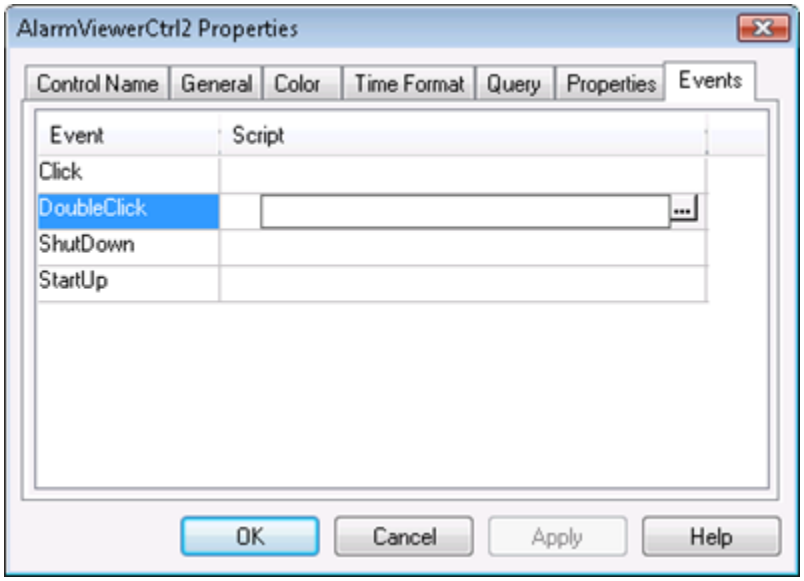
You can create or reuse an event script that is executed every time a specific ActiveX control event occurs, such as selecting an ActiveX control.

Create an ActiveX event script

1. Double-click the ActiveX control. The properties dialog box appears.
2. Select the **Events** tab.



3. Select the event to associate. Brackets and ellipses appear in the **Script** column.



4. In the **Script** column of the corresponding row, select the area between the brackets.
5. Enter a new name and select **OK**. When a message appears, select **OK**. The **ActiveX Event Scripts** dialog box appears.

6. Create the script according to your specifications.

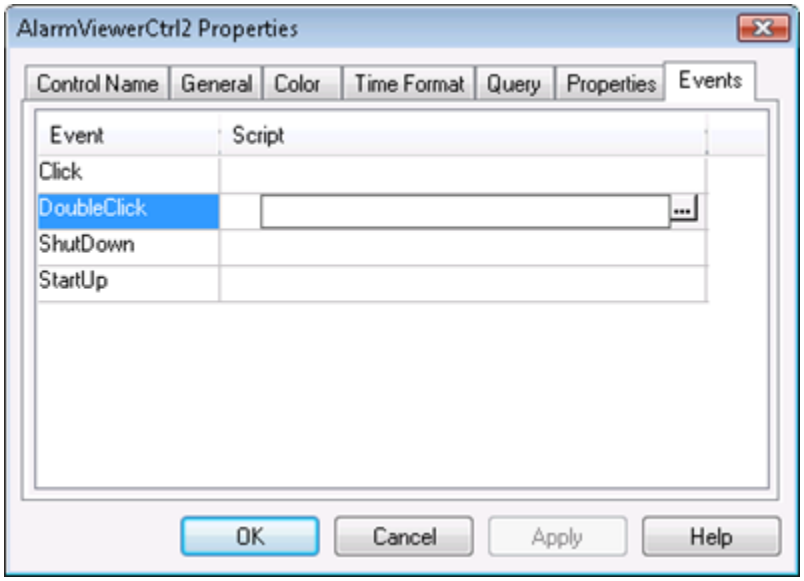
Reuse ActiveX event scripts

You can reuse ActiveX event scripts if they are created by the same ActiveX control parent and event.

For example, if you have multiple AlarmViewer ActiveX controls in an application, they can share event scripts for the DoubleClick event.

Reuse an ActiveX event script

1. Double-click the ActiveX control. The properties dialog box appears.
2. Select the **Events** tab.
3. Select the event to associate. Brackets and ellipses appear in the **Script** column.



4. In the **Script** column of the corresponding row, select the ellipsis button. The **Choose ActiveX Script** dialog box appears.
5. Select an ActiveX script and click **OK**.
6. Select **OK** again.

Create self-referencing ActiveX event scripts

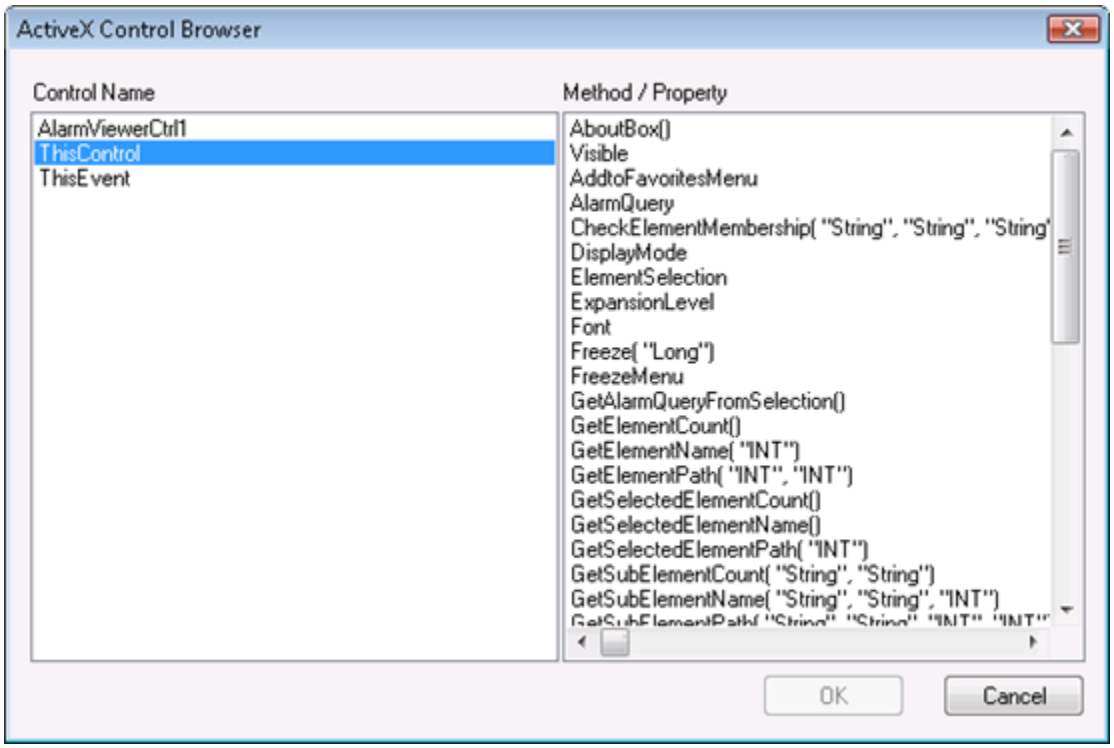
If you use ActiveX event scripts, you can configure them to reference themselves instead of an absolute ActiveX control name. This is useful when you create ActiveX event scripts that will be reused. ActiveX event scripts can either:

- Reference the specific ActiveX control that produced the event (ThisControl).
- Reference the specific event that called the script (ThisEvent).

Referencing the specific event enables the ActiveX control to pass other parameters to the ActiveX control script.

Create self-referencing ActiveX event scripts

1. Create an ActiveX event script for a specific ActiveX event. See Create ActiveX event scripts.
2. In the **ActiveX Event Script** dialog box, choose **Insert**, and then select **ActiveX**. The **ActiveX Control Browser** dialog box appears.



3. In the left pane, do one of the following:
   - Select **ThisControl** to see properties and methods that you can use in connection with this control (and any other control that you reuse this script in).
   - Select **ThisEvent** to see properties and methods of the ActiveX control that you can use in connection with the self-referencing event.
4. In the right pane, highlight one of the properties or methods and select **OK**. The selected property or method is pasted to the script window.
5. Configure the script.
6. Select **OK**.

For example, this statement writes the value of the ClicknRow event parameter to the ClickedRow tag:

```
ClickedRow = ThisEvent.ClicknRow;
```

Import ActiveX event scripts

You can import ActiveX event scripts from other InTouch HMI applications so as to reuse them in the application currently under development.

Import ActiveX event scripts from other applications

1. Open WindowMaker.

2. On the **File** menu, select **Import**, choose **Visualization**, and then select **Windows and Scripts**.

   The **Open folder** dialog box appears.
3. Browse to the InTouch HMI application that contains the ActiveX event scripts to import.
4. Select **OK**.
5. Select the **ActiveX Event Scripts** checkbox and select **Import**. All ActiveX event scripts are imported into the current InTouch HMI application.

ActiveX controls

ActiveX controls are stand-alone software components that bring additional functionality to InTouch applications. ActiveX controls have properties, methods, and events that can be modified while an application is running to change the behavior of the control.

You can use several types of ActiveX controls in your InTouch application.

- The InTouch HMI includes ActiveX controls for alarming.
- Other products, such as ActiveFactory, provide controls for manipulating and analyzing data.
- You can use third-party ActiveX controls.
- You can develop your own ActiveX controls in Visual Basic or C.

You can use any number of ActiveX controls in your InTouch application. You can:

- Select and paste an ActiveX control into any application window.
- Adjust the size of the control, if sizing is supported by the control.
- Duplicate, cut, copy, paste and delete ActiveX controls.
- Align ActiveX controls: left, right, top, bottom, and center point.
- Add ActiveX controls
- Combine ActiveX controls with other objects when creating a cell.
- Use the properties, methods, and events supported by the particular ActiveX control properties.

InTouch applications do not support the following types of ActiveX controls:

- Windowless controls
- Simple frame site or area box
- Containers
- Data controls
- Dispatch objects

InTouch applications only support these data types: discrete (Boolean), integer (32-bit numbers), real (floating point IEEE notation with 32 bits), and message (strings up to 131 characters). Unsupported data types include variant, pointers, arrays, structures, and parameterized properties.

ActiveX controls cannot overlap other InTouch objects, such as window controls or graphic objects. Too many ActiveX controls on one window can reduce system performance.

Use ActiveX controls

You select an ActiveX control and paste it into a window and then double-click it. Its configuration dialog box appears.

When you configure an ActiveX control, you can give it a unique control name. The control name can then be referenced in scripts.

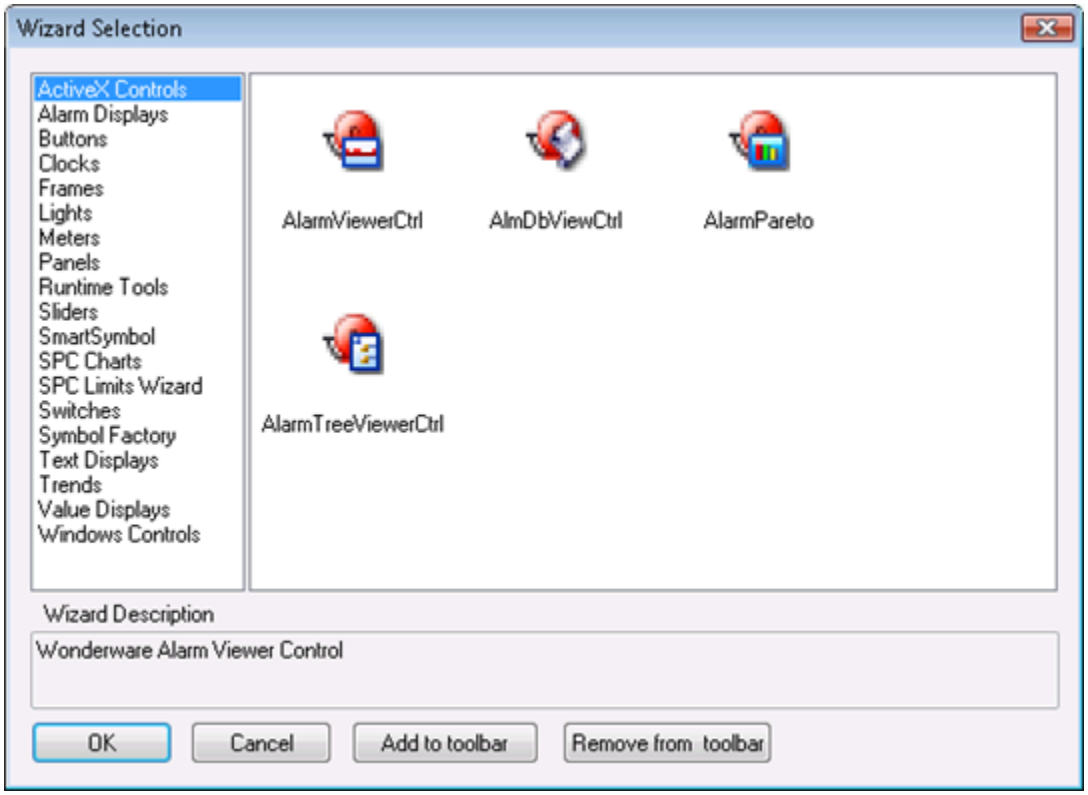
Use an ActiveX control in an InTouch application

1. Install the ActiveX control. See Install and remove ActiveX controls.
2. Select and paste the ActiveX control into an InTouch window.
3. Configure the ActiveX control properties and bind them to tags.
4. Associate ActiveX events to ActiveX Event scripts.
5. Call ActiveX methods and set ActiveX control properties in ActiveX Event scripts, or other InTouch QuickScripts.

Place an ActiveX control in a window

1. On the **Draw** menu, in the **Insert** group, select **Wizards**.

   The Wizard Selection dialog box appears.



2. In the list of wizards, select the **ActiveX Controls** category. All available ActiveX controls appear in the display area.
3. Double-click the ActiveX control to use. The dialog box closes and the cursor changes to the corner symbol.
4. Select the location to paste the ActiveX control.

Add ActiveX controls to the toolbar

1. On the **Draw** menu, in the **Insert** group, select **Wizards**.
   The Wizard Selection dialog box appears.

1. Select the ActiveX control to add.
2. Select **Add to toolbar**.

Remove ActiveX controls from the toolbar

1. On the **Draw** menu, in the **Insert** group, select **Wizards**.
   The Wizard Selection dialog box appears.
2. Select **Remove from toolbar**. The **Remove Wizard from Toolbar** dialog box appears.
3. Select the ActiveX control to remove.
4. Select **OK**.

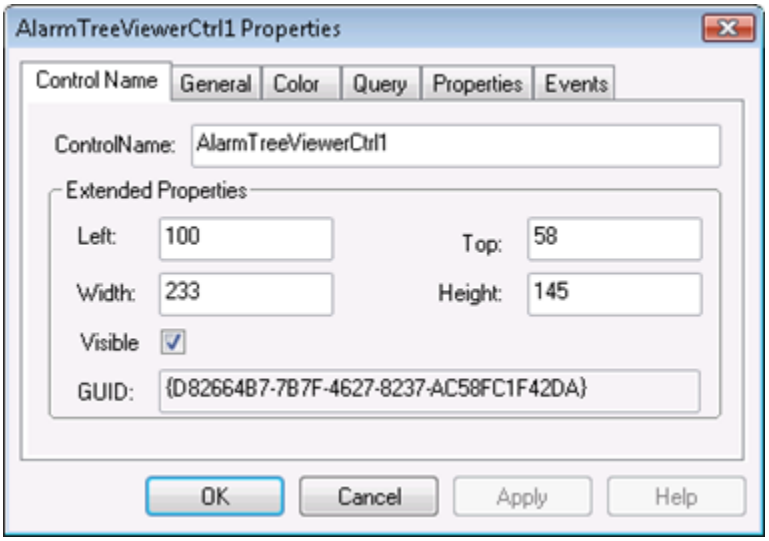
Configure ActiveX controls

The properties that you can configure for a particular ActiveX control are determined by the ActiveX control itself. All properties have a default value.

You must paste an ActiveX control into an InTouch window before you can configure its properties.

- A default control name, such as Calendar1, is generated when you paste the ActiveX control in a window. You reference the control name in scripts. An ActiveX control must be running in an open application window for any script to run against it.
- You can assign the ActiveX control properties to InTouch tags. You must assign each property type to a corresponding InTouch tag type.

ActiveX controls have three standard tabs: **Control Name**, **Properties**, and **Events**.



Use the **Events** tab to assign scripts to available control events, such as when the user double-clicks the mouse.

Any other tabs and their configurations are unique to the control and depend upon its properties. For example, some controls may require you to configure colors and fonts, while others may not have these properties.

Name ActiveX controls

A new instance of a control is created and given a unique name when you:

- Select **Duplicate**.
- Select **Cut** or **Copy** and then **Paste**.
- Select **Save Window As**.
- Select **Undo** and then **Redo.**
- Import a window that contains a control.

ActiveX controls must have unique names. If you try to use a name that already exists for a control, an error message appears.

Name an ActiveX control

1. Paste the ActiveX control into your development window.
2. Double-click the control. The control **Properties** dialog box appears.
3. Select the **Control Name** tab and then type a name for the ActiveX control in the **ControlName** box.

Standard operations on ActiveX controls

You can perform standard operations on ActiveX controls just like any other InTouch object. For more information, see Special manipulations for all objects.

Install and remove ActiveX controls

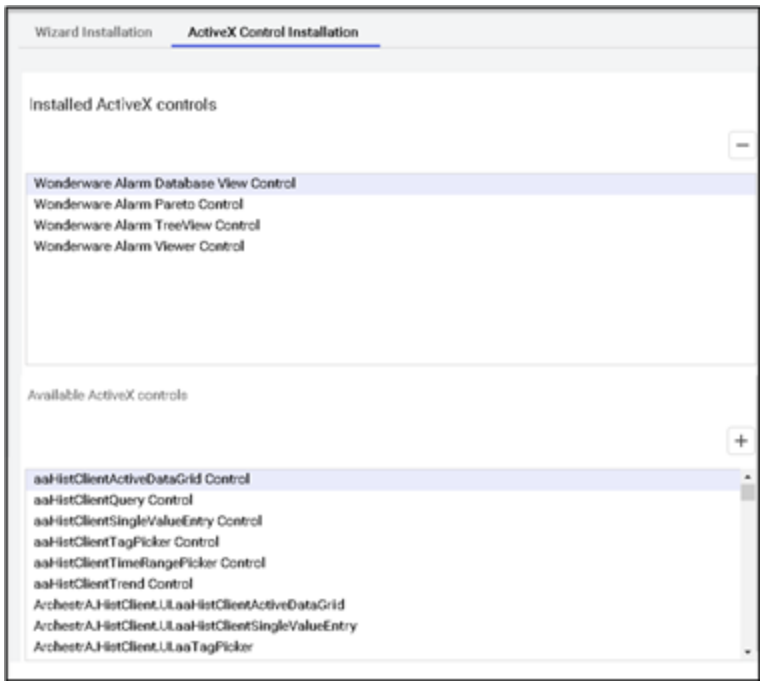
Even though you have already installed an ActiveX control on your computer, you must also make it known to the InTouch HMI by installing it to WindowMaker.

When you remove a control from WindowMaker, it is not deleted from your computer. It is simply removed from memory and does not function.

Install an ActiveX control

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Wizard/ActiveX**.

   The **Wizard/ActiveX** screen appears.
3. Select the **ActiveX Control Installation** tab. The **ActiveX Control Installation** property sheet appears.



4. In the **Installed ActiveX controls** list, select the control to install in the **Available ActiveX controls** list and then select the + icon..

   Tip: To select multiple controls, use the SHIFT key or CTRL key.

5. Select **Save**.

Remove an ActiveX control

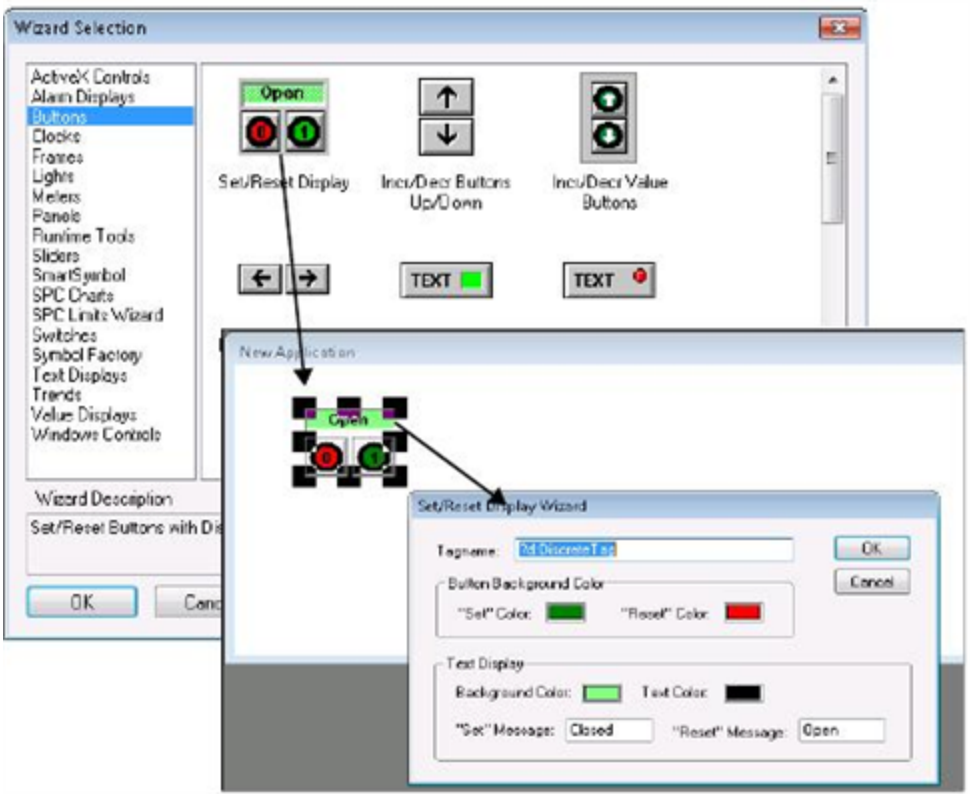
1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Wizard/ActiveX**. The **Wizard/ActiveX** screen appears.
3. Select the **ActiveX Control Installation** tab. The **ActiveX Control Installation** property sheet appears.
4. In the **Installed ActiveX controls** list, select the control to remove from your application and then select - icon.

   Tip: To select multiple controls, use the SHIFT key or CTRL key.

5. Select **Yes** to remove the control. The control is moved to the **Available ActiveX controls** list.
6. Select **Save**.

Wizards

Wizards are pre-designed, pre-built, and pre-programmed objects you only need to select, place and configure for your application.



Work with wizards

Using wizards, you do not spend time drawing the individual components for the object, or entering the value ranges for the object, or animating the object.

- You select wizards from the Wizards Selection dialog.
- You configure wizards by typing tags and QuickScripts in configuration dialog boxes.
- When you paste a selected wizard into a window and then double-click it, the **Configuration** dialog box appears.

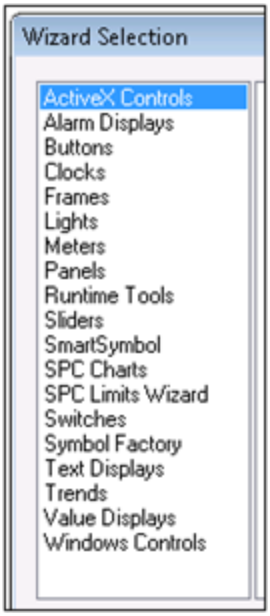
For example, in the case of a slider wizard, the configuration includes items such as the tagname to effect, the minimum and maximum range labels for the slider, the fill color, and so on. After the required configuration information is provided, the wizard is ready to use.

You can also develop your own complex wizards to provide behind the scenes types of operations. These operations can include creating complete display windows, creating or converting a database, importing an AutoCAD drawing, and configuring other applications.

Before creating your own wizards, you should investigate Industrial graphics, which offer wizard functionality, but do not require programming.

Types of wizards

Categories of wizards are shown in the **Wizard Selection** dialog box.



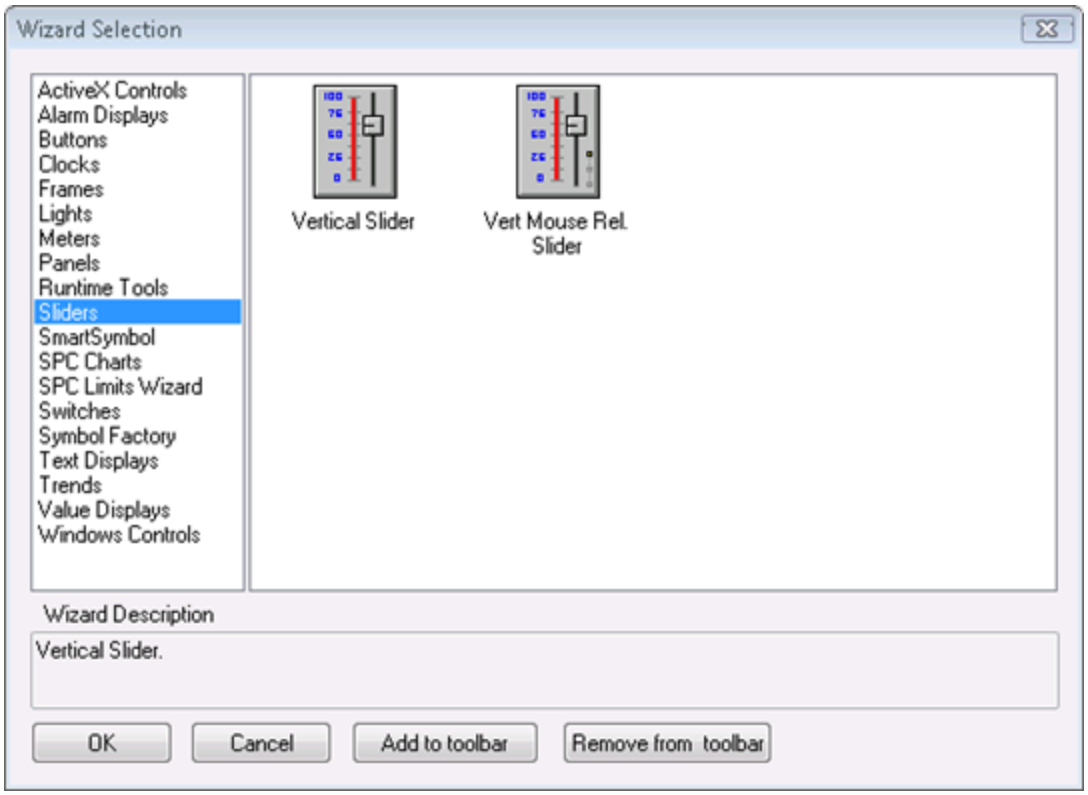
Trend Objects and Windows Controls Wizards are special wizards with unique parameters. For more information, see Trend objects and Windows Controls Wizards.

Add wizards to the toolbar

You can add frequently used wizards to the Wizard/ActiveX Toolbar for quick availability.

Add a wizard to the Wizards Toolbar

1. On the **Draw** menu, in the **Insert** group, select **Wizards**.
   The **Wizard Selection** dialog box appears.



2. In the left pane, select a wizard category, such as **Sliders**.
3. In the right pane, select a wizard, and then select **Add to toolbar**. The wizard button appears in the toolbar.

Remove wizards from the toolbar

1. On the **Draw** menu, in the **Insert** group, select **Wizards**.
   The **Wizard Selection** dialog box appears.
2. Select **Remove from toolbar**. The **Remove Wizard from Toolbar** dialog box appears.
3. Select the wizard to remove and select **OK**.

Paste wizard instances

You can place instances of a wizard in a window.

Place a wizard in a window

1. On the **Draw** menu, in the **Insert** group, select **Wizards**.
   The **Wizard Selection** dialog box appears.
2. In the left pane, select a wizard category.
3. In the right pane, select a wizard.
4. Select **OK**.
   The dialog box closes, and the cursor becomes a corner symbol.
5. Select the location to put the wizard.

Configure wizards

After you place a wizard in your application, double-click it to configure its properties. A properties dialog box appears that is custom to the selected wizard.

For more information on each particular type of wizard, see the wizard Help, if available.

Perform standard operations on wizards

You can cut, copy, paste, delete, and duplicate wizards, in the same way and with the same results as with other objects.

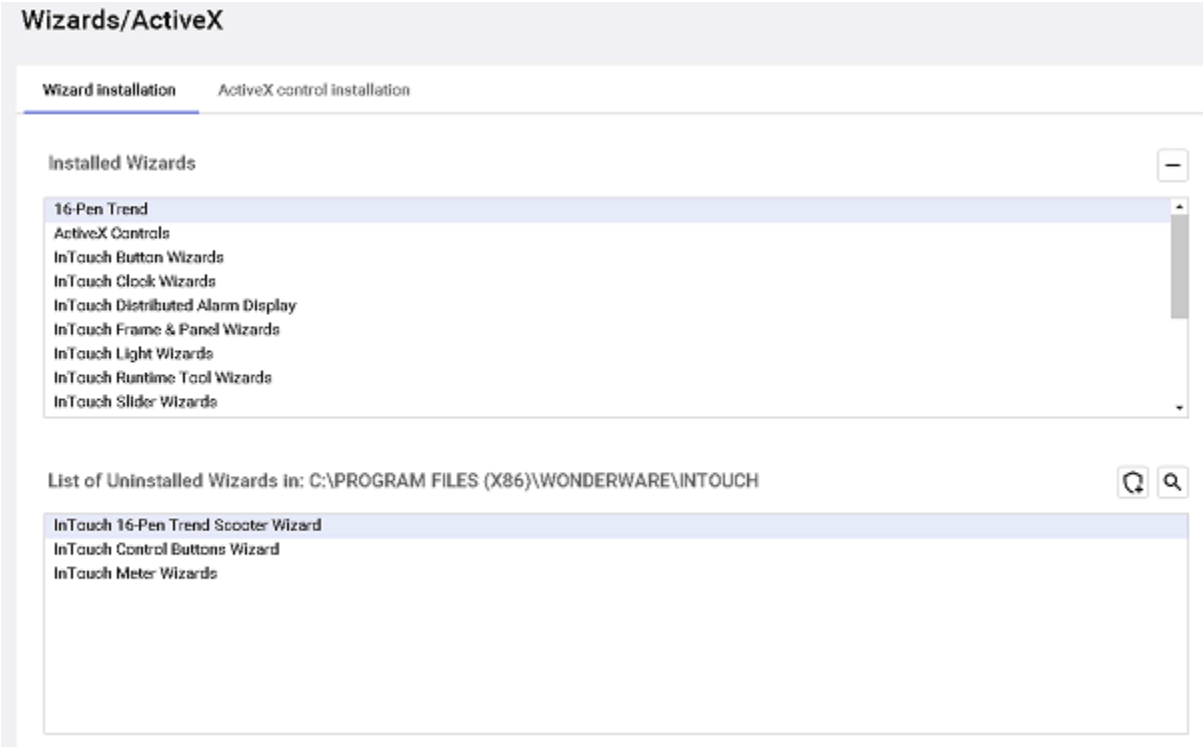
Install and remove wizards

You must install a wizard to WindowMaker so that you can use it in your application. When you remove a wizard from WindowMaker, it is not deleted from your computer.

Install a wizard

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Wizard/ActiveX**.

   The **Wizard/ActiveX configuration** screen appears.



3. Select from the **List of Uninstalled Wizards** and then select the **+** icon.
4. Select **Save**.

Remove a wizard

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Wizard/ActiveX**.

   The **Wizard/ActiveX** configuration screen appears.
3. In the **Installed Wizards** list, select the wizard to remove and then select the - icon.

   Tip: SHIFT+click or CTRL+click to make multiple selections.

   The wizard appears in the **List of Uninstalled Wizards** list.
4. Select **Save**.

Import wizards from another directory

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then click **Wizard/ActiveX**. The **Wizard/ActiveX** screen appears.
3. Click the **Search** icon. The **Browse for Folders** dialog box appears.
4. Browse to the directory containing the wizards to install and select **OK**. The **Wizard Installation** dialog box reappears with the imported wizards in the **List of Uninstalled Wizards**.

Trend objects

Trend objects are wizards that chart the values of tags over time.

There are three main types of trend objects:

- Real-time trends chart up to four tags in real time.
- Historical trends chart up to eight tags over a past time period.
- 16 Pen trends chart real-time and historical data for up to sixteen tags.

There is no limit to the number of trend objects, real-time or historical, you can place in a window.

Configure trend objects with the following items:

- Time span
- Value range
- Grid resolution
- Location of time and value stamps
- Pens and colors

Before you can use a trend wizard, you must enable logging for each tag to track, and also enable logging within the InTouch application.

Enable logging for tags

1. From the **Tagname Dictionary**, select a tag and then select **Log Data**.
2. If you have not done so previously, enable logging in the InTouch application.
   a. Open WindowMaker.

b. On the **File** menu, point to **Configure**, and then select **Historical Logging**.

   The **Historical Logging** screen appears.

c. Select the **Enable Historical Logging** checkbox or **Enable Storage to Historian** checkbox and select **OK**.

For more information on configuring and using trend objects, see Trending tag data.

Windows Controls Wizards

The Windows Controls Wizards are user interface objects such as drop-down lists, combo boxes, option selections or checkboxes.

Use Windows Control Wizards to present predefined sets of choices to your users. For example, you might create a drop down list of processes, recipes, or operator IDs. You can also enable and disable specified controls. You can even load and modify the contents of drop-down lists.

Run-time properties for Windows Controls Wizards are accessed through QuickScript functions rather than animation link expressions.

Windows Controls Wizards have properties like InTouch tag dotfields. They can be read-write or read-only. Some properties are accessible at development and some at run time. They are identified as ControlName.x, where x is the property.

For example, if the .Visible property of a windows control is equal to 0, the control is not visible in the window. As with InTouch tags, .Value is the default property for Windows Control Wizards.

Note: A more robust and flexible set of .NET based Windows controls is available if you use Industrial graphics.

Create and configure Windows Controls

When you create and configure Windows Control Wizards, keep in mind that:

- Windows Control Wizards cannot overlap one another.
- During run time, Windows Control Wizards will always be on top if they overlap with other objects in the window.
- Each Windows Control Wizard has a unique control name—which does not add to the tag count.
- The initial value of tags assigned to either a list box or combo box does not initialize the value of the list box or combo box. You must use the SetPropertyX QuickScript functions in a script to assign initial values that need to be different than the default values.

  Tip: Paste Windows Controls Wizards into your windows just like other wizards. To achieve the best readability, select a gray background for your windows controls. If your background color cannot be gray, place a gray Panel Wizard behind the Windows Control Wizards.

For each Windows Controls Wizard, you must specify an alphanumeric control name, where the first character is a letter. Underscores are allowed, but other special characters are not. For example, "Checkbox_1" is allowed, but "Checkbox#1" is not.

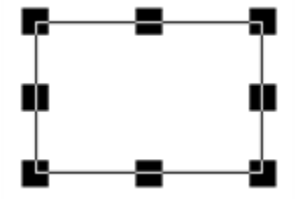
You can configure Windows Control Wizards using InTouch QuickScripts.

Create a text box

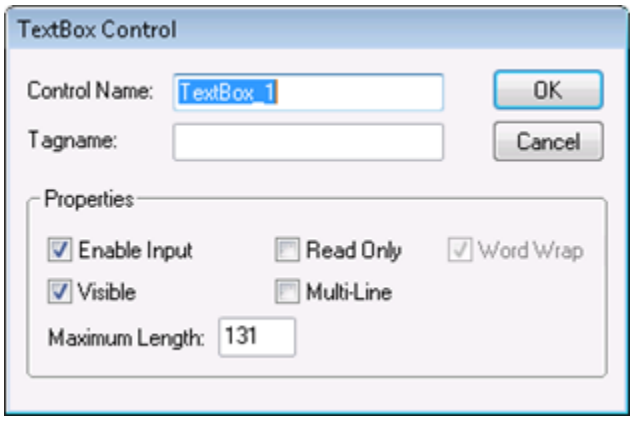
You use text boxes in your application to enable operators to enter text strings.

Create and configure a text box

1. Create and place a text box. Do the following:
   a. From the **Wizard Selection** dialog box, select **Windows Control Wizards**. The control wizard icons appear.
   b. Double-click the text box icon. Your application window reappears and the cursor changes to a left corner symbol.
   c. Select a location in the application window to place the wizard. The text box wizard appears with heavy black handles at the corners.



   a. Drag and resize the wizard to suit your application.
2. Double-click the wizard. The **Textbox Control** dialog box appears.



3. Configure the dialog box. Do the following:
   a. Type a control name, such as *TextBox_1*, in the **ControlName** box.
   b. Type a memory message tag name, such as *New_Value*, in the **Tagname** box.
   c. In the **Properties** area, select **Enable Input** and **Visible**.
4. Select **OK**. The **Textbox Control** dialog box closes.

Create a list box

You can create applications that enable your users to select items from a list of options.

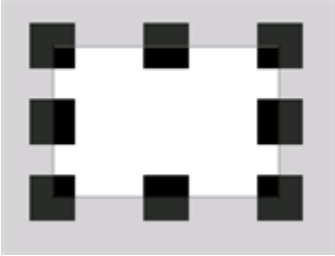
When adding a list box to your application, you place the control on the screen, set the properties to configure the list, and then write any scripts you may need.

List boxes can be loaded with items from a file or from keyboard input during run time. For more information,

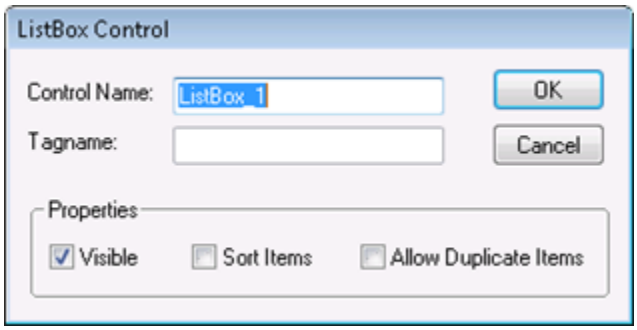
see Script windows controls.

Create a list box

1. Create and place the list box control. Do the following:
   a. In the **Wizard Selection** dialog box, select **Windows Controls**.
   b. Double-click the list box icon. Your application window reappears and the cursor changes to a left corner symbol.
   c. Select a location in the application window to place the control. The list box control appears.



2. Double-click the control. The **ListBox Control** dialog box appears.



3. Configure the control. Do the following:
   a. Type a control name, such as *ListBox_1,* in the **ControlName** box.
   b. Type a memory message tag name, such as *LB1_Value*, in the **Tagname** box.
   c. In the **Properties** area, configure how the control appears and functions.
4. Select **OK**.

Create a combo box

You can create applications that enable your users to select items from lists of options with a combo box. A combo box is a Windows control combining a text box and a list box.

When adding a combo box to your application, place the control on the screen, set the properties to configure the combo box, and then write any scripts you may need.

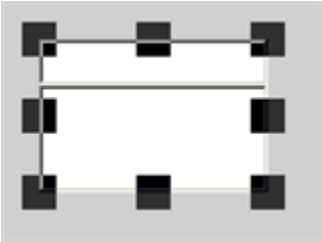
Combo boxes can be loaded with items from a file or from keyboard input during run time. For more information, see Script windows controls.

Create a combo box

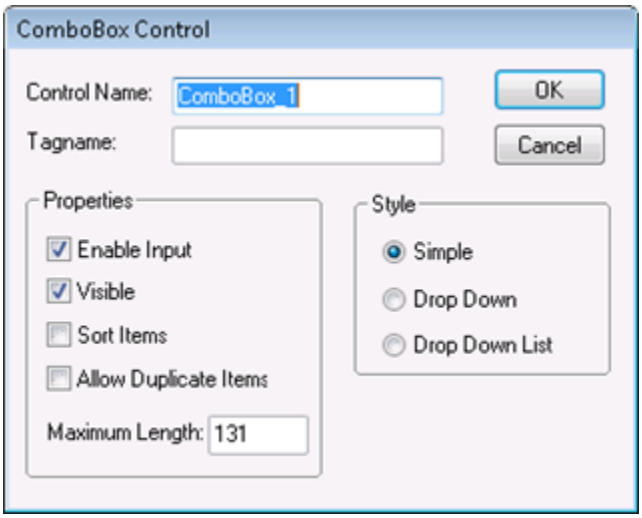
1. Create and place the combo box control. Do the following:
   a. In the **Wizard Selection** dialog box, select **Windows Controls**.

b. Double-click the combo box icon. Your application window reappears and the cursor changes to a left corner symbol.

c. Select a location in the application window to place the control. The combo box control appears.



2. Double-click the control. The **ComboBox Control** dialog box appears.



3. Configure the control. Do the following:

a. Type a control name, such as *ComboBox_1,* in the **ControlName** box.

b. Type a memory message tag name, such as *CB1_Value*, in the **Tagname** box.

c. In the **Properties** area, configure how the control appears and functions.

d. In the **Style** area, select the type of combo box.

4. Select **OK**.

Create a checkbox

A checkbox control enables an operator to select an option.

To configure a checkbox control

1. Create the checkbox. Do the following:

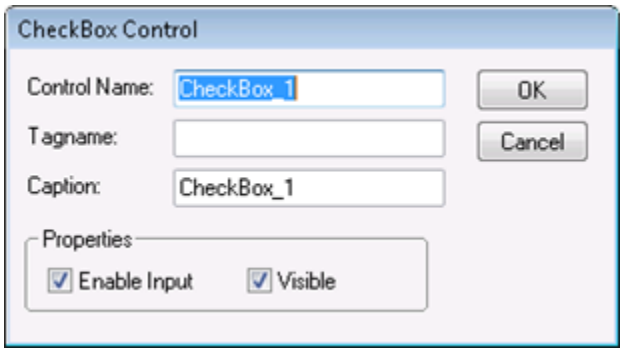
a. From the **Wizard Selection** dialog box, select **Windows Control Wizards**. The control wizard icons appear.

b. Double-click the checkbox icon. Your application window reappears and the cursor changes to a Left corner symbol.

c. Select a location in the application window to place the wizard. The checkbox wizard appears.

a. Drag and resize the wizard.

2. Double-click the wizard. The **Checkbox Control** dialog box appears.



3. Configure the wizard. Do the following:
   a. Type the control name.
   b. Type the discrete tag name, or double-click the empty tagname box to show the **Select Tagname** dialog, and select a tag.
   c. Type the caption to appear on the face of the button.
4. Select **OK**.

Create a radio button group

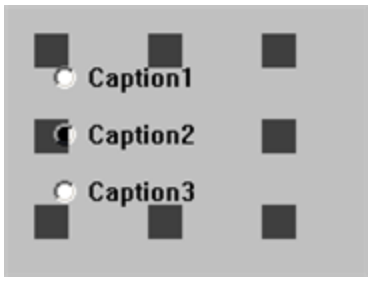
You use radio buttons when the user must select from one of several choices. When a user selects an option, the previously selected option is un-selected.

You create the options for a user as a group of radio buttons. Each radio button has a caption, and provides a unique value to your script.

You can only assign integer tags to radio button controls.

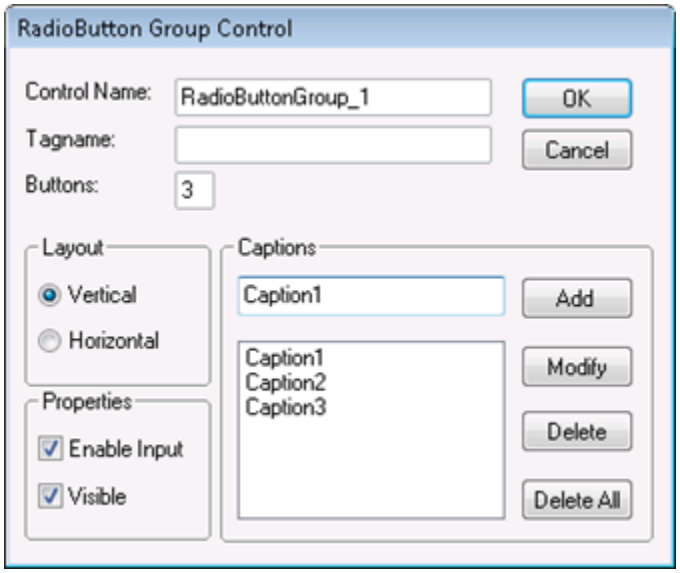
Create a Radio Button group

1. Create the Radio Button control wizard. Do the following:
   a. From the **Wizard Selection** dialog box, select **Windows Control Wizards**. The control wizard icons appear.
   b. Double-click the Radio button icon. Your application window reappears and the cursor changes to a left corner symbol.
   c. Select a location in the application window to place the wizard. The Radio Group control wizard appears with three radio buttons.



a. Drag and resize the wizard to suit your application.

2. Double-click the wizard. The **Radio Button Group Control** dialog box appears.



3. Configure the wizard. Do the following:
   a. Type the control name.
   b. Type an integer type tag name to link to this control.
   c. Type the number of buttons to show.
   d. Type the captions for each button.
   e. Set the layout and properties.
4. Select **OK**.

Script windows controls

You can use QuickScript functions in scripts to:

- Get or set the value of a control.
- Enable, disable, or hide a control.
- Work with items in combo boxes, list boxes, text boxes, and checkboxes.

The run-time properties can be either read-write or read-only, depending on the property. Use the GetPropertyX() and SetPropertyX() functions to control or retrieve these properties.

Get or set the value of a control

The .Value property is the default property for all InTouch Windows Controls Wizards.

Changes made to this property are synchronized in the InTouch tagname and the Windows Controls Wizards.

.Value Dotfield

Default property for all InTouch windows control wizards. Changes made to this property are synchronized in the InTouch tagname and the windows control wizards.

Category

windows control

Usage

The M, I and D are for the Memory, Integer and Discrete versions of the GetProperty and SetProperty functions.

```
[ErrorNumber=]GetPropertyM("ControlName[.Value]", Tagname);
[ErrorNumber=]SetPropertyM("ControlName[.Value]", Value);
[ErrorNumber=]GetPropertyI("ControlName[.Value]", Tagname);
[ErrorNumber=]SetPropertyI("ControlName[.Value]", Value);
[ErrorNumber=]GetPropertyD("ControlName[.Value]", Tagname);
[ErrorNumber=]SetPropertyD("ControlName[.Value]", Value);
```

Parameters

ControlName

Name of the windows control, e.g., ChkBox_4.

Tagname

The tagname to which the value of the property is written.

[.Value]

This property is optional. If not specified, the function always assumes the .Value property is being used.

Value

The actual value to be written or a valid InTouch tagname (of the same type as the property to be written to) that holds the property value to be written when the function is processed.

Remarks

The initial value of tagnames assigned to either a list box or combo box cannot be used to initialize the value of the list box or combo box.

This dotfield is read/write during development and run time. If the .Value dotfield is accessed by associating a tagname to either a list box or a combo box, it is read-only. If the .Value dotfield is assigned to a checkbox, radio button, or text box, it is read/write. The value you specify at development serves as the default for run time.

Data Type

Message (read/write) for text boxes, list boxes and combo boxes.

Integer (read/write) for radio buttons.

Discrete (read/write) for checkboxes.

Applies To

Text boxes, list boxes, combo boxes, checkboxes and radio buttons.

Example(s)

The following statement sets the .Value dotfield of the radio button object "RadioButton_1" to a value of 4:

```
SetPropertyI( "RadioButton_1.Value", 4 );
```

See Also

GetPropertyM(), SetPropertyM(), GetPropertyI(), SetPropertyI(), GetPropertyD(), SetPropertyD()

Enable or disable a control for user input

Use the .Enabled property to determine whether the control object can respond to operator-generated events.

.Enabled Dotfield

Determines whether the control object can respond to user-generated events.

Category

windows control

Usage

```
[ErrorNumber=] GetPropertyD("ControlName.Enabled",
Tagname);
[ErrorNumber=] SetPropertyD("ControlName.Enabled",
Discrete);
```

Parameters

ControlName

Name of the windows control. For example, ChkBox_4.

Tagname

A discrete tagname that holds the property requested.

Discrete

A discrete value or a discrete tagname that holds the value to be written when the function is processed. For a discrete value:

0 = Control is disabled.

1 = Control is enabled.

Remarks

This property is read/write during both development and run time.

Data Type

Discrete (read/write)

Applies To

Text boxes, list boxes, combo boxes, checkboxes and radio buttons.

Example(s)

The following statement disables the list box object named "ListBox_1".

```
SetPropertyD("ListBox_1.Enabled", 0);
```

See Also

GetPropertyD(), SetPropertyD()

Hide windows controls dynamically

Use the .Visible property to determine whether the windows control is visible in the window.

.Visible Dotfield

Determines whether the windows control is visible in the window.

Category

windows control

Usage

```
[ErrorNumber=]GetPropertyD("ControlName.Visible", Tagname);
[ErrorNumber=]SetPropertyD("ControlName.Visible", Number);
```

Parameters

ControlName

Name of the windows control. For example, ListBox_1.

Tagname

A tagname (of the same type to be returned) that holds the property value when the function is processed.

Number

A discrete value or a discrete tagname that holds the value to be written when the function is processed. For a discrete value:

0 = Control is invisible.

1 = Control is visible.

Remarks

This property is read/write in both development and run time.

Data Type

Discrete (read/write)

Valid Values

Applies To

Text boxes, list boxes, combo boxes, checkboxes and radio buttons.

Example(s)

The following statement hides the text box named "TextBox_1".

```
SetPropertyD("TextBox_1.Visible",0);
```

See Also

GetPropertyD(), SetPropertyD()

Add and delete items in combo boxes

Use the following script functions to add and delete items from combo boxes and lists

Script function	Effect
wcAddItem()	Adds an item to the end of the list of a list box or combo box. If sorting is enabled, the list is sorted after

Script function	Effect
	the item is added.
wcInsertItem()	Adds an item at a specified position in the list of a list box or combo box.
wcDeleteItem()	Deletes an item from a specified position in the list of a list box or combo box.
wcDeleteSelection()	Deletes the currently selected item from the list or combo box.
wcClear()	Removes all items from the list or combo box.

wcAddItem() Function

Adds an item to the end of the list of a list box or combo box. If sorting is enabled, the list is sorted after the item is added.

Category

windows control

Syntax

```
[ErrorNumber=]wcAddItem("ControlName", "MessageTag");
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

MessageTag

The message string to be shown. Actual string or message tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes and combo boxes.

Example(s)

The following statement adds the contents of the message string to the list box when the window (using On Show Window QuickScript) containing the list box is opened:

```
wcAddItem("ListBox_1", "Chocolate");
wcAddItem("ListBox_1", "Vanilla");
wcAddItem("ListBox_1", "Strawberry");
```

See Also

wcInsertItem()

wcInsertItem() Function

Inserts the specified string into the list of a list box or combo box at the specified position. Unlike the wcAddItem() function, the wcInsertItem() function does not sort a list, even if it is created as a sorted list box or combo box.

Category

windows control

Syntax

```
[ErrorNumber=]wcInsertItem("ControlName", ItemPosition, "Message");
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

ItemPosition

A number corresponding to the position of the item to be added. If this parameter is -1, the string is added to the end of the list. Any number or Integer tagname.

Message

Contains the string to insert at the position indicated by ItemPosition. Actual string or message tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes and combo boxes.

Example(s)

The following statement inserts a new item called "Blueberry" into a list box at fourth position from the top when an action script runs.

```
wcInsertItem("ListBox_1", 4, "Blueberry");
```

See Also

wcAddItem()

wcDeleteItem() Function

Deletes the item at a specified position from the list of either a list box or combo box.

Category

windows control

Syntax

```
[ErrorNumber=]wcDeleteItem("ControlName", ItemPosition);
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

ItemPosition

A number corresponding to the position of the item. Any number or Integer tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes and combo boxes.

Example(s)

The following statement deletes the third item in a list when an action script runs:

```
wcDeleteItem("ListBox_1", 3);
```

wcDeleteSelection() Function

Deletes the currently selected item from the list.

Category

windows control

Syntax

```
[ErrorNumber =]wcDeleteSelection("ControlName");
```

Parameter

ControlName
The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes and combo boxes.

Example(s)

The following statement deletes the currently selected item in a list box list when an action script runs:

```
wcDeleteSelection("ListBox_1");
```

wcClear() Function

Removes all items from the list box or combo box.

Category

windows control

Syntax

```
[ErrorNumber=]wcClear("ControlName");
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes and combo boxes.

Example(s)

The following statement clears all items in a list box when an action script runs:

```
wcClear("ListBox_1");
```

Load and save list items from or to a file

Use the following script functions to load and or save items in a combo box or list from or to a file.

Script function	Effect
wcLoadList()	Loads the contents of a list box or combo box with new items from a file.
wcSaveList()	Saves the contents of a list box or combo box to a file.

wcLoadList() Function

Loads a list box or combo box with new items from a file.

Category

windows control

Syntax

```
[ErrorNumber=]wcLoadList("ControlName", "Filename");
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

Filename

Contains the name of a file. If you do not supply a complete path name as part of the message parameter, the function checks the application directory for the message file. Actual string or message tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

If you use external files to fill the list and combo boxes, they must follow specific formatting and contain specific information. Format:

ControlType, ListCount

ListItem, ItemIndex

ListItem, ItemIndex

: :

: :

ListItem, ItemIndex

The ControlType is either COMBOBOX or LISTBOX.

For example, you want to load a list file to a combo box and it contains three items to select from and those items have no item data assigned. The format of the file appears as:

COMBOBOX, 3

Chocolate, 0

Vanilla, 0

Strawberry, 0

The COMBOBOX is the control type. The list count is 3 for Chocolate, Vanilla, and Strawberry. Chocolate is then listed as the first item or position 1. Vanilla as position 2, and Strawberry as position 3. Each of these items has a data value of 0.

For more information on item data, see wcSetItemData() Function.

Applies To

List boxes and combo boxes.

Example(s)

The following statement loads a properly formatted list (located in c:\wclist.txt.) into a combo box:

```
wcLoadList("Combobox_1", "c:\wclist.txt");
```

See Also

wcAddItem(), wcSaveList()

wcSaveList() Function

Saves the items of a list box or combo box to a file.

Category

windows control

Syntax

```
[ErrorNumber=]wcSaveList("ControlName","Filename");
```

Parameters

ControlName
The name of the windows control object. For example, ListBox_1. Actual string or message tagname.
Filename
Contains the name of a file. If the file does not exist, it is created. Actual string or message tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes and combo boxes.

Example(s)

The following statement saves the current items in a list box in a file (c:\newlist.txt) when an action script runs:

```
wcSaveList("ListBox_1", "c:\newlist.txt");
```

See Also

wcLoadList(), wcSetItemData()

Find items in a combo box or list

Use the wcFindItem() function to search for a specified item in a list box or combo box. If the item is found, this function returns the corresponding position to an integer tagname as the fourth parameter.

wcFindItem() Function

Determines the corresponding position of the first item in the list box or combo box that matches the supplied message string.

Category

windows control

Syntax

```
[ErrorNumber=]wcFindItem ("ControlName", "MessageTag", CaseSens, Tagname);
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

MessageTag

The message string to be compared. Actual string or message tagname.

CaseSens

Determines the type of the string comparison. It can either be a discrete value or tagname. The following values are valid:

0 = case-insensitive.

1 = case-sensitive.

Tagname

Integer tag into which the position of the matching item is returned. If no matching item is found, -1 is returned.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes, combo boxes

Example(s)

Assuming that ListBox_1 is a list box that contains "ItemA", "ItemB", and "ItemC", the function returns the following values into Result:

```
wcFindItem("ListBox_1", "ItemB", 0, Result);
```

returns 2

```
wcFindItem("ListBox_1", "Itemb", 1, Result);
```

returns -1

```
wcFindItem("ListBox_1", "itemc", 0, Result);
```

returns 3

```
wcFindItem("ListBox_1", "XYZ", 0, Result);
```

returns -1

Work with item indexes in a combo box or list

Use the following dotfields to work with the item index of a list box or combo box.

Dot Field	Effect
.TopIndex	The integer index of the topmost item in the list box.
.NewIndex	The integer index (tagname) of the last item added to the list box or combo box through the wcAddItem() or wcInsertItem() functions.
.ListIndex	The index (tagname or number) of the currently selected item in the list.

.TopIndex Dotfield

Sets or reads the integer index of the top-most item in a list box.

Category

windows control

Usage

```
[ErrorNumber=]GetPropertyI("ControlName.TopIndex", Tagname);
[ErrorNumber=]SetPropertyI("ControlName.TopIndex", Number);
```

Parameters

ControlName

Name of the windows control. For example, ListBox_1.

Tagname

An Integer tagname that holds the property value when the function is processed.

Number

The index number that defines the top-most item in the list box. Can be a literal integer value or a integer tagname or expression that provides an integer value.

Remarks

This property is available only in run time.

Data Type

Integer (read/write)

Applies To

List boxes.

Example(s)

The following statement sets the TopIndex of the list box object "ListBox_1" to a value of 14:

```
SetPropertyI("ListBox_1.TopIndex",14);
```

See Also

GetPropertyI(), SetPropertyI(), .ListIndex, .NewIndex

.NewIndex Dotfield

Returns the integer index (Tagname) of the last item added to the list box or combo box via the wcAddItem() or wcInsertItem().

Category

windows control

Usage

```
[ErrorNumber=]GetPropertyI("ControlName.NewIndex", Tagname);
```

Parameters

ControlName

Name of the windows control. For example, ListBox_4.

Tagname

A tagname containing the integer index of the last item added to the list or combo box. For empty lists, a value of -1 is returned.

Remarks

This property is only available in run time.

Data Type

Integer (read-only)

Applies To

List boxes and combo boxes.

Example

The following statement retrieves the index of the most recently added item in the list box named "ListBox_1" and writes that value to the memory integer tagname NewItemIndex.

```
GetPropertyI("ListBox_1.NewIndex", NewItemIndex);
```

See Also

GetPropertyI(), wcAddItem(), wcInsertItem(), .ListIndex, .TopIndex

.ListIndex Dotfield

Sets or reads the index (Tagname or Number) of the currently selected item in the list.

When using a list box, an index of -1 indicates that no item is currently selected.

When using a combo box, an index of -1 indicates that the user has typed new text into the text entry field of the control.

Syntax

```
[ErrorNumber=]GetPropertyI("ControlName.ListIndex", Tagname);
[ErrorNumber=]SetPropertyI("ControlName.ListIndex", Number);
```

Parameter

ControlName

Name of the windows control. For example, ListBox_4.

Tagname

The tagname to which the index of the currently selected item is written.

Number

The index number that defines a specific item in the list.

Remarks

The index number defines a specific item in a list. Use the .ListIndex dotfield to set or determine the index of the currently selected item in a list or combo box.

This property is available only in run time.

Data type

Integer (read or write)

Applies to

List boxes and combo boxes.

Example

This statement retrieves the index of the currently selected item in the list box named "ListBox_1" and writes that value to the memory integer tagname MyListBoxIndex.

```
GetPropertyI( "ListBox_1.ListIndex", MyListBoxIndex );
```

See Also

GetPropertyI(), SetPropertyI(), .NewIndex, .TopIndex

Count list box or combo box items

The .ListCount dotfield contains the number of items in a list box or combo box.

.ListCount Dotfield

Reads the number of items in the list box or combo box.

Category

Windows control

Syntax

```
[ErrorNumber=]GetPropertyI("ControlName.ListCount", Tagname);
```

Parameter

ControlName

Name of the windows control.

Tagname

A valid tagname that contains the integer count of the items in the list.

Remarks

This property is available only in run time.

Data Type

Integer (read-only)

Applies To

List boxes and combo boxes.

Example(s)

The following statement retrieves the number of items in the list box named *ListBox_1* and writes that value to the memory integer tag *MyListBoxCount*.

```
GetPropertyI("ListBox_1.ListCount", MyListBoxCount);
```

See Also

GetPropertyI(), .ListIndex

Get or set the value of a list item

Use the wcGetItemData() function to find the integer value associated with the list item identified by the item index.

Use the wcSetItemData() function to assign an integer value to the item in the list specified by item index. This assigns a number to a string.

wcGetItemData() Function

Reads the integer value associated with the list item identified by the ItemIndex parameter.

Category

windows control

Syntax

```
[ErrorNumber=]wcGetItemData("ControlName", ItemIndex, Tagname);
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

ItemIndex

A number corresponding to the position of the item. Any number or integer tagname.

Tagname

Actual name of a real or integer tagname. The wcGetItemData() function places the numeric value corresponding to the item into this tagname upon return from the function.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes and combo boxes.

Example(s)

The following statement retrieves the numeric value associated with the fifth item in a list box and returns it to the ItemValue Integer tagname when an action script runs:

```
wcGetItemData("ListBox_1", 5, ItemValue);
```

If the fifth item in the list is assigned the integer value 4500, the ItemValue tagname contains 4500.

See Also

wcSetItemData()

wcSetItemData() Function

Assigns an integer value of the item (the Number parameter) to the item in the list specified by the ItemIndex parameter. This function enables the assignment of a number to a string.

Category

windows control

Syntax

```
[ErrorNumber=]wcSetItemData("ControlName", ItemIndex, Number);
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

ItemIndex

An integer value specifying the list item you want to edit. Any number or Integer tagname.

Number

An integer value representing the item data. Any number or Integer tagname.

Remarks

You can create complete lists containing the items using a program like Notepad and then load them using one function call. Format the list as required by the wcSaveList() function.

For a list of returned error numbers, see Understand windows controls error messages.

Use the wcGetItemData() function to return the value (item data) associated with the list item. The tagname parameter contains the returned numeric value. This parameter could be an I/O Integer tagname that writes directly to the real world device.

Example(s)

A recipe has three ingredients; flour, sugar and salt. The quantity of flour is 4500 grams, sugar is 1500 and salt is 325 grams. The values are assigned to each of the list box items by using a data change script triggered by what recipe (tagname, RecipeName) is selected:

```
wcSetItemData("ListBox_1", 1, 4500); {set 1st item in the list (flour)=4500}
wcSetItemData("ListBox_1", 2, 1500); {set 2nd item in the list (sugar)=1500}
wcSetItemData("ListBox_1", 3, 325); {set 3rd item in the list (salt)=325}
```

See Also

wcLoadList(), wcSaveList(), wcGetItemData()

Get the name of a list item

Use the wcGetItem() function to return the item string associated with the corresponding item index in the list box or combo box.

wcGetItem() Function

Returns a string containing the contents of the item corresponding to the ItemIndex in the list box or combo box.

Category

windows control

Syntax

```
[ErrorNumber=]wcGetItem("ControlName", ItemIndex, Tagname);
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

ItemIndex

A number corresponding to the position of the item. Any number or Integer tagname.

Tagname

Message tagname. The wcGetItem function will place the data corresponding to the item index into this tagname upon return from the function.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes and combo boxes.

Example(s)

The following statement returns the string value of the tenth item in a combo box to the ListSelection message tag when an action script runs:

```
wcGetItem("Combobox_1", 10, ListSelection);
```

If item ten in the list is "Vanilla," then the ListSelection tag contains the string "Vanilla".

Load the contents of a text box

Use the wcLoadText() function to load the contents of a text box from a file. Use the wcSaveText() function to save the contents of a text box to a text file.

Note: If the tag name has been defined with a maximum length, only that number of characters can be assigned from the text box contents to the tag. If no tag is assigned to the text box, its contents can be up to 65,535 characters.

wcLoadText() Function

Replaces the contents of the text box with the contents of the file.

Category

windows control

Syntax

```
[ErrorNumber=]wcLoadText("ControlName", "Filename");
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

Filename

Contains the name of a file. If a complete path name is not supplied as part of the message parameter, the function will check the application directory for the file. Actual string or message tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

Text boxes.

Example(s)

The following statement loads a text file (c:\InTouch.32\readme.txt.) into a text box when the window (On Show Window script) containing the text box opens:

```
wcLoadText("Textbox_1", "c:\InTouch.32\readme.txt");
```

wcSaveText() Function

Saves the text contained in the text box to the specified file. If the file doesn't exist, it is created. If it does exist, it must be read/write.

Category

windows control

Syntax

```
[ErrorNumber=]wcSaveText("ControlName", "Filename");
```

Parameters

ControlName

The name of the windows control object. For example, ListBox_1. Actual string or message tagname.

Filename

Contains the name of the destination file. If you do not supply a complete path name, the file is saved in the application directory. If the file exists, it is overwritten. If the file does not exist, it is created. The resulting file can subsequently be loaded into a text box object using the wcLoadText() function. Actual string or message tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

Text boxes.

Example(s)

The following statement saves the current information entered in a text box to a file in c:\InTouch.32\newtext.txt when an action script runs:

```
wcSaveText("Textbox_1", "c:\InTouch.32\newtext.txt");
```

See Also

wcLoadText()

Check if a text box is read-only

Use the .ReadOnly dotfield to determine whether the content of the text box is read-only or read/write.

.ReadOnly Dotfield

Determines whether the content of the text box is read-only or read/write.

Category

windows control

Usage

```
[ErrorNumber=]GetPropertyD("Control<i>Name</i>.<i>ReadOnly</i>", Tagname);
```

Parameters

ControlName

Name of the windows control. For example, Textbox_1.

Tagname

A discrete tagname that holds the property value when the function is processed.

0 = Contents of the text box is read/write

1 = Contents of the text box is read-only

Remarks

This property is available in both development and run time.

Data Type

Discrete (read-only)

Applies To

Text boxes.

Example(s)

The following statement retrieves the read-only status of the Text box named "TextBox_1":

```
GetPropertyD("TextBox_1.ReadOnly",A_Tagname);
```

See Also

GetPropertyD(), SetPropertyD()

Get or set the label of a checkbox

The .Caption dotfield defines the message text of a checkbox.

.Caption Dotfield

Determines the message to be displayed with the checkbox.

Category

windows control

Usage

```
[ErrorNumber=]GetPropertyM ("ControlName.Caption", Tagname);
[ErrorNumber=]SetPropertyM ("ControlName.Caption", "Message");
```

Parameters

ControlName

Name of the windows control. For example, ChkBox_4.

Tagname

A message tagname that holds the property requested.

Message

A message string surrounded in quotes.

Remarks

This property is read/write during both development and run time.

Data Type

Message (read/write)

Applies To

Checkboxes.

Example

This statement sets the caption of the checkbox object "CheckBox_1" to "Blue Paint Option."

```
SetPropertyM("CheckBox_1.Caption","Blue Paint Option");
```

See Also

GetPropertyM(), SetPropertyM()

Understand windows controls error messages

Given an error number, wcErrorMessage(), returns a string message describing the error. It applies to list boxes, text boxes, combo boxes, radio buttons and checkboxes.

The Window Controls functions return values based on the result of processing QuickScript functions. The return value is used for error diagnostics. You can assign these values to integer tag names. For example:

```
ErrorNumber = wcGetItem("ControlName", Number, Tagname);
```

In this script, `ErrorNumber` is an integer tag that contains the returned error value. The returned value of the

function can be passed to the wcErrorMessage(). The wcErrorMessage() will return a string description of the error. For example:

```
ErrorMsg = wcErrorMessage(ErrorNumber);
```

In this script, `ErrorMsg` is a message type tag that contains the text of the returned error. The following table identifies numeric error values and their definitions.

Error Message	Definition
0	Success
-1	General failure
-2	Insufficient memory available
-3	Property is read-only
-4	Specified item already present
-5	Object name unknown
-6	Property name unknown
-x	Unknown error.

wcErrorMessage() Function

Returns a message string describing the error.

Category

windows control

Syntax

```
ErrorMessage=wcErrorMessage(ErrorNumber);
```

Parameters

ErrorMessage

Message tagname.

ErrorNumber

Number returned by all windows control functions. Any number or Integer tagname.

Remarks

For a list of returned error numbers, see Understand windows controls error messages.

Applies To

List boxes, text boxes, combo boxes, checkboxes and radio buttons.

Example(s)

If an error occurs while a list is being loaded, show the text description of the error into the ErrorDescription message tagname. In this example, a Value Display animation link is assigned to the ErrorDescription tagname to show the error message.

In the "On Show" window QuickScript:

```
ErrorNumber=wcLoadList("ListBox_1","c:\recipe.txt");
ErrorDescription=wcErrorMessage(errornumber);
```

You can use this function with all windows control functions to show error messages:

```
ErrorNumber=wcAddItem("ListBox_1","AM_4A4356");
ErrorMsg=wcErrorMessage(ErrorNumber);
```

Use an XML File to import a window

You create a command file, and then run WindowMaker with the command file to import a window from an XML file. In addition to importing a window, the command file can be used to:

- Create a new application.
- Delete a window.
- Rename a window.
- Send print information to a file or printer.
- Send a cross-reference to a file.

When the XML import is in progress, the following operations are not supported on a referenced symbol:

- Deleting a symbol
- Renaming a symbol
- Swapping symbol names

However, you can perform these operations after the XML import is complete.

The following system level operations are not supported:

- Launching console sessions
- Switching focus to other applications

You can run the InTouch DBDump and DBLoad utilities from the command line. However, for managed InTouch applications, run the DBDump and DBLoad utilities from the System Platform IDE extension. DBDump creates an export file containing tag information from an InTouch application. DBLoad imports tag information into an InTouch application. For more information about the DBDump and DBLoad utilities, see AVEVA™ InTouch HMI Application Maintenance.

Prepare an XML file

The following procedure describes how to create an XML file and import a window definition into an InTouch application.

1. Create an XML file. This does not define the tools to use, or basic XML syntax.
2. Check the XML file for adherence to XML formatting standards. You can verify the syntax of your XML file with the help of XML validation tools. To view your XML file, open it in Internet Explorer.
3. Prepare the tags in the application. You can use the InTouch DBDump and DBLoad utilities to extract, modify, and reload tags. For more information on DBDump, DBLoad, and the CSV file, see *AVEVA™ InTouch HMI Application Maintenance*.
4. Prepare a WindowMaker command file to perform the functions you need to accomplish.
5. Exit from your InTouch application, WindowMaker, and WindowViewer.
6. Run the WindowMaker command file from a command line prompt, or run the command file from a program or from an IDE extension.
7. Check for errors. If you defined an error log file, check it and check the Logger. If you ran the command file from a program, check the return code.

The following sample files help you create and import window definitions using the XML import functionality:

- Sample XML file that loads a window definition into an InTouch application.
- Sample command file.
- Two schema files. The import parser does not use these schema files. The schema files are more restrictive than the XML import parser.

  Note: Schema files are available in the InTouch installation folder after the installation is complete.

Prepare a command file

The WindowMaker command file is a text file. Commands are read from the file and then actions are taken against the InTouch application data set.

The command file uses the single-byte ANSI character set. You cannot use MultiByte Character Set (MBCS) or Unicode characters.

Specifics of the text file syntax are:

- The 8-bit ANSI character set is used, but only characters under 127 are allowed. Most control characters are stripped out.
- Lines that begin with the # character are treated as comments.
- Blank lines are ignored.
- Lines are terminated with a carriage return line feed (CRLF) sequence.
- Spaces can occur at the beginning of a line, at the end of a line, between the commands, the equal character, and its argument.
- For any arguments requiring quotation marks, spaces are not allowed between the quotation marks and other text.

- All leading and trailing white space is removed before each line is processed.
- Each command starts with a period.
- Each command file must start with the WindowMaker command-file command (.WINDOMAKERCOMMANDFILE).

A command file can contain multiple commands. If a command fails, the command processor continues with the next command in the file.

Create a minimum command file

An example of the minimum command file is:

```
.WINDOWMAKERCOMMANDFILE
.VERSION=1
```

Send print information to a file

You can send a listing of the entire InTouch application to a printer or to a text file. If the InTouch window contains Industrial Graphics then the output will be sent to an HTML file.

The following example command file prints application information to the printer named Printer01:

```
.WINDOWMAKERCOMMANDFILE
.VERSION=1
.PRINTAPPLICATIONINFORMATION
.OUTPUTTARGET=Printer
.OUTPUTTARGETNAME=Printer01
.GO
```

The following is an example command file that prints to a file. If the file exists, it is overwritten.

```
.WINDOWMAKERCOMMANDFILE
.VERSION=1
.PRINTAPPLICATIONINFORMATION
.OUTPUTTARGET=TextFile
.OUTPUTTARGETNAME=C:\MyApps\AppInfo.txt
.GO
```

Send a cross-reference to a file

You can create a cross-reference and send it to a file. If the file exists, it is overwritten. No user interaction is required.

An example command file:

```
.WINDOWMAKERCOMMANDFILE
.VERSION=1
.CROSSREFERENCE
.SEARCHFOR=TagName
.REFERENCETYPE=ByWindow
.OUTPUTFILE=C:\MyApps\AppCrossRef.Csv
.GO
```

Create a log file

You can log informational messages, warnings, and errors to a text file. If you do not specify this command, the processing status is sent to the Logger. If the file already exists, it is overwritten.

An example command file that logs errors to a file.

```
.WINDOWMAKERCOMMANDFILE
.VERSION=1
.COMMANDLOGFILE=C:\MyApps\LogFile.Txt
```

Ensure that the folder MyApps exists. If the folder MyApps does not exist, the log file is not created.

Command syntax

The following table lists the commands you can use in a WindowMaker command file.

Command	Description
.WINDOWMAKERCOMMANDFILE	Identifies the file as being a WindowMaker command file. If this command is not found, the file is not processed and WindowMaker exits with an error code.
.VERSION=1	This command indicates to WindowMaker that the file is not too new to be read.
.COMMANDLOGFILE=<Full File Path>	Information, warnings, and errors are logged to this file. If the file already exists, it is overwritten. If this command is omitted, check the Logger for information about the processing status.
.GO	Runs the command. Required after a command with one or more parameters.
.WINDOWCREATE	Creates a window from an XML file.
.XMLFILEPATH=<Full File Path>	The full file path to an XML file containing a window specification. Required when the WINDOWCREATE command is used.
.WINDOWDELETE	Deletes the specified InTouch application window by name. An error is not generated if the window name is not found.
.WINDOWNAME=<Window Name>	Name of the window to delete. Required when the WINDOWDELETE command is used.

Command	Description
.WINDOWRENAME	Renames a window. If the 'old' name cannot be found, the command issues a warning. If the new name exists, an error message is generated. If the window cannot be renamed, an error occurs. If the old name matches the new name, then nothing is done.
.OLDWINDOWNAME=<Existing Window Name>	Current window name. Required when the WINDOWRENAME command is used.
.NEWWINDOWNAME=<New Window Name>	New window name. A window with this name must not exist. Required when the WINDOWRENAME command is used.
.PRINTAPPLICATIONINFORMATION	Prints or dumps the InTouch application data set to a text file. This is the same as selecting the **Print** option on the **File** menu commands from WindowMaker.
.OUTPUTTARGET=Printer|TextFile	Sends output to a printer or text file.
.OUTPUTTARGETNAME=<Printer Name>|<Full Text File Path>	Name of the printer or output file. If the file exists, it is overwritten.
.CROSSREFERENCE	Generates cross-reference information in Comma Separated Variable (CSV) format.
.SEARCHFOR= TagName|QuickFunctions	Searches for tags or QuickFunctions by name.
. REFERENCETYPE= ByTagName|ByWindow	Cross-references by tag or window name.
. OUTPUTFILE=<Full File Path>	Full path to the output file. If the file exists, it is overwritten.

Create an application

You can use a WindowMaker command file to create a new default InTouch application. You create a blank application using the minimum command file. However, you cannot create a blank managed InTouch application using the minimum command file. For more information, see Create a minimum command file.

The command file must be placed in the folder where you want the application created. If the folder contains an existing InTouch.ini file, the application is not generated.

The path to the new application folder cannot contain an embedded "-l" or "-L" string sequence. For example, the folder `C:\MyApps\App-Large` cannot be created.

The created InTouch.ini file has contents similar to the following example. Window positions vary with the screen resolution in which the application is shown. The application has the title and description "Generated InTouch

Application." The default language is English.

Example InTouch.ini file contents:

```
[InTouch]
AppMode=2
AppName0=Generated InTouch Application
AppName1=
AppName2=
AppName3=
AppDesc0=Generated InTouch Application
AppDesc1=
AppDesc2=
AppDesc3=
LanguageBase=English (United States)
LanguageBaseID=1033
InTouchView=0
SAOConverted=1
WinFullScreen=1
WinLeft=-4
WinTop=-4
WinWidth=1288
WinHeight=1004
SnapOn=1
```

Add tags to newly generated application

A new InTouch application contains only system tags. If you need to add tags to an application before importing a window, then:

1. Create a new application.
2. Run DBLoad to import tags to the new application.
3. Import the window.

For example:

```
WM.Exe C:\MyApps\App001 COMMANDFILE="C:\BlankFile.Txt"
DBLoad C:\MyApps\App001,C:\TagDumps\App001.Csv,0
WM.Exe C:\MyApps\App01 COMMANDFILE="C:\Commands.Txt"
```

Delete a window

If you add a window to an existing application that has a window with the same name, you must delete the existing window. You can delete the existing window using the WindowMaker command file.

Important: An error message does not appear if the window selected to delete does not exist.

A message appears in the Logger if an existing window cannot be deleted. No user interaction is required.

You enter a sequence of commands in your command file to delete a window from an InTouch application. A command file can include several delete window command sequences.

The following example shows the command sequence to delete a window from an InTouch application:

```
.WINDOWMAKERCOMMANDFILE
.VERSION=1
```

```
.WINDOWDELETE
.WINDOWNAME=Window002
.GO
```

Rename a window

You can change the name of a window in an InTouch application.

You enter a sequence of commands in your command file to rename a window from an InTouch application. A command file can include several rename window command sequences.

An error message does not appear if the window selected to be renamed does not exist.

If the new window name already exists, nothing is done, and a warning message is logged.

If the old name exists and the new name does not exist, but the rename fails for some other reason, an error message is logged.

No user interaction is required. Warnings and error messages appear in the Logger.

If you rename a window with the name of an existing window, a warning message is logged in the log file or the Logger. In the following example, renaming the Window005 to an existing window Window006, logs a warning message in the log file or the Logger.

```
.WINDOWRENAME
.OLDWINDOWNAME=Window005
.NEWWINDOWNAME=Window006
.GO
```

Import a window

You can import a window into an InTouch application. The full file path to the XML file containing a single InTouch window's specification is required. The new window name is contained within the XML file.

The window is not imported if:

- A window with the same name exists in the application.
- An unresolved error occurs in one of the window scripts.
- An unresolved error occurs trying to add any part of an object to the window.

User interaction may be required if:

- Elements specified in scripts or expressions contain errors or are missing.
- Scripts contain syntax errors.

It is possible that nothing is shown to allow a user to correct the problem. Examine the log file and the Logger for details.

You enter a sequence of commands in your command file to import a window.

For example:

```
.WINDOWCREATE
.XMLFILEPATH=C:\WMCommandTest\WMCreateFile.Xml
.GO
```

A command file may contain more than one import window command and several create window command

sequences.

Handle errors

Your XML file must follow general XML formatting rules. If you cannot open your XML file with Internet Explorer, your XML file contains XML formatting errors. Fix all errors before using the file with WindowMaker.

If you use a file containing general format errors, WindowMaker logs the error message "XML file could not be loaded" in the Logger.

WindowMaker stops when any element is missing from a script, on any script error, or any other element specification error occurs. For example:

- Missing tags
- Missing external WindowMaker script extension DLL
- Missing ActiveX controls
- Missing Wizard DLL

In some cases, such as for particular animation links or custom property overrides, no message box appears if the tag is missing. A note is written to the Logger, the animation link and object are not created, and the window is not created. In other cases, the script and expression parsing stops. You are given an opportunity to create the tag, and if it is successfully created, processing continues.

Missing SmartSymbols

If a SmartSymbol template does not exist in the target application, the window is not imported. An error message appears in the logger and the output text file.

If a SmartSymbol instance fails to be created for any reason, the window is not created. A SmartSymbol import can fail even though the SmartSymbol template exists. There may be additional information about the failure in the Logger.

Missing Industrial Graphics

If an Industrial Graphic reference does not exist in the Galaxy repository, the window is not created. An error message appears in the Logger and the output text file.

The XML import process can fail even though the Industrial Graphic reference exists. Additional information about the failure is logged in the Logger.

Expressions, tag names, and scripts

When expressions, tag names, and scripts are required, they cannot be empty text, or text that is only blanks. If these text items are not formulated correctly; the expression, tag name, or script do not parse correctly and result in an error.

In the case of tag names, the tag type must match the type expected for an object. In most cases, it is possible to use dotfields to access tag properties. For example, the second bit of an integer tag iTag005 can be accessed as:

```
iTag005.02
```

In this case, the tag and dotfield evaluates to a discrete value and can be used where a discrete tag name is required, whereas the iTag005 alone would result in an error.

Run WindowMaker from the command prompt

You can run WindowMaker from the command prompt.

The command line is:

```
WM.EXE AppPath,Commandfile="Command.txt"
```

Parameters

AppPath

Defines the full path to the InTouch application. This parameter is optional if you are running WindowMaker in the application's folder.

CommandFile

Defines the full path to the command file. Quotation marks must enclose the path name. Extra spaces are not allowed around the equal sign.

Suppose the InTouch application is in the `C:\MyApps` folder, and the command file is `C:\WMCommandFile.txt`. The following example shows the WindowMaker command you enter from the command prompt:

```
WM.EXE C:\MyApps,COMMANDFILE="C:\WMCommandFile.Txt"
```

Note: You cannot create managed InTouch applications from the command prompt.

System Platform IDE extension

Use the System Platform IDE extension to run the InTouch XML import functionality for managed applications. You can use the IDE extension to select a command file for the XML import process. After you select a command file, WindowMaker starts up and the associated XML file is parsed.

Important: WindowMaker stops responding if you start a new console session or maximize an existing console session when the XML import is in progress.

The System Platform IDE has a context menu item used to select a command file for the XML import. The menu item **Process InTouch Command File** is shown in the context menu when you right-click an InTouchViewApp derived template. However, the menu item will not be shown, when you select:

- Multiple InTouchViewApp templates
- InTouchViewApp base template
- InTouchViewApp instance

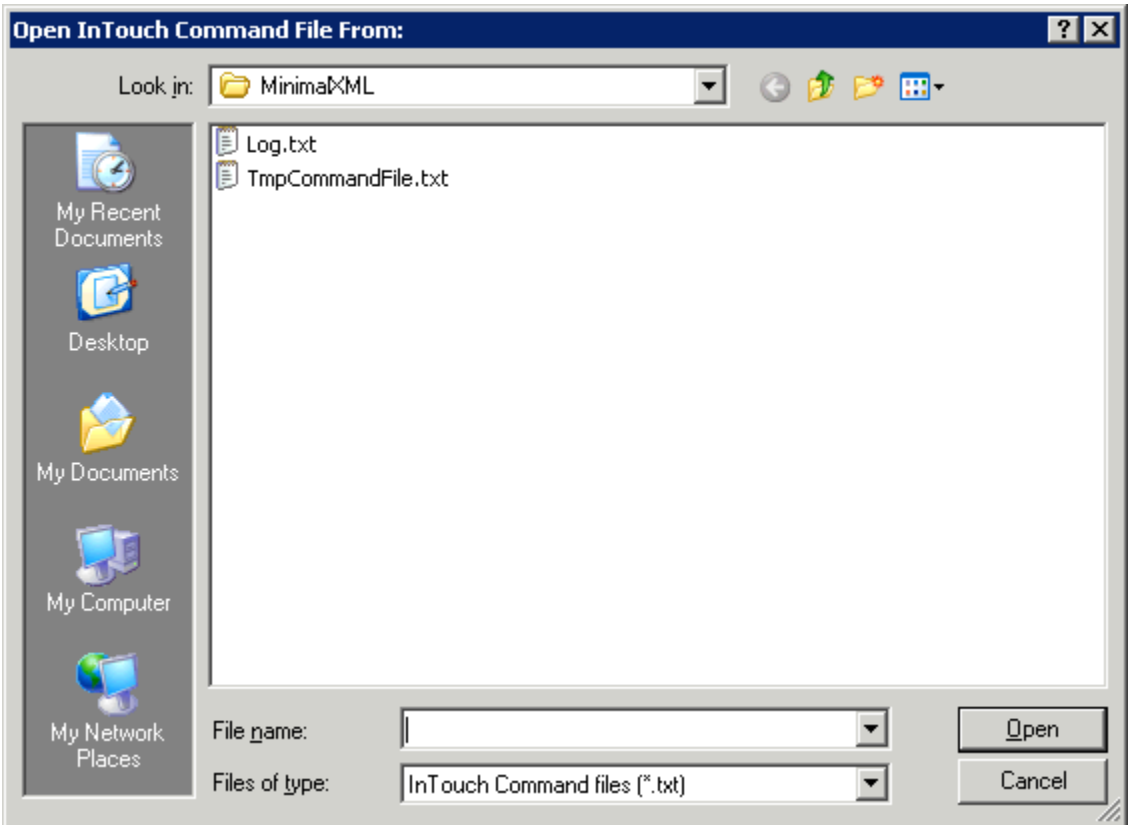
If you migrate a managed InTouch application from older version of InTouch, then you must open the InTouch application in WindowMaker at least once before processing the command file.

Run WindowMaker from managed InTouch applications

You can run WindowMaker from a managed InTouch application.

Run WindowMaker from a managed InTouch application

1. Open the System Platform IDE.
2. In the **Template Toolbox**, create a derived template from the InTouchViewApp template.
3. Associate an InTouch application with the derived template by creating a new InTouch application or by importing an existing standalone InTouch application.
4. Right-click the derived template and then select **Process InTouch Command File**. The **Open InTouch Command File From:** dialog box appears.



Note: The InTouchViewApp derived template must be configured in WindowMaker at least once, before executing the **Process InTouch Command File** command.

5. Browse to the location of the command file, and select **Open**. The WindowMaker starts up.

Run DBDump from the command prompt

You use DBDump to extract tag information from an InTouch application. You can run DBDump from the command prompt. You must stop WindowMaker before running DBDump.

Commas are required between DBDump command parameters. Parameters are position dependent. If you want to omit a parameter between included parameters, you must include a comma for the missing parameter. The following example shows the syntax of the DBDump command when you run it from the command prompt:

```
DBDump AppPath,CsvPath,GroupTypes,OverwriteCsvFile, MessageBoxes
```

Parameters

AppPath

Defines the path to the InTouch application.

CsvPath

Defines the path to the export file containing tag definitions from the Tagname Dictionary of the InTouch application.

GroupTypes

Specifies whether tags are grouped in the DBDump export file by InTouch tag types. 1 indicates the tag database names should be sorted by tag group type. 0 indicates the tag names are not to be sorted by type.

OverwriteCsvFile

Specifies if the export file should be overwritten. 1 indicates the export file should be overwritten. 0 indicates the export file should not be overwritten.

MessageBoxes

Specifies whether messages appear when the DBDump utility exports the contents of the application's Tagname Dictionary. 1 indicates the message boxes are to be displayed. 0 indicates message boxes are not to be displayed.

Example

Suppose the InTouch application is located in the `C:\MyInTouchApps\App001` folder and you want to write the contents of the tag database to the `C:\TagDumps\App001.csv` file. You do not want any message boxes to appear, and you want to overwrite the target export file if it exists. The command is:

```
DBDump C:\MyInTouchApps\App001,C:\TagDumps\App001.Csv,1,1,0
```

Run DBDump for managed InTouch applications

You can run DBDump for managed InTouch applications.

Run DBDump for a managed InTouch application

1. Open the System Platform IDE.
2. In the **Template Toolbox**, right-click an InTouchViewApp derived template, point to **Export**, and then choose **Selected as CSV**.

   The **Exporting automation objects to CSV** dialog box appears.
3. Type the name of the CSV file and select **Save**. The application tag data is successfully dumped into the CSV file.

Run DBLoad from the command prompt

You use DBLoad to import tag information into an InTouch application. You can run DBLoad from the command prompt. You must stop WindowMaker before running DBLoad.

DBLoad command parameters are position dependent, if you want to omit a parameter between included parameters, you must include a comma for the missing parameter. The following example shows the syntax of

the DBLoad command when you run it from the command prompt:

```
DBLoad AppPath,CsvPath,MessageBoxes
```

Parameters

AppPath

Specifies the path to the InTouch application.

CsvPath

Specifies the path to the file containing tag definitions to be imported into an application's Tagname Dictionary.

MessageBoxes

Specifies whether messages appear when the DBLoad utility imports the contents of the import file into the application's Tagname Dictionary. 1 indicates the message boxes are to be displayed. 0 indicates message boxes are not to be displayed.

Example

Suppose the InTouch application is located in the `C:\MyInTouchApps\App001` folder and you want to read the tag database information from the `C:\TagDumps\App001.Csv` file. You do not want the message boxes to appear. The following example shows the DBLoad command you enter at the command prompt:

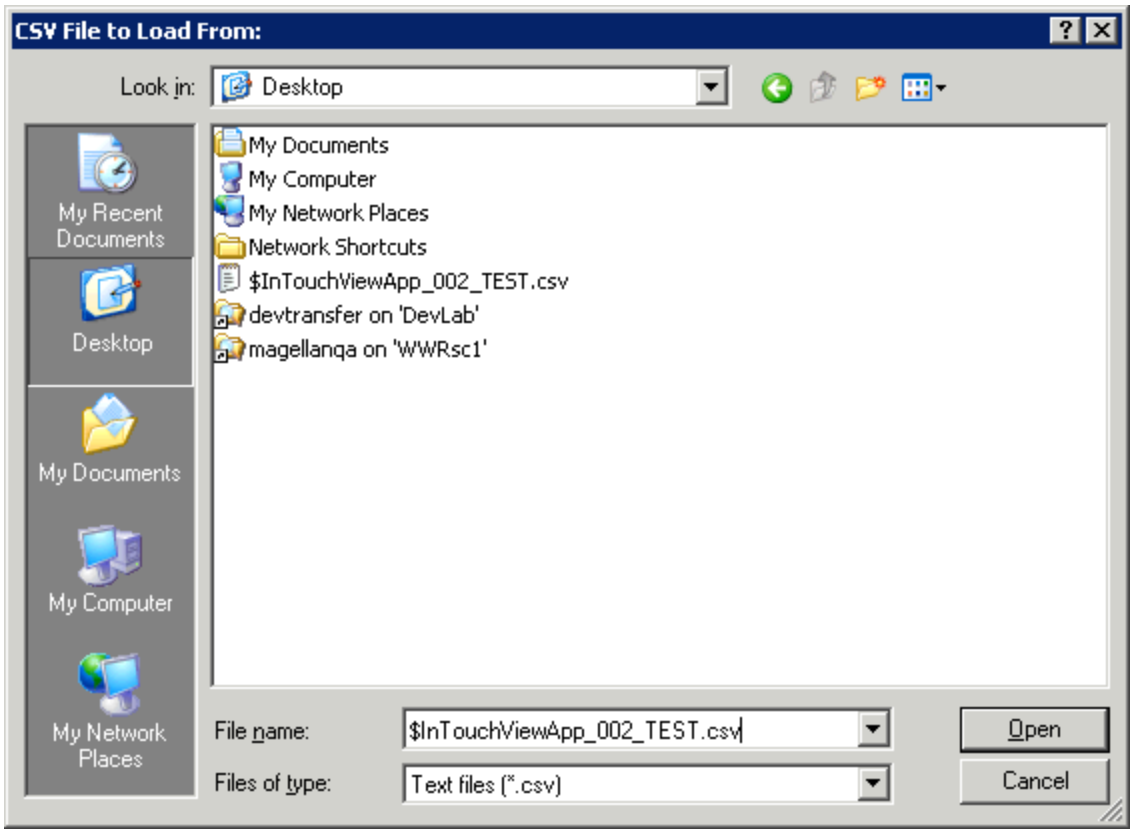
```
DBLoad C:\MyInTouchApps\App001,C:\TagDumps\App001.Csv,0
```

Run DBLoad for managed InTouch applications

You can run DBLoad for managed InTouch applications.

Run DBLoad for a managed InTouch application

1. Open the System Platform IDE.
2. In the **Template Toolbox**, right-click an InTouchViewApp derived template, point to **Import**, and then select **DB Load**. The **CSV File to Load From:** dialog box appears.



3. Browse to locate the CSV file and select **Open**. The tag data is successfully loaded into the InTouch application.

XML formats

You can import a window and most window elements defined in an XML file. Tags cannot be imported from an XML file. You must use the DBLoad utility instead.

General XML file format

Your XML file must comply with all general XML formatting rules. You should be able to open your XML file with Internet Explorer. If not, your XML file contains formatting errors.

Use a schema

You can validate the XML file against the schema file. Validating your XML file with the schema file detects most XML formatting errors.

The schema is more restrictive than the XML input parser.

- The schema requires the order of elements to match the order listed in the tables.
- Element names and values are case sensitive.
- In some cases, the schema requires an element to be explicitly defined.

You invoke schema validation in the window element. For information about how to specify a schema, see Window definition.

XML file header

The XML declaration must be included at the top of the file. The minimal declaration is:

```
<?xml version="1.0"?>
```

Extraneous XML elements

The InTouch XML import functionality does not generate errors or warnings for undefined elements, attributes, or nodes.

You must type everything correctly. If you mean to have an <ONRIGHTDOWN> action script, but accidentally type <ONRITEDOWN>, no warning is generated, and your action script is not added to the window object.

Warnings and errors are created for XML definitions that are missing required elements.

User-supplied text

If your script uses XML field delimiter characters, the script text must be encapsulated within a CDATA element. Text in a CDATA element is not parsed by the XML file import functionality.

You can enter text without enclosing it within a CDATA element provided the text does not contain any XML field delimiter characters.

Preserve text white space

When text not in a CDATA element is processed, the leading and trailing spaces are removed. As a result, an element such as <Title> MyWindowName</Title> results in a window name of 'MyWindowName' without leading spaces.

To preserve leading and trailing white space in any text, enclose the text within a CDATA element. For example, an element <Title><![CDATA[MyWindowName]]></Title> results in a window name of "MyWindowName".

Common element definitions

Some elements or definitions are shared by many elements.

Color elements

Color elements can be specified by RGB value, name, reference value, or integer value. The color elements are R, G, B, Name, Ref, and Value. These elements are used in other elements. For example, FillColor, TextColor, and BGCOLOR.

RGB elements

You use RGB values to specify a color. The values assigned to RGB elements range from 0 to 255. When an

element is missing, a default is used. The default depends on the window object.

Examples:

```
<FillColor>
<R>192</R>
<G>192</G>
<B>192</B>
</FillColor>
<TextColor> <R>0</R><G>0</G><B>0</B> </TextColor>
```

Color name elements

You can specify a color using a name supported by Internet Explorer version 3.0 or later. The color name is case sensitive and must match a known HTML color.

The following web sites have a list of the color names and values:

http://www.w3schools.com/html/html_colornames.asp

http://www.learningwebdesign.com/colornames.html

http://www.oreilly.com/catalog/wdnut/excerpt/color_names.html

The following web sites have a list of the color names:

http://www.geocities.com/SiliconValley/Pines/6986/colortbl.html

```
Examples:
<FillColor>
<Name>White</Name>
</FillColor>
<TextColor> <Name>White</Name> </TextColor>
```

Color reference elements

You can specify a color using a hexadecimal color value. The color value must always contain six hexadecimal digits.

The following web sites list colors by name and their corresponding hexadecimal values:

http://www.w3schools.com/html/html_colornames.asp

http://www.learningwebdesign.com/colornames.html

This web site has a list of non-dithering color values:

http://www.htmlgoodies.com/tutorials/colors/article.php/3479001

This site has 4096 colors and their hex codes:

http://yorktown.cbe.wwu.edu/sandvig/MIS314/Assignments/A03/ColorHexCodes.asp

Examples:

```
<FillColor>
<Ref>#FF00FF</Ref>
</FillColor>
<TextColor> <Ref>#FF00FF</Ref> </TextColor>
```

Color value elements

You can specify a color using a positive integer color value. The value must be in the range 0 to 16777215.

Examples:

```
<FillColor>
<Value>16711935</Value>
</FillColor>
<TextColor> <Value>16711935</Value> </TextColor>
```

TextInfo elements

You can specify how text appears on the screen with the TextInfo element.

The following elements are available for specifying text:

Elements	Description
Font	Name of font families, such as 'System' or 'Arial'.
FontStyle	Values can be {Regular, Italic, Bold, BoldItalic}.
FontSize	Need to indicate font size unit of measure. Usually, points. Values can be 0 or larger.
Underline	Underlined text: {true or false}.
Strikeout	Struck out text: {true or false}.
TextColor	Color element for text color.
TextJustify	Text justification: {Left, Center, or Right}

Example:

```
<TextInfo>
<Font>Arial</Font>
<FontStyle>Regular</FontStyle>
<FontSize>12</FontSize>
<Underline>false</Underline>
<Strikeout>false</Strikeout>
<TextColor>
<R>0</R>
<G>0</G>
<B>0</B>
</TextColor>
<TextJustify>Left</TextJustify>
</TextInfo>
```

Point elements

You use a point element to define the position of other elements. Point elements contain two elements, X and Y. X and Y elements must contain values in the range -32000 and 32000.

Elements	Description
X	Left coordinate in pixels. Required.
Y	Top coordinate in pixels. Required

Example:

```
<Point>
<X>10</X>
<Y>25</Y>
</Point>
```

Pen elements

You use pen elements to specify the line characteristics of an object's border.

The following elements are available for specifying the pen element:

Field	Description
PenColor	Uses color elements: RGB element, name element, reference element, or value element.
PenWidth	Pen width in pixels. Values can be 1, 2, 4, 6, 9, or 11.
PenStyle	Values can be None, Solid, Dash, Dot, DashDot, DashDotDot.

Example:

```
<Pen>
<PenColor>
<R>0</R>
<G>0</G>
<B>0</B>
</PenColor>
<PenWidth>4</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
```

Dimension elements

The dimension element contains elements to specify the coordinates of the upper left corner of an object on the screen. The dimension element also includes elements to specify the width and height of a rectangular object.

In WindowMaker, the coordinate limits are -32000 to +32000 in both the vertical and horizontal directions. If you specify a combination of X and Y location with width or height values that place the calculated coordinates outside of a (-32000, -32000, 32000, 32000) boundary, a warning appears and the values are clamped to the maximum values. Height and width values should be positive.

You use the following elements within a dimension element to specify an area on the screen.

Elements	Description
Left	Left edge coordinate. Required.
Top	Top edge coordinate. Required.
Width	Width in pixels.
Height	Height in pixels.

Example:

```
<Dimension>
<Left>4</Left>
<Top>4</Top>
<Width>632</Width>
<Height>278</Height>
</Dimension>
```

Expressions

You enclose expression text in CDATA sections. This prevents XML delimiters, which are valid in expressions, from causing file parsing to fail.

Example:

```
<EXPRESSION>
<![CDATA[
First line of expression text
Second line of expression text
N-th line of expression text
]]>
</EXPRESSION>
```

Virtual key codes and virtual key flags

Some InTouch animation links support keyboard input.

The XML parser translates the name of a virtual key into its virtual key code. Also, the flags for the modifier keys are specified using text instead of numeric bit combinations. Key names are not case sensitive.

InTouch applications can use the virtual key names listed in the following table.

Key Represented	Virtual Key Names
ADD	Add
Alpha Keys	A through Z
BACKSPACE	Backspace
CANCEL	CtrlBreak
CLEAR	Clear

Key Represented	Virtual Key Names
Copy	Copy
Decimal	Decimal
DELETE	Delete
Divide	Divide
DOWN ARROW	Down
Empty string means no assignment.	<Blank>
END	End
ENTER	Return
ESC	Escape
Execute	Execute
F1	F1
F2	F2
F3	F3
F4	F4
F5	F5
F6	F6
F7	F7
F8	F8
F9	F9
F10	F10
F11	F11
F12	F12
F13	F13
F14	F14
F15	F15

Key Represented	Virtual Key Names
F16	F16
HELP	Help
HOME	Home
INSERT	Insert
LEFT ARROW	Left
MULTIPY	Multiply
Numeric keys	1 through 9
Numeric Keypad 0	NUMPAD0
Numeric Keypad 1	NUMPAD1
Numeric Keypad 2	NUMPAD2
Numeric Keypad 3	NUMPAD3
Numeric Keypad 4	NUMPAD4
Numeric Keypad 5	NUMPAD5
Numeric Keypad 6	NUMPAD6
Numeric Keypad 7	NUMPAD7
Numeric Keypad 8	NUMPAD8
Numeric Keypad 9	NUMPAD9
NUM LOCK	NumLock
PAGE UP	PageUp
PAGE DOWN	PageDown
PRINT SCREEN	Print
RIGHT ARROW	Right
SELECT	Select
Separator	Separator
SPACEBAR	Space

Key Represented	Virtual Key Names
SUBTRACT	Subtract
TAB	Tab
UP ARROW	Up

Two modifier keys are supported. Modifier keys can be specified separately or in combination for the CKEYFLAGS element. The modified name is the element value. A modifier key is applied as long as the modifier name is in the attribute value string.

Name	Key Represented
CTRL	CONTROL key must be pressed with regular key.
SHIFT	SHIFT key must be pressed with regular key.
CTRL + SHIFT	Both the CONTROL and SHIFT keys must be pressed.

To generate a CTRL+A sequence, the XML is:

```
<VirtualKeyType>
   <KeyCode>A</KeyCode>
   <KeyFlags>CTRL</KeyFlags>
</VirtualKeyType>
```

Window element

You can include only a single window element in your XML file. The window element must be the first element specified in the XML file.

The window name must not match an existing window name in the InTouch application.

Window definition

You can use the following elements to specify a window.

Elements	Description
Title	Window name as non-empty string. Maximum of 32 characters and trailing spaces are not counted. Required.
Comment	Comment text. Maximum of 59 characters.

Elements	Description
Dimension	Required when using the optional schema. When the schema is not used, a default dimension is applied if a dimension element is not included. The default values are 4 4 632 278.
WindowStyle	Window Type = {Replace | Overlay| Popup}.
BackgroundColor	Window background color specified by a color element: RGB element, name element, reference element, or value element.
FrameStyle	{Single | Double | None}.
TitleBar	Title bar enabled = {true | false}.
CloseButton	Close window button enabled = {true | false} Can be enabled only when title bar is enabled.
SizeControls	Size controls enabled = {true | false}.
ScriptOnShow	Script element.
ScriptWhileShowing	Script element.
ScriptOnHide	Script element.
ObjectList	Objects contained by the window, including SmartSymbols.

This example creates an empty window:

```
<?xml version="1.0" encoding="UTF-8"?>
<iw:InTouchWindow
xmlns:iw="http://www.wonderware.com/InTouch/Window"
xmlns:itc="http://www.wonderware.com/InTouch/Common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wonderware.com/InTouch/Window
wwInTouchWindow.xsd" Version="1">
<Title>ApplicationSecondWindow</Title>
<Comment>Main application window</Comment>
<Dimension>
<Left>4</Left>
<Top>4</Top>
<Width>632</Width>
<Height>278</Height>
</Dimension>
<BackgroundColor>
<R>192</R>
```

```
<G>192</G>
<B>192</B>
</BackgroundColor>
<WindowStyle>Overlay</WindowStyle>
<FrameStyle>Single</FrameStyle>
<TitleBar>true</TitleBar>
<CloseButton>True</CloseButton>
<SizeControls>true</SizeControls>
</iw:InTouchWindow>
```

Minimal example:

```
<iw:InTouchWindow
xmlns:iw="http://www.wonderware.com/InTouch/Window"
xmlns:itc="http://www.wonderware.com/InTouch/Common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wonderware.com/InTouch/Window
wwInTouchWindow.xsd" Version="1">
<Title>ApplicationSecondWindow</Title>
<Dimension><Left>4</Left><Top>4</Top>
   <Width>632</Width>
<Height>278</Height></Dimension>
</iw:InTouchWindow>
```

Activate schema validation

The schema validation and processing are activated by placing specific data within the InTouchWindow element. If you activate the schema, you must place the InTouchCommon.Xsd and InTouchWindow.Xsd files in the same folder as the XML file.

Example of using a schema:

```
<iw:InTouchWindow
xmlns:iw="http://www.wonderware.com/InTouch/Window"
xmlns:itc="http://www.wonderware.com/InTouch/Common"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wonderware.com/InTouch/Window wwInTouchWindow.xsd"
Version="1">
</iw:InTouchWindow>
```

Example of not using a schema:

```
<InTouchWindow Version="1">
</InTouchWindow>
```

Window scripts

You can create three types of window scripts in your XML import file: OnShow, WhileShowing, and OnHide. You place the script elements within the window element.

Script text can be enclosed in a CDATA section to prevent XML delimiter characters in the script from interfering with XML file parsing.

OnShow window script element

The OnShow script element contains the script text.

Example:

```
<ScriptOnShow><![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]></ScriptOnShow>
```

Minimal example:

```
<ScriptOnShow>Single line of script text</ScriptOnShow>
```

WhileShowing window script element

The WhileShowing script has two elements. The script text can be placed in a CDATA section.

Elements	Description
Text	Text of the window script. Required.
FREQUENCY	Script execution frequency in milliseconds. Required when using the optional schema. Default is 1000.

Example:

```
<ScriptWhileShowing>
<Text><![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]></Text>
<Frequency>1000</Frequency>
</ScriptWhileShowing>
```

Minimal example:

```
<ScriptWhileShowing>
<Text>Single line of script text</Text>
<Frequency>1000</Frequency>
</ScriptWhileShowing>
```

OnHide window script element

The OnHide window script has one element. The script text can be placed in a CDATA section.

Example:

```
<ScriptOnHide>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</ScriptOnHide>
```

Minimal example:

```
<ScriptOnHide>Line of script text</ScriptOnHide>
```

Window objects

Window objects are placed in an object list defined by the ObjectList element.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<iw:InTouchWindow xmlns:iw="http://www.wonderware.com/InTouch/Window"
xmlns:itc="http://www.wonderware.com/InTouch/Common" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.wonderware.com/InTouch/Window.xsd"
Version="1">
<Title>Window0001</Title>
<Comment>Sample simple window</Comment>
<Dimension>
   <Left>10</Left>
   <Top>10</Top>
   <Width>400</Width>
   <Height>400</Height>
</Dimension>
<BackgroundColor>
   <R>255</R>
   <G>0</G>
   <B>255</B>
</BackgroundColor>
<WindowStyle>Replace</WindowStyle>
<FrameStyle>Double</FrameStyle>
<TitleBar>true</TitleBar>
<SizeControls>true</SizeControls>
<ObjectList>
   <Rectangle>
      <Title>Rectangle1</Title>
      <Pen>
         <PenColor>
            <Name>Black</Name>
         </PenColor>
         <PenWidth>4</PenWidth>
         <PenStyle>Solid</PenStyle>
      </Pen>
      <Dimension>
         <Left>100</Left> <Top>50</Top>
         <Width>270</Width> <Height>80</Height>
      </Dimension>
      <FillColor>
         <R>128</R> <G>128</G> <B>128</B>
      </FillColor>
   </Rectangle>
</ObjectList>
</iw:InTouchWindow>
```

Each window object can contain animation links. A window object containing animation links includes an animation links element.

```
<AnimationLinks> </AnimationLinks>
```

Default element values

When window object elements are not specified, the element values in the following table are used as defaults.

Elements	Default Value
FILLCOLOR	<R>212</R><G>208</G> <B>200</B>
PENWIDTH	1
PENCOLOR	Black
PENSTYLE	Solid
CORNERDIMENSION	<Width>20</Width> <Height>20</Height>
TEXTCOLOR	Black
TEXTJUSTIFY	Left
ROTATION	0
FONT	System
FONTSTYLE	regular
FONTWEIGHT	Ignored
FONTSIZE	10
UNDERLINE	False
STRIKEOUT	False
FLIP	None
TRANSPARENT	<R>0</R><G>255</G> <B>0</B>

Pen style limitations

If you specify a pen style other than solid, the pen width is forced to 1. To specify pen widths larger than 1, the pen style option must be removed or set to SOLID.

Pen dimensions

The dimension specification is for the centerline of an object. If an object's pen width is larger than 1, then an object may not fit within the specified dimension. Instead, the pen width straddles the boundary of the object. Some pixels are inside the object's boundary; others are outside of the boundary.

Rectangle object

The following elements can be specified for a rectangle element.

Elements	Description
FillColor	Fill color of a rectangle. Specified with a color element. Default is rgb(212, 208, 200).
Pen	Pen element.
Dimension	Location and size of an object. Elements are top, left, width, height. Top, left, width, and height values are pixels. The resulting coordinates must be within -32000 and +32000. Both width and height cannot be zero. Required. An object is not created if the Dimension element is assigned invalid values or is missing values.
Title	Object name. Optional.
AnimationLinks	Optional animation links list.

Example:

```
<Rectangle>
<Title>Rectangle1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>4</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Dimension>
<Left>100</Left> <Top>50</Top>
<Width>270</Width> <Height>80</Height>
</Dimension>
<FillColor>
<R>128</R> <G>128</G> <B>128</B>
</FillColor>
<AnimationLinks> </AnimationLinks>
</Rectangle>
```

Minimal example:

```
<Rectangle>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
</Rectangle>
```

Rounded rectangle object

The following elements can be specified for a rounded rectangle object.

Elements	Description
FillColor	Fill color. Specified by a color element. Default is rgb(212, 208, 200).
Pen	Pen element.
Dimension	Location and size of an object. Elements are top, left, width, height. Top, left, width, and height values are pixels. The resulting coordinates must be within -32000 and +32000. Both width and height cannot be zero. Required. An object is not created if the Dimension element is assigned invalid values or is missing values.
CornerDimension	Elements width and height. Corner width cannot exceed the width of the rectangle. Corner height cannot exceed the height of the rectangle. Corner width and corner height must be at least 1. Default is 20, 20.
Title	Object name. Optional.
AnimationLinks	Optional animation links list.

Example:

```
<RoundedRectangle>
<Title>RoundedRectangle1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>4</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Dimension>
<Left>100</Left>
<Top>50</Top>
<Width>270</Width>
<Height>80</Height>
</Dimension>
<FillColor>
<R>128</R> <G>128</G> <B>128</B>
</FillColor>
<AnimationLinks>
</AnimationLinks>
<CornerDimension>
<Width>8</Width>
<Height>8</Height>
```

```
</CornerDimension>
</RoundedRectangle>
```

Minimal example:

```
<RoundedRectangle>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
</RoundedRectangle>
```

Ellipse object

The following elements can be specified for an ellipse object.

Elements	Description
FillColor	Fill color. Specified with a color element. Default is rgb(212, 208, 200).
Pen	Pen element.
Dimension	Location and size of an object. Elements are top, left, width, height. Top, left, width, and height values are pixels. The resulting coordinates must be within -32000 and +32000. Both width and height cannot be zero. Required. An object is not created if the Dimension element is assigned invalid values or is missing values.
Title	Object name. Optional.
AnimationLinks	Optional animation links list.

Example:

```
<Ellipse>
<Title>Ellipse1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>4</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Dimension>
<Left>100</Left>
<Top>50</Top>
<Width>270</Width>
<Height>80</Height>
</Dimension>
<FillColor>
<R>128</R> <G>128</G> <B>128</B>
```

```
</FillColor>
<AnimationLinks>
</AnimationLinks>
</Ellipse>
```

Minimal example:

```
<Ellipse>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
</Ellipse>
```

Line object

The following elements can be specified for a line object. From and to points cannot be the same point.

Elements	Description
Pen	Pen element
Title	Object name. Optional.
AnimationLinks	Optional animation links list.
Points	Point elements. Must contain two. Extra point elements are ignored.

Example:

```
<Line>
<Title>Line1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>1</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>150</X><Y>160</Y></Point>
</Points>
<AnimationLinks>
</AnimationLinks>
</Line>
```

Minimal example:

```
<Line>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>150</X><Y>160</Y></Point>
</Points> </Line>
```

Horizontal line object

The following elements can be specified for a horizontal line object. This results in a WindowMaker HV Line

object.

Elements	Description
Pen	Pen element.
Title	Object name. Optional.
AnimationLinks	Optional animation links list.
Points	Must contain two points. The Y coordinate of the second point is ignored and set to the Y coordinate of the first point. Extra point elements are ignored. Required. An object is not created if the Points element is assigned invalid values or is missing values.

Example:

```
<HorizontalLine>
<Title>Line1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>1</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>150</X><Y>150</Y></Point>
</Points>
<AnimationLinks>
</AnimationLinks>
</HorizontalLine>
```

Minimal example:

```
<HorizontalLine>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>150</X><Y>150</Y></Point>
</Points>
</HorizontalLine>
```

Vertical line object

The following elements can be specified for a vertical line object. This will result in a WindowMaker HV Line object.

Elements	Description
Pen	Pen element.
Title	Object name. Optional.

Elements	Description
AnimationLinks	Optional animation links list.
Points	Must contain two points. The X coordinate of the second point is ignored and set to the X coordinate of the first point. Extra point elements are ignored. Required. An object is not created if the Points element is assigned invalid values or is missing values.

Example:

```
<VerticalLine>
<Title>Line1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>1</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>50</X><Y>250</Y></Point>
</Points>
<AnimationLinks>
</AnimationLinks>
<VerticalLine>
```

Minimal example:

```
<VerticalLine>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>50</X><Y>250</Y></Point>
</Points>
<VerticalLine>
```

Polyline object

You must define at least two points in order to load the polyline object. Using the same coordinates for the two points is not recommended.

The following elements can be specified for a polyline object.

Elements	Description
Points	Point elements, at least two required. An object is not created if the Points element is assigned invalid values or is missing values.
Pen	Pen element.

Elements	Description
Title	Object name. Optional.
AnimationLinks	Optional animation links list. Fill Color, Text Color, Percent Fill, Value Display are not permitted.

Example:

```
<Polyline>
<Title>polyline1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>1</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>60</X><Y>250</Y></Point>
<Point><X>70</X><Y>350</Y></Point>
<Point><X>80</X><Y>450</Y></Point>
</Points>
<AnimationLinks>
</AnimationLinks>
</Polyline>
```

Minimal example:

```
<Polyline>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>80</X><Y>450</Y></Point>
</Points>
</Polyline>
```

Polygon object

You must define at least two points to load a polygon object. Using the same coordinates for the two points is not recommended.

The following elements can be specified for a polygon object.

Elements	Description
Points	Contains Point elements. At least two points are required. X and Y values are in pixels.
FillColor	Fill color of polygon object. Contains a color element. Default is rgb(212, 208, 200).
Pen	Pen element.
Title	Object name. Optional.

Elements	Description
AnimationLinks	Optional animation links list. Text Color, Value Display animation links are not permitted.

Example:

```
<Polygon>
<Title>polygon1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>1</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>60</X><Y>250</Y></Point>
<Point><X>70</X><Y>350</Y></Point>
<Point><X>80</X><Y>450</Y></Point>
</Points>
<FillColor>
<R>128</R> <G>128</G> <B>128</B>
</FillColor>
<AnimationLinks>
</AnimationLinks>
</Polygon>
```

Minimal example:

```
<Polygon>
<Points>
<Point><X>50</X><Y>150</Y></Point>
<Point><X>80</X><Y>450</Y></Point>
</Points>
</Polygon>
```

Text object

The following elements can be specified for a text object.

Elements	Description
Title	Object name. Optional.
TextString	Displayed text string. Required. Text object is not created if the Textstring element is assigned an invalid value or is missing a value.
TextInfo	Defines how the text will be shown.

Elements	Description
Rotation	Defines orientation of the text in degrees. Possible values:{0, 90, 180, 270}. Default is 0.
AnimationLinks	Optional animation links list.
Dimension	Location and size of an object. Elements are top, left, width, height. Top, left, width, and height values are pixels. The resulting coordinates must be within -32000 and +32000. Both width and height cannot be zero. Required. Object is not created if the Dimension element is assigned invalid values or is missing values.

Example:

```
<Text>
<Title>text1</Title>
<TextString>This is some text to display</TextString>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
<TextInfo>
<Font>Arial</Font>
<FontStyle>Regular</FontStyle>
<FontSize>12</FontSize>
<Underline>false</Underline>
<Strikeout>false</Strikeout>
<TextColor>
<R>0</R>
<G>0</G>
<B>0</B>
</TextColor>
<TextJustify>Left</TextJustify>
</TextInfo>
<Rotation>0</Rotation>
<AnimationLinks>
</AnimationLinks>
</Text>
```

Minimal example:

```
<Text>
<TextString>This is some text to display</TextString>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
</Text>
```

Bitmap object

If a source image is specified, it must exist. Supported graphic file formats are: JPG, JPEG, BMP, TIF, PCX, BIF, and TGA.

A bitmap without a specified source image appears as a default rectangle element on a window.

Transparent colors

The InTouch internal application object data structure for bitmaps contains a Boolean field indicating that transparent color should be applied. Another field stores the actual transparent color.

In WindowMaker, if you create a new bitmap object, it initially has a default transparent color that is black and is unavailable. After you select the transparent color tool, you have applied a permanent transparent color to the bitmap object. From the transparent color tool, there is no visual cue that the transparent color has been assigned to a bitmap.

For bitmap objects imported from an XML file, the transparent color node for a bitmap is an optional entry, but if it is present, the transparent color is enabled and the transparent color is assigned to the bitmap object. There is no mechanism for the end user to open the window in WindowMaker, to select the bitmap object, and then to disable the transparent color.

Bitmap object elements

The following elements can be specified for a bitmap object.

Elements	Description
Title	Bitmap object name. Optional.
FillColor	Interior RGB color of the bitmap object. Contains a color element. Default is rgb(0, 0, 0).
SourceImage	Path to the bitmap file. Default is Null.
Transparent	RGB color for transparent color. Default is rgb(0, 255, 0).
Pen	Defines the line around the bitmap object.
AnimationLinks	Optional animation links list.

Elements	Description
Dimension	Location and size of an object. Elements are top, left, width, height. Top, left, width, and height values are pixels. The resulting coordinates must be within -32000 and +32000. Both width and height cannot be zero. Required. An object is not created if the Dimension element is assigned invalid values or is missing values.

Example:

```
<Bitmap>
<Title>bitmap1</Title>
<Pen>
<PenColor><Name>Black</Name></PenColor>
<PenWidth>1</PenWidth>
<PenStyle>Solid</PenStyle>
</Pen>
<Dimension>
<Left>100</Left>
       <Top>50</Top>
<Width>270</Width>
       <Height>80</Height>
</Dimension>
<FillColor>
<R>128</R>
       <G>128</G> <
       B>128</B>
</FillColor>
<Transparent>
<R>0</R> <G>128</G> <B>255</B>
</Transparent>
<SourceImage>C:\MyPictures\hello.jpg</SourceImage>
<AnimationLinks>
</AnimationLinks>
</Bitmap>
```

Example:

```
<Bitmap>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
<SourceImage>C:\MyPictures\hello.jpg</SourceImage>
</Bitmap>
```

Example with empty SourceImage node:

```
<Bitmap>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
<FillColor>
```

```
<R>128</R> <G>128</G> <B>128</B>
</FillColor>
<SourceImage></SourceImage>
</Bitmap>
```

Minimal example:

```
<Bitmap>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
</Bitmap>
```

Button object

The following elements can be specified for a button object.

Elements	Description
Title	Button object name. Optional.
TextString	Displayed caption. Default string is "Text".
TextInfo	Defines how the caption is shown.
AnimationLinks	Optional animation links list. Line Color, Fill Color, Text Color, Percent Fill, Orientation animation links are not permitted.
Dimension	Location and size of an object. Elements are top, left, width, height. Top, left, width, and height values are pixels. The resulting coordinates must be within -32000 and +32000. Both width and height cannot be zero. Required. An object is not created if the Dimension element is assigned invalid values or is missing values.

Example:

```
Button>
<Title>button1</Title>
<TextInfo>
<Font>Arial</Font>
<FontStyle>Regular</FontStyle>
<FontSize>12</FontSize>
<Underline>false</Underline>
```

```
<Strikeout>false</Strikeout>
<TextColor>
<R>0</R>
<G>0</G>
<B>0</B>
</TextColor>
<TextJustify>Left</TextJustify>
</TextInfo>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
<TextString>Stop All Robots</TextString>
<AnimationLinks>
</AnimationLinks>
</Button>
```

Minimal example:

```
<Button>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
</Button>
```

SmartSymbols

A SmartSymbol template must be defined in the application before importing a window that uses the SmartSymbol template. If the SmartSymbol template does not exist, the import fails, and the window is not created.

Multiple SmartSymbols can be specified for a window. Each SmartSymbol is declared in a separate <SmartSymbol> </SmartSymbol> element.

The following elements can be specified for a Smart Symbol.

Elements	Description
SymbolName	SmartSymbol name. Required. An object is not created if the name does not match a SmartSymbol template name in the application, or the name is missing.
Dimension	Location and size of an object. Elements are top, left, width, height. Top, left, width, and height values are pixels. The resulting coordinates must be within -32000 and +32000. Both width and height cannot be zero. Required. An object is not created if the Dimension element is assigned invalid values or is missing values.

Elements	Description
TagReplace	Instance tag replacement.
StringReplace	Instance string replacement.

Example:

```
<SmartSymbol>
<SymbolName>MyCoolSymbol</SymbolName>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
<TagReplace>
<Find>Tag001</Find>
<Replace>Tag007</Replace>
</TagReplace>
<StringReplace>
<Find>Find This Text</Find>
<Replace>Replace it with this text</Replace>
</StringReplace>
</SmartSymbol>
```

Minimal example:

```
<SmartSymbol>
<SymbolName>MyCoolSymbol</SymbolName>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
</SmartSymbol>
```

Tag replacement

The tags within a SmartSymbol instance can be replaced. You should only replace tags within a SmartSymbol instance that currently exists in the application.

Tag replacement is case insensitive and the entire tag name string must match. You can add one or more <TagReplace> elements to the SmartSymbol node to replace multiple tags.

For example:

```
<TagReplace>
<Find>Tag1</Find>
<Replace>DTagA</Replace>
</TagReplace>
<TagReplace>
<Find><![CDATA[Tag1]]></Find>
<Replace><![CDATA[DTagA]]></Replace>
</TagReplace>
```

If the tag names specified for replacement do not exist in an application, the SmartSymbol is not created. Also, the window containing the SmartSymbol is not created.

The replaced tag's type must be the same as the original tag's type.

String replacement

Strings can be replaced within a SmartSymbol instance. String replacements are case sensitive and the entire source string must match. Multiple string replacement elements can be used within a single SmartSymbol element.

```
<StringReplace>
<FIND>
<![CDATA[Open]]>
</FIND>
<REPLACE>
<![CDATA[Off]]>
</REPLACE>
</StringReplace>
<StringReplace>
<FIND>Open</FIND>
<REPLACE>Off</REPLACE>
</StringReplace>
```

SmartSymbol example

This is an example of a SmartSymbol element containing tag and string replacement elements.

```
<SmartSymbol>
<SymbolName>MyCoolSymbol</SymbolName>
<Dimension>
<Left>100</Left><Top>50</Top>
<Width>270</Width><Height>80</Height>
</Dimension>
<TagReplace>
<Find>DTag1</Find>
<Replace>DTagA</Replace>
</TagReplace>
<TagReplace>
<Find>ATag001</Find>
<Replace>ATag002</Replace>
</TagReplace>
<StringReplace>
<Find><![CDATA[Open]]> </Find>
<Replace><![CDATA[On]]> </Replace>
</StringReplace>
<StringReplace>
<FIND>
<![CDATA[Closed]]>
</FIND>
<REPLACE>
<![CDATA[Off]]>
</REPLACE>
</StringReplace>
</SmartSymbol>
```

Industrial Graphics

An Industrial Graphic must be defined in the Galaxy before you import a window that uses the Industrial Graphic reference. If the Industrial Graphic reference does not exist, the import fails, and the window is not created.

You can specify symbols from an Instance or from the Graphic Toolbox in the symbol reference. However, you cannot specify Template symbols in the symbol reference.

Multiple Industrial Graphics can be specified for a window. Each Industrial Graphic is declared in a separate <Industrial Graphic> </Industrial Graphic> element.

The following elements can be specified for an Industrial Graphic.

Elements	Description
SymbolReference	Reference to look up the symbol in the Galaxy. Required. An object will not be created if the Industrial Graphic specified by the symbol reference does not exist or if the symbol reference field is missing or empty.
Dimension	The location and size of an object. The Dimension sub-elements are top, left, width, and height. Required. Specifying width and height is optional. If width and height are not specified, the original width and height of the Industrial Graphic is used. If only width is specified, then the height is calculated using the aspect ratio and vice versa. For example, if the original width was 200 and height was 100, and if you specify width as 100, the height is changed to 50. An object is not created if the Dimension element is assigned invalid values or does not contain a value. A window containing this symbol is also not created.
Title	The text name for an element. Optional. This name should be unique within the XML file; otherwise, it is ignored.
Flip	The flip type of an element. Optional.
Rotation	The rotation angle of an element. Optional.

Elements	Description
StringReplace	One or more string replacement nodes. Optional.
CustomPropertyOverride	One or more custom property overrides. Optional.
AnimationLinks	The list of animation links. Optional. Specifying Line Color, Fill Color, Text Color, Percent Fill, Orientation, Slider, Tooltip, ValueDisplay, Blink or UserInput is not permitted.

```
<ArchestrASymbol>
<Title>EmbedSym1</Title>
<Dimension>
       <Left>200</Left>
       <Top>200</Top>
       <Width>150</Width>
       <Height>150</Height>
   </Dimension>
   <Flip>None</Flip>
   <Rotation>0</Rotation>

   <SymbolReference>ButtonChromeMomentaryRed</SymbolReference>
   <AnimationLinks>
   </AnimationLinks>
   <StringReplace>
      <Find>LABEL</Find>
      <Replace>OFF</Replace>
   </StringReplace>
   </ArchestrASymbol>
```

Minimal example:

```
<ArchestrASymbol>
<Dimension>
       <Left>200</Left>
       <Top>200</Top>
   </Dimension>
   <SymbolReference>ButtonChromeMomentaryRed</SymbolReference>
</ArchestrASymbol>
```

CustomPropertyOverride

You can override only those custom properties which are already defined for the Industrial Graphic.

To override multiple custom properties, add one or more <CustomPropertyOverride> elements to the Industrial Graphic node.

Example:

```
<ArchestrASymbol>
   <Title>EmbedSym1</Title>
```

```
   <Dimension>
   <Left>200</Left>
   <Top>200</Top>
   <Width>150</Width>
   <Height>150</Height>
   </Dimension>
   <Flip>None</Flip>
   <Rotation>0</Rotation>
   <SymbolReference>ButtonChromeMomentaryRed</SymbolReference>
   <AnimationLinks>
   </AnimationLinks>
   <CustomPropertyOverride> <CustomPropertyName>cp1</CustomPropertyName>
      <OverrideValue>DTagA</OverrideValue>
      <IsConstant>false</IsConstant>
   </CustomPropertyOverride>
</ArchestrASymbol>
```

In this example, cp1 is the name of the existing custom property. The override is applied to the custom property with the new value set to DTagA. IsConstant is an optional field used to indicate whether the value should be interpreted as a constant. The IsConstant flag is only applicable if the type of the custom property is String, Time, or Elapsed Time. The IsConstant flag is set to false by default.

Note: If the tag name specified for OverrideValue does not exist in the tag database, the Industrial Graphic reference is not created on the window and import for that particular window fails. Error messages are logged in the log file or in the Logger.

Industrial Graphic string replacement type

The existing strings can be replaced within the Industrial Graphic instance. The string replacements are case sensitive. You can use multiple string replacement nodes within a single Industrial Graphic node.

```
<StringReplace>
<FIND>
<![CDATA[Open]]>
</FIND>
<REPLACE>
<![CDATA[Off]]>
</REPLACE>
</StringReplace>
<StringReplace>
<FIND>Open</FIND>
<REPLACE>Off</REPLACE>
</StringReplace>
```

Unsupported window objects

Cell, Symbol, Real-Time Trend, and Historical Trend objects cannot be imported. If elements for unsupported objects are included in the XML file, they are ignored.

Import windows using XML utility

You can add the close button script to the XML script while importing a window using the XML utility.

Note: The windows that you want to import are specified in the batch file (wm.bat).

```
<CloseButton>True</CloseButton>
```

If your XML file contains this script, the window you import will have the close window button on the title bar.

Window animation links

Window object animation links are declared within the <AnimationLinks> </AnimationLinks> element.

There can be zero or more animation links within a window element.

Not all elements support all animation link types. The Industrial Graphic support the following animation links:

- ObjSize_Height
- ObjSize_Width
- Location_Vert
- Location_Hori
- TPushB_Disc
- TPushB_Action
- TPushB_ShowWin
- TPushB_HideWin
- Misc_Visib
- Misc_Disable

For more information about Window Animation Links, see the InTouch HMI documentation.

Some animation link types prevent other animation link types from being created. Animation link processing occurs in the order specified in the XML file.

Script/expression/tag name requirements matrix

Each animation link has a control field. The control field can be a script, an expression, or a tag name. Some control fields are limited by the type of tags or expressions allowed. The following table lists the control field required for each animation link and any limitation of the control field.

Animation Link Type	Control Field	Control Field Limit
Discrete User Input	Tag	Discrete tag
Analog User Input	Tag	Analog tag
String User Input	Tag	String tag
Discrete Line Color	Expression	Discrete expression
Analog Line Color	Expression	Analog expression

Animation Link Type	Control Field	Control Field Limit
Discrete Alarm Line Color	Tag	Discrete tag alarm status
Analog Alarm Line Color	Tag	Analog tag alarm status
Discrete Fill Color	Expression	Discrete expression
Analog Fill Color	Expression	Analog expression
Discrete Alarm Fill Color	Tag	Discrete tag alarm status
Analog Alarm Fill Color	Tag	Analog tag alarm status
Discrete Text Color	Expression	Discrete expression
Analog Text Color	Expression	Analog expression
Discrete Alarm Text Color	Tag	Discrete tag alarm status
Analog Alarm Text Color	Tag	Analog tag alarm status
Vertical Slider	Tag	Valid analog tag
Horizontal Slider	Tag	Valid analog tag
Object Size Height	Expression	Analog value
Object Size Width	Expression	Analog value
Vertical Location	Expression	Valid expression
Horizontal Location	Expression	Valid expression
Vertical Percent Fill	Expression	Analog value
Horizontal Percent Fill	Expression	Analog value
Discrete Touch Pushbutton	Tag	Discrete value
Action Touch Pushbutton	Script	Valid script

Animation Link Type	Control Field	Control Field Limit
Show Window Touch Pushbutton	Window Name	Window must exist.
Hide Window Touch Pushbutton	Window Name	Window must exist.
Visibility	Expression	Discrete value
Blink	Expression	Discrete value
Orientation	Expression	Analog value
Disable	Expression	Discrete value
Tooltip	Expression	String tag
Discrete Value Display	Expression	Discrete expression
Analog Value Display	Expression	Analog expression
String Value Display	Expression	String expression

Discrete user input

The following elements can be specified for a discrete user input animation link:

Elements	Description
Title	Object name. Optional.
Message	Message text to user. Default is no message text. Required if the optional schema is used.
InputOnly	If input only: {true, false}. Default is false.
OnMessage	On message text for user. Default is On. Cannot be empty text.
OffMessage	Off message text for user. Default is Off. Cannot be empty text.
SetPrompt	Set prompt message text for user. Default is On. Cannot be empty text.

Elements	Description
ResetPrompt	Reset prompt message text for user. Default is Off. Cannot be empty text.
KeyAssignment	Virtual key element. Default is no assignment. An empty string means no assignment occurs.
Expression	Discrete tag. Required. Object is not created if invalid or missing.

Example:

```
<UserInputDiscrete>
<Title>UserInputDiscrete1</Title>
<InputOnly>false</InputOnly>
<KeyAssignment>
<KeyCode>F1</KeyCode>
<KeyFlags>Ctrl</KeyFlags>
</KeyAssignment>
<Message>Pump Valve State</Message>
<Expression><![CDATA[dTag001]]></Expression>
<OnMessage>On Message Text</OnMessage>
<OffMessage>Off Message Text</OffMessage>
<ResetPrompt>
<![CDATA[Reset Prompt Text]]>
</ResetPrompt>
<SetPrompt>Set Prompt Text</SetPrompt>
</UserInputDiscrete>
```

Minimal example:

```
<UserInputDiscrete>
<Message>Pump Valve State</Message>
<Expression>dTag001</Expression>
</UserInputDiscrete>
```

Analog user input

The following elements can be specified for an analog user input animation link.

Elements	Description
Title	Object name. Optional.
Message	Message text to user. Required if optional schema is used. Default is no message text.
InputOnly	If input only: true, false. Default is false.

Elements	Description
MinAnalogValue	Minimum floating point value allowed. Required if optional schema is used. Default is 0.0
MaxAnalogValue	Maximum floating point value allowed. Required if optional schema is used. Default is 100.0. Must be larger than the value assigned to the MINANALOGVALUE element.
KeyPadEnabled	Specifies whether a keypad is visible. Possible values are True or False. The default is False.
KeyAssignment	Virtual key element, empty string, or absent. Default is no assignment. Empty string means no assignment occurs.
Expression	Analog type tag name. Required. Object is not created if invalid or missing.

Example:

```
<<UserInputAnalog>
<Title>UserInputAnalog1</Title>
<InputOnly>false</InputOnly>
<KeyAssignment>
<KeyCode>F1</KeyCode>
<KeyFlags>Ctrl</KeyFlags>
</KeyAssignment>
<Message>Flush Pump Speed</Message>
<Expression>aTag001</Expression>
<KeyPadEnabled>false</KeyPadEnabled>
<MinAnalogValue>0.0</MinAnalogValue>
<MaxAnalogValue> 100.0</MaxAnalogValue>
</UserInputAnalog>
```

Minimal example:

```
<UserInputAnalog>
<Message>Pump Valve State</Message>
<Expression><![CDATA[aTag001]]></Expression>
<MinAnalogValue>0.0</MinAnalogValue>
<MaxAnalogValue>100.0</MaxAnalogValue>
</UserInputAnalog>
```

String user input

You should use either EchoEnabled or EchoMode, but not both. EchoMode allows you to specify the password mode, whereas the EchoEnabled allows only the states enabled or disabled.

You use the Echo Character element to control how user input is shown during run time. Possible values are Yes,

No, and Password.

- If the element is set to Yes, then during run time the string characters are shown in the input edit box. Input can only be enabled. Password character and encryption are disabled.
- If the element is set to No, then during run time the input characters are not shown in the input edit box. Input can only be enabled. Password character and encryption are disabled.
- If the element is set to Password, then during run time the password character is shown instead of the password typed by the user. A password character is optional, but if specified, cannot be empty. The default password character is an asterisk. Encryption is optional and is off by default. Input only is mandatory and is forced on.

If the elements do not match these criteria, then default options are used.

You can specify the following elements for a string user input animation link.

Elements	Description
Title	Object name. Optional.
Message	Message text to user. Required if optional schema is used. Default is no message text.
InputOnly	If input only: {true, false}. Default is false.
EchoCharacters	Echo characters: no, yes, password. Required if optional schema is used. Default is yes.
KeyPadEnabled	If keypad is visible: true, false. Default is false.
PasswordCharacter	Password character. Default is "*"
EncryptEnabled	Encryption enabled: yes, no. Default is no.
KeyAssignment	Virtual key element, empty string, or absent. Default is no assignment. Empty string means no assignment occurs.
Expression	Message-type tag. Required. Object is not created if invalid or missing.

Example:

```
<UserInputString>
<Title>UserInputString1</Title>
<InputOnly>false</InputOnly>
<KeyAssignment>
<KeyCode>F1</KeyCode>
<KeyFlags>Ctrl</KeyFlags>
</KeyAssignment>
```

```
<Message>Select Pump</Message>
<Expression>mTag001</Expression>
<EchoCharacters>true</EchoCharacters>
<EncryptEnabled>false</EncryptEnabled>
<PasswordCharacter>*</PasswordCharacter>
</UserInputString>
```

Minimal example:

```
<UserInputString>
<Message>Select Pump</Message>
<Expression>mTag001</Expression>
<EchoCharacters>true</EchoCharacters>
</UserInputString>
```

Discrete line color

The following elements can be specified for a discrete line color animation link.

Elements	Description
Title	Object name. Optional.
Oncolor	Contains a color element. Default is rgb(0,0,0).
Offcolor	Contains a color element. Default is rgb(0,0,0).
Expression	Discrete tag or expression. Required. Object is not created if the expression element values are missing or invalid.

Example:

```
<LineColorDiscrete>
<Title>LineColorDiscrete1</Title>
<Expression>
<![CDATA[dTag001]]>
</Expression>
<OnColor><Name>Green</Name></OnColor>
<OffColor><Name>Red</Name></OffColor>
</LineColorDiscrete>
```

Minimal example:

```
<LineColorDiscrete>
<Expression>dTag001</Expression>
</LineColorDiscrete>
```

Analog line color

The required breakpoint values must have increasing values. If this is not the case, a warning is logged, and the offending value is one plus the previous value.

The object containing the animation link is not created if a required value is missing or invalid.

You can specify the following elements for an analog line color animation link.

Elements	Description
Title	Object name. Optional.
Color1	Contains a color element. Default is rgb(0,0,0).
Color2	Contains a color element. Default is rgb(0,0,0).
Color3	Contains a color element. Default is rgb(0,0,0).
Color4	Contains a color element. Default is rgb(0,0,0).
Color5	Contains a color element. Default is rgb(0,0,0).
Color6	Contains a color element. Default is rgb(0,0,0).
Color7	Contains a color element. Default is rgb(0,0,0).
Color8	Contains a color element. Default is rgb(0,0,0).
Color9	Contains a color element. Default is rgb(0,0,0).
Color10	Contains a color element. Default is rgb(0,0,0).
Values	Constant analog values. Must contain nine Value elements. Required.
Expression	Analog tag or expression. Required.

Example:

```
<LineColorAnalog>
<Title>LineColorAnalog1</Title>
<Expression>aTag001</Expression>
<Colors>
<Color1><Name>White</Name></Color1>
<Color2><Name>Red</Name></Color2>
<Color3><Name>Orange</Name></Color3>
<Color4><Name>Yellow</Name></Color4>
<Color5><Name>Green</Name></Color5>
<Color6><Name>Blue</Name></Color6>
<Color7><Name>Cyan</Name></Color7>
<Color8><Name>Magenta</Name></Color8>
<Color9><Name>Violet</Name></Color9>
<Color10><Name>Black</Name></Color10>
</Colors>
<Values>
```

```
<Value>10.0</Value>
<Value>20.0</Value>
<Value>30.0</Value>
<Value>40.0</Value>
<Value>50.0</Value>
<Value>60.0</Value>
<Value>70.0</Value>
<Value>80.0</Value>
<Value>90.0</Value>
</Values>
</LineColorAnalog>
```

Minimal example:

```
<LineColorAnalog>
<Expression>aTag001</Expression>
<Values>
<Value>10.0</Value>
<Value>20.0</Value>
<Value>30.0</Value>
<Value>40.0</Value>
<Value>50.0</Value>
<Value>60.0</Value>
<Value>70.0</Value>
<Value>80.0</Value>
<Value>90.0</Value>
</Values>
</LineColorAnalog>
```

Discrete alarm line color

The following elements can be specified for a discrete alarm line color animation link.

Elements	Description
Title	Object name. Optional.
NormalColor	Contains a color element. Default is rgb(0,0,0.)
AlarmColor	Contains a color element. Default is rgb(0,0,0.)
Expression	Discrete tag. Required. Object is not created if expression is invalid or missing.

Example:

```
<LineColorDiscreteAlarm>
<Title>LineColorDiscreteAlarm1</Title>
<Expression>
<![CDATA[dTag001]]>
</Expression>
<NormalColor><Name>Black</Name></NormalColor>
<AlarmColor><Name>Red</Name></AlarmColor>
</LineColorDiscreteAlarm>
```

Minimal example:

```
<LineColorDiscreteAlarm>
<Expression>dTag001</Expression>
</LineColorDiscreteAlarm>
```

Analog alarm line color

The following elements can be specified for an analog alarm line color animation link.

Elements	Description
Title	Object name. Optional.
Value: LOLOCOLOR	Contains a color element. Default is rgb(0,0,0).
Value:LOCOLOR	Contains a color element. Default is rgb(0,0,0).
Value: NORMALCOLOR	Contains a color element. Default is rgb(0,0,0).
Value:HICOLOR	Contains a color element. Default is rgb(0,0,0).
Value: HIHICOLOR	Contains a color element. Default is rgb(0,0,0).
Deviation: NORMALCOLOR	Contains a color element. Default is rgb(0,0,0).
Deviation: MINORCOLOR	Contains a color element. Default is rgb(0,0,0).
Deviation: MAJORCOLOR	Contains a color element. Default is rgb(0,0,0).
ROC: NORMALCOLOR	Contains a color element. Default is rgb(0,0,0).
ROC: ROCCOLOR	Contains a color element. Default is rgb(0,0,0).
Expression	Analog tag. Required. Object is not created if expression is invalid or missing.

Example of a value alarm:

```
<LineColorAnalogAlarm>
<Title>LineColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<Value>
<LoLoColor><Name>Red</Name><LoLoColor>
<LoColor><Name>DarkRed</Name><LoColor>
```

```
<NormalColor><Name>Black</Name></NormalColor>
<HiColor><Name>DarkGreen</Name></HiColor>
<HiHiColor><Name>Green</Name></HiHiColor>
</Value>
</LineColorAnalogAlarm>
```

Example of a deviation alarm:

```
<LineColorAnalogAlarm>
<Title>LineColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
   <Deviation>
<NormalColor><Name>Black</Name></NormalColor>
<MinorColor><Name>Green</Name></MinorColor>
<MajorColor><Name>Red</Name></MajorColor>
</Deviation>
</LineColorAnalogAlarm>
```

Example of a ROC alarm:

```
<LineColorAnalogAlarm>
<Title>LineColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<ROC>
<NormalColor><Name>Black</Name></NormalColor>
<ROCColor><Name>Red</Name></ROCColor>
</ROC>
</LineColorAnalogAlarm>
```

Discrete fill color

The following elements can be specified for a discrete fill color animation link.

Elements	Description
Title	Object name. Optional.
OnColor	Contains a color element. Default is rgb(0,0,0).
OffColor	Contains a color element. Default is rgb(0,0,0).
Expression	Discrete tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<FillColorDiscrete>
<Title>FillColorDiscrete1</Title>
<Expression>
<![CDATA[dTag001]]>
</Expression>
<OnColor><Name>Green</Name></OnColor>
```

```
<OffColor><Name>Red</Name></OffColor>
</FillColorDiscrete>
```

Minimal example:

```
<FillColorDiscrete>
<Expression>dTag001</Expression>
</FillColorDiscrete>
```

Analog fill color

The object containing the animation link is not created if a required value is missing or invalid.

You can specify the following elements for an analog fill color animation link.

Elements	Description
Title	Object name. Optional.
Color1	Contains a color element. Default is rgb(0,0,0).
Color2	Contains a color element. Default is rgb(0,0,0).
Color3	Contains a color element. Default is rgb(0,0,0).
Color4	Contains a color element. Default is rgb(0,0,0).
Color5	Contains a color element. Default is rgb(0,0,0).
Color6	Contains a color element. Default is rgb(0,0,0).
Color7	Contains a color element. Default is rgb(0,0,0).
Color8	Contains a color element. Default is rgb(0,0,0).
Color9	Contains a color element. Default is rgb(0,0,0).
Color10	Contains a color element. Default is rgb(0,0,0).
Values	Constant analog values. Must contain nine value elements. Required.
Expression	Analog tag or expression. Required.

Example:

```
<FillColorAnalog>
<Title>FillColorAnalog1</Title>
<Expression>aTag001</Expression>
```

```
<Colors>
<Color1><Name>White</Name></Color1>
<Color2><Name>Red</Name></Color2>
<Color3><Name>Orange</Name></Color3>
<Color4><Name>Yellow</Name></Color4>
<Color5><Name>Green</Name></Color5>
<Color6><Name>Blue</Name></Color6>
<Color7><Name>Cyan</Name></Color7>
<Color8><Name>Magenta</Name></Color8>
<Color9><Name>Violet</Name></Color9>
<Color10><Name>Black</Name></Color10>
</Colors>
<Values>
<Value>10.0</Value>
<Value>20.0</Value>
<Value>30.0</Value>
<Value>40.0</Value>
<Value>50.0</Value>
<Value>60.0</Value>
<Value>70.0</Value>
<Value>80.0</Value>
<Value>90.0</Value>
</Values>
</FillColorAnalog>
```

Minimal example:

```
<FillColorAnalog>
<Expression>aTag001</Expression>
<Values>
<Value>10.0</Value>
<Value>20.0</Value>
<Value>30.0</Value>
<Value>40.0</Value>
<Value>50.0</Value>
<Value>60.0</Value>
<Value>70.0</Value>
<Value>80.0</Value>
<Value>90.0</Value>
</Values>
</FillColorAnalog>
```

Discrete alarm fill color

The following elements can be specified for a discrete alarm fill color animation link.

Elements	Description
Title	Object name. Optional.
NormalColor	Contains a color element. Default is rgb(0,0,0).
AlarmColor	Contains a color element. Default is rgb(0,0,0).

Elements	Description
Expression	Discrete tag. Required. Object is not created if the expression is invalid or missing.

Example:

```
<FillColorDiscreteAlarm>
<Title>FillColorDiscreteAlarm1</Title>
<Expression>
<![CDATA[dTag001]]>
</Expression>
<NormalColor><Name>Black</Name></NormalColor>
<AlarmColor><Name>Red</Name></AlarmColor>
</FillColorDiscreteAlarm>
```

Minimal example:

```
<FillColorDiscreteAlarm>
<Expression>dTag001</Expression>
</FillColorDiscreteAlarm>
```

Analog alarm fill color

The following elements can be specified for an analog alarm fill color animation link. The object containing the animation link is not created if a required value is missing or invalid.

Elements	Description
Title	Object name. Optional.
Value: LoLoColor	Contains a color element. Default is rgb(0,0,0).
Value: LoColor	Contains a color element. Default is rgb(0,0,0).
Value: NormalColor	Contains a color element. Default is rgb(0,0,0).
Value: HiColor	Contains a color element. Default is rgb(0,0,0).
Value: HiHiColor	Contains a color element. Default is rgb(0,0,0).
Deviation: NormalColor	Contains a color element. Default is rgb(0,0,0).
Deviation: MinorColor	Contains a color element. Default is rgb(0,0,0).
Deviation: MajorColor	Contains a color element. Default is rgb(0,0,0).

Elements	Description
ROC: NormalColor	Contains a color element. Default is rgb(0,0,0).
ROC: ROCColor	Contains a color element. Default is rgb(0,0,0).
Expression	Analog tag. Required. Object is not created if the expression is invalid or missing.

Example of a value alarm:

```
<FillColorAnalogAlarm>
<Title>FillColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<Value>
<LoLoColor><Name>Red</Name><LoLoColor>
<LoColor><Name>DarkRed</Name><LoColor>
<NormalColor><Name>Black</Name></NormalColor>
<HiColor><Name>DarkGreen</Name></HiColor>
<HiHiColor><Name>Green</Name></HiHiColor>
</Value>
</FillColorAnalogAlarm>
```

Example of a Deviation Alarm:

```
<FillColorAnalogAlarm>
<Title>FillColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<Deviation>
<NormalColor><Name>Black</Name></NormalColor>
<MinorColor><Name>Green</Name></MinorColor>
<MajorColor><Name>Red</Name></MajorColor>
</Deviation>
</FillColorAnalogAlarm>
```

Example of a ROC Alarm:

```
<FillColorAnalogAlarm>
<Title>FillColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<ROC>
<NormalColor><Name>Black</Name></NormalColor>
<ROCColor><Name>Red</Name></ROCColor>
</ROC>
</FillColorAnalogAlarm>
```

Discrete text color

The following elements can be specified for a discrete text color animation link.

Elements	Description
Title	Object name. Optional.
OnColor	Contains a color element. Default is rgb(0,0,0).
OffColor	Contains a color element. Default is rgb(0,0,0).
Expression	String tag expression. Required.

Example:

```
<TextColorDiscrete>
<Title>TextColorDiscrete1</Title>
<Expression>
<![CDATA[dTag001]]>
</Expression>
<OnColor><Name>Green</Name></OnColor>
<OffColor><Name>Red</Name></OffColor>
</TextColorDiscrete>
```

Minimal example:

```
<TextColorDiscrete>
<Expression>dTag001</Expression>
</TextColorDiscrete>
```

Analog text color

The following elements can be specified for an analog text color animation link. The object containing the animation link is not created if a required value is missing or invalid.

Elements	Description
Title	Object name. Optional.
Color1	Contains a color element. Default is rgb(0,0,0).
Color2	Contains a color element. Default is rgb(0,0,0).
Color3	Contains a color element. Default is rgb(0,0,0).
Color4	Contains a color element. Default is rgb(0,0,0).
Color5	Contains a color element. Default is rgb(0,0,0).
Color6	Contains a color element. Default is rgb(0,0,0).

Elements	Description
Color7	Contains a color element. Default is rgb(0,0,0).
Color8	Contains a color element. Default is rgb(0,0,0).
Color9	Contains a color element. Default is rgb(0,0,0).
Color10	Contains a color element. Default is rgb(0,0,0).
Values	Constant analog values. Must contain nine value elements. Required.
Expression	Analog tag expression. Required.

Example:

```
<TextColorAnalog>
<Title>TextColorAnalog1</Title>
<Expression>aTag001</Expression>
<Colors>
<Color1><Name>White</Name></Color1>
<Color2><Name>Red</Name></Color2>
<Color3><Name>Orange</Name></Color3>
<Color4><Name>Yellow</Name></Color4>
<Color5><Name>Green</Name></Color5>
<Color6><Name>Blue</Name></Color6>
<Color7><Name>Cyan</Name></Color7>
<Color8><Name>Magenta</Name></Color8>
<Color9><Name>Violet</Name></Color9>
<Color10><Name>Black</Name></Color10>
</Colors>
<Values>
<Value>10.0</Value> <Value>20.0</Value>
<Value>30.0</Value> <Value>40.0</Value>
<Value>50.0</Value> <Value>60.0</Value>
<Value>70.0</Value> <Value>80.0</Value>
<Value>90.0</Value>
</Values>
</TextColorAnalog>
```

Minimal example:

```
<TextColorAnalog>
<Expression>aTag001</Expression>
<Values>
<Value>10.0</Value>
<Value>20.0</Value>
<Value>30.0</Value>
<Value>40.0</Value>
<Value>50.0</Value>
<Value>60.0</Value>
<Value>70.0</Value>
```

```
<Value>80.0</Value>
<Value>90.0</Value>
</Values>
</TextColorAnalog>
```

Discrete alarm text color

The following elements can be specified for a discrete alarm text color animation link.

Elements	Description
Title	Object name. Optional.
NormalColor	Contains a color element. Default is rgb(0,0,0).
AlarmColor	Contains a color element. Default is rgb(0,0,0).
Expression	Discrete tag. Required. Object is not created if the expression is invalid or missing.

Example:

```
<TextColorDiscreteAlarm>
<Title>TextColorDiscreteAlarm1</Title>
<Expression>
<![CDATA[dTag001]]>
</Expression>
<NormalColor><Name>Black</Name></NormalColor>
<AlarmColor><Name>Red</Name></AlarmColor>
</TextColorDiscreteAlarm>
```

Minimal example:

```
<TextColorDiscreteAlarm>
<Expression>dTag001</Expression>
</TextColorDiscreteAlarm>
```

Analog alarm text color

The following elements can be specified for an analog alarm text color animation link.

Elements	Description
Title	Object name. Optional.
Value: LoLoColor	Contains a color element. Default is rgb(0,0,0).
Value: LoColor	Contains a color element. Default is rgb(0,0,0).
Value: NormalColor	Contains a color element. Default is rgb(0,0,0).

Elements	Description
Value: HiColor	Contains a color element. Default is rgb(0,0,0).
Value: HiHiColor	Contains a color element. Default is rgb(0,0,0).
Deviation: NormalColor	Contains a color element. Default is rgb(0,0,0).
Deviation: MinorColor	Contains a color element. Default is rgb(0,0,0).
Deviation: MajorColor	Contains a color element. Default is rgb(0,0,0).
ROC: NormalColor	Contains a color element. Default is rgb(0,0,0).
ROC: ROCColor	Contains a color element. Default is rgb(0,0,0).
Expression	Analog tag. Required. Object is not created if the expression is invalid or missing.

Example:

```
<TextColorAnalogAlarm>
<Title>TextColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<Value>
<LoLoColor><Name>Red</Name><LoLoColor>
<LoColor><Name>DarkRed</Name><LoColor>
<NormalColor><Name>Black</Name></NormalColor>
<HiColor><Name>DarkGreen</Name></HiColor>
<HiHiColor><Name>Green</Name></HiHiColor>
</Value>
</TextColorAnalogAlarm>
```

Example of a Deviation Type Alarm:

```
<TextColorAnalogAlarm>
<Title>TextColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<Deviation>
<NormalColor><Name>Black</Name></NormalColor>
<MinorColor><Name>Green</Name></MinorColor>
<MajorColor><Name>Red</Name></MajorColor>
</Deviation>
</TextColorAnalogAlarm>
```

Example of a ROC Type Alarm:

```
<TextColorAnalogAlarm>
```

```
<Title>TextColorAnalogAlarm1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<ROC>
<NormalColor><Name>Black</Name></NormalColor>
<ROCColor><Name>Red</Name></ROCColor>
</ROC>
</TextColorAnalogAlarm>
```

Vertical slider

You can specify the following elements for a vertical slider animation link.

Elements	Description
Title	Object name. Optional.
ReferenceLocation	Vertical reference: top, middle, bottom. Default is bottom.
TopValue	Top value. Cannot be the same as BottomValue. Default is 0.
BottomValue	Bottom value. Default is 100.
UpwardMovement	Upward movement. Must range from 0 to 32767. Out of range values are clamped and an error message is logged. Default is 50.
DownwardMovement	Downward movement. Must range from 0 to 32767. Out of range values are clamped and an error message is logged. Default is 50.
Expression	Analog tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<SliderVertical>
<Title>SliderVertical1</Title>
<ReferenceLocation>Bottom</ReferenceLocation>
<TopValue>10.0</TopValue>
<BottomValue> 110.0</BottomValue>
<UpwardMovement> 20.0</UpwardMovement>
<DownwardMovement> 120.0</DownwardMovement>
<Expression> <![CDATA[aTag001]]> </Expression>
</SliderVertical>
```

Minimal example:

```
<SliderVertical>
<Expression>aTag001</Expression>
```

```
</SliderVertical>
```

Horizontal slider

The following elements can be specified for a horizontal slider animation link.

Elements	Description
Title	Object name. Optional.
ReferenceLocation	Reference location: left, center, right. Default is left.
LeftValue	Left value of a slider. Default is 0.
RightValue	Right value of a slider. Default is 100.
LeftMovement	Left horizontal movement. Must range from 0 to 32767. Out of range values are clamped and an error message is logged. Default is 50.
RightMovement	Right horizontal movement. Must range from 0 to 32767. Out of range values are clamped and an error message is logged. Default is 50.
Expression	Analog tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<SliderHorizontal>
<Title>SliderHorizontal1</Title>
<ReferenceLocation>Left</ReferenceLocation>
<LeftValue>10.0</LeftValue>
<RightValue>120.0</RightValue>
<LeftMovement>20.0</LeftMovement>
<RightMovement>150.0</RightMovement>
<Expression>
<![CDATA[aTag001]]>
</Expression>
</SliderHorizontal>
```

Minimal example:

```
<SliderHorizontal>
<Expression><![CDATA[aTag001]]></Expression>
</SliderHorizontal>
```

Object height

The following elements can be specified for an object height animation link. An object height animation link cannot be used with an orientation animation link.

Elements	Description
Title	Object name. Optional.
SizeAnchor	Defines where on the object the anchor is located. Values can be bottom, middle, top. Default is bottom.
SizeMin	Value at minimum height. Default is 0.
SizeMax	Value at maximum height. Default is 100. Must be greater than the value assigned to the SizeMin element.
MinPercent	Minimum percentage height. Default is 0. Range is from 0 to 100.
MaxPercent	Maximum percentage height. Default is 100. Must be more than minimum percent. Range is from 0 to 100.
Expression	Analog tag name or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<ObjectSizeHeight>
<Title>ObjectSizeHeight1</Title>
<Expression> <![CDATA[aTag001]]> </Expression>
<SizeMin>0.0</SizeMin>
<SizeMax>100.0</SizeMax>
<MinPercent>0.0</MinPercent>
<MaxPercent>100.0</MaxPercent>
<SizeAnchor>Top</SizeAnchor>
</ObjectSizeHeight>
```

Minimal example:

```
<ObjectSizeHeight>
<Expression>aTag001</Expression>
</ObjectSizeHeight>
```

Object width

The following elements can be specified for an object size width animation link. An object width animation link cannot be used with an orientation animation link.

Elements	Description
Title	Object name. Optional.
SizeAnchor	Defines where on the object the anchor is located. Values can be: {Left, center, right}. Default is left.

Elements	Description
SizeMin	Value at minimum width. Default is 0.
SizeMax	Value at maximum width. Default is 100. Must be larger than SizeMin.
MinPercent	Minimum percentage width. Default is 0. Range is from 0 to 100.
MaxPercent	Maximum percentage width. Default is 100. Must be more than MinPercent. Range is from 0 to 100.
Expression	Analog tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
< ObjectSizeWidth>
<Title>ObjectSizeWidth1</Title>
<Expression> <![CDATA[aTag001]]> </Expression>
<SizeMin>0.0</SizeMin>
<SizeMax>100.0</SizeMax>
<MinPercent>0.0</MinPercent>
<MaxPercent>100.0</MaxPercent>
<SizeAnchor>Center</SizeAnchor>
</ObjectSizeWidth>
```

Minimal example:

```
<ObjectSizeWidth>
<Expression>aTag001</Expression>
</ObjectSizeWidth>
```

Vertical location

The following elements can be specified for a vertical location animation link. An object vertical location animation link cannot be used with an orientation animation link.

Elements	Description
Title	Object name. Optional.
MinValue	Value at top. Default is 0. Cannot be the same value as at bottom.
MaxValue	Value at bottom. Default is 100. Cannot be same value as at top.

Elements	Description
DecreaseMovement	Vertical movement upward. Default is 0. Cannot be less than 0.
IncreaseMovement	Vertical movement downward. Default is 100. Cannot be greater than 100.
Expression	Analog tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<LocationVertical>
<Title>LocationVertical1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<MinValue>0.0</MinValue>
<MaxValue>100.0</MaxValue>
<DecreaseMovement>0</DecreaseMovement>
<IncreaseMovement>100</IncreaseMovement>
</LocationVertical>
```

Minimal example:

```
<LocationVertical>
<Expression>aTag001</Expression>
</LocationVertical>
```

Horizontal location

The following elements can be specified for a horizontal location animation link. An object horizontal location animation link cannot be used with an orientation animation link.

Elements	Description
Title	Object name. Optional.
MinValue	Value at left. Default is 0. Cannot be same value as at right end.
MaxValue	Value at right. Default is 100. Cannot be same value as at left end.
DecreaseMovement	Horizontal movement to left. Default is 0. Must not be less than 0.

Elements	Description
IncreaseMovement	Horizontal movement to right. Default is 100. Must not be greater than 100.
Expression	Analog tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<LocationHorizontal>
<Title>LocationHorizontal1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
<MinValue>0.0</MinValue>
<MaxValue>100.0</MaxValue>
<DecreaseMovement>0</DecreaseMovement>
<IncreaseMovement>100</IncreaseMovement>
</LocationHorizontal>
```

Minimal example:

```
<LocationHorizontal>
<Expression>aTag001</Expression>
</LocationHorizontal>
```

Vertical percent fill

The following elements can be specified for a vertical percent fill animation link. A vertical percent fill animation link cannot be used with an orientation animation link.

Elements	Description
Title	Object name. Optional.
FillDirection	Defines the direction of motion. Possible values are: {Up, down}. Default is up. Required.
FillColor	Solid fill color element. Default is rgb(0, 0, 0).
FillMin	Defines minimum value. Default is 0.
FillMax	Defines maximum value. Default is 100. Must be more than the minimum fill.
FillMinPercent	Defines minimum value as a percentage. Default is 0. Range from 0 to 100.

Elements	Description
FillMaxPercent	Defines maximum value as a percentage. Default is 100. Range from 0 to 100. Must be more than the minimum percentage.
Expression	Analog tag or expression. Required.

Example:

```
<PercentFillVertical>
<Title>PercentFillVertical1</Title>
<Expression><![CDATA[aTag001]]></Expression>
<FillMin>0.0</FillMin>
<FillMax>100.0</FillMax>
<FillMinPercent>0</FillMinPercent>
<FillMaxPercent>100</FillMaxPercent>
<FillColor><Name>Purple</Name></FillColor>
<FillDirection>Up</FillDirection>
</PercentFillVertical>
```

Minimal example:

```
<PercentFillVertical>
<Expression>aTag001</Expression>
<FillDirection>Up</FillDirection>
</PercentFillVertical>
```

Horizontal percent fill

The following elements can be specified for a horizontal percent fill animation link. A horizontal percent fill animation link cannot be used with an orientation animation link.

Elements	Description
Title	Object name. Optional.
FillDirection	Left, right. Default is right.
FillColor	Contains a color element. Default is rgb(0, 0, 0).
FillMin	Value at minimum fill. Default is 0.
FillMax	Value at maximum fill. Default is 100. Must be more than the minimum fill.
FillMinPercent	Minimum percent to fill object. Default is 0. Range from 0 to 100.
FillMaxPercent	Maximum percent to fill object. Default is 100.Range from 0 to 100. Must be more than the minimum percent fill.

Elements	Description
Expression	Analog tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<PercentFillHorizontal>
<Title>PercentFillHorizontal1</Title>
<Expression><![CDATA[aTag001]]></Expression>
<FillMin>0.0</FillMin>
<FillMax>100.0</FillMax>
<FillMinPercent>0</FillMinPercent>
<FillMaxPercent>100</FillMaxPercent>
<FillColor><Name>Purple</Name></FillColor>
<FillDirection>Right</FillDirection>
</PercentFillHorizontal>
```

Minimal example:

```
<PercentFillHorizontal>
<Expression>aTag001</Expression>
<FillDirection>Left</FillDirection>
</PercentFillHorizontal>
```

Discrete pushbutton

The following elements can be specified for a discrete pushbutton animation link.

Elements	Description
Title	Object name. Optional.
ButtonType	Type of discrete button: {direct, reverse, toggle, reset, or set}. Required. Object is not created if invalid or missing.
Expression	Discrete type tag or expression. Required. Object is not created if the expression element is invalid or missing.
KeyAssignment	Virtual key element, empty string, or absent. Default is no assignment. An empty string means no assignment occurs.

Example:

```
<ButtonDiscreteValue>
<Title>ButtonDiscreteValue1</Title>
<ButtonType>Reverse</ButtonType>
<KeyAssignment>
<KeyCode>F2</KeyCode>
<KeyFlags>Shift</KeyFlags>
```

```
</KeyAssignment>
<Expression>
<![CDATA[dTag001]]>
</Expression>
</ButtonDiscreteValue>
```

Minimal example:

```
<ButtonDiscreteValue>
<ButtonType>Reverse</ButtonType>
<Expression><![CDATA[dTag001]]></Expression>
</ButtonDiscreteValue>
```

Show window pushbutton

If a named window does not exist at the time the link is generated, a warning is logged. The link is generated without any action for the window. At least one of the named windows must exist, or the ShowWindow animation link is not imported.

You can specify the following elements for a show window pushbutton animation link.

Element	Description
Title	Object name. Optional.
WindowName	Name of window to show. There must be at least one window name specified or this animation link is not added to the object.

Example:

```
<ButtonShowWindow>
<Title>ButtonShowWindow1</Title>
<WindowName>
<![CDATA[Window006]]>
</WindowName>
<WindowName>Window007</WindowName>
<WindowName><![CDATA[SliderWindow]]></WindowName>
</ButtonShowWindow>
```

Hide window pushbutton

If a named window does not exist at the time the link is generated, a warning is logged. The link is generated without any action for the window. At least one of the named windows must exist, or the HideWindow animation link is not imported.

The following elements can be specified for a hide window pushbutton animation link.

Element	Description
Title	Object name. Optional.
WindowName	Name of window to hide. There must be at least one window name specified or this animation link is not added to the object.

Example:

```
<ButtonHideWindow>
<Title>ButtonHideWindow1</Title>
<WindowName>
<![CDATA[Window001]]>
</WindowName>
<WindowName>Window002</WindowName>
<WindowName> <![CDATA[SliderWindow]]> </WindowName>
</ButtonHideWindow>
```

Visibility

The following elements can be specified for a visibility animation link.

Elements	Description
Title	Object name. Optional.
State	Defines if the object can be seen. Possible values are: {On, off}. Default is On. Required if optional schema is used.
Expression	Discrete tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<Visibility>
<Title>Visibility1</Title>
<Expression>dTag001</Expression>
<State>On</State>
</Visibility>
```

Minimal example:

```
<Visibility>
<Expression>dTag001</Expression>
<State>On</State>
</Visibility>
```

Blink

The following elements can be specified for a blink animation link. The animation link blinks when the expression is true.

Elements	Description
Title	Object name. Optional.
BlinkAttribute	{Invisible, Visible}. Default is Visible.
BlinkSpeed	Defines the rate the object blinks. Possible values are: {Slow, Medium, Fast}. Default is Medium.
TextColor	Defines the color of the text that blinks. Contains a color element. Default is rgb(0,0,0).
LineColor	Contains a color element. Default is rgb(0,0,0).
FillColor	Contains a color element. Default is rgb(0,0,0).
Expression	Discrete tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<Blink>
<Title>Blink1</Title>
<Expression>dTag001</Expression>
<TextColor><R>0</R><G>0</G><B>0</B></TextColor>
<LineColor><R>0</R><G>0</G><B>0</B></LineColor>
<FillColor><R>0</R><G>255</G><B>0</B></FillColor>
<BlinkAttribute>Invisible</BlinkAttribute>
<BlinkSpeed>Slow</BlinkSpeed>
</Blink>
```

Minimal example:

```
<Blink>
<Expression>dTag001</Expression>
<BlinkAttribute>Visible</BlinkAttribute>
<BlinkSpeed>Fast</BlinkSpeed>
</Blink>
```

Orientation

The following elements can be specified for an orientation animation link. An orientation animation link cannot be used with slider, size, location, or percent fill animation links.

Elements	Description
Title	Object name. Optional.
X	Horizontal offset from object center point. Optional. Default is 0.

Elements	Description
Y	Vertical offset from object center point. Optional. Default is 0.
CWMax	Value at maximum clockwise rotation. Default is 100.
CWRotation	Clockwise rotation. Default is 360. Must range from 0 to 360. CWROTATION+CCWROTATION cannot exceed 360.
CCWMax	Value at maximum counter clockwise rotation. Default is 0.
CCWRotation	Counter clockwise rotation. Default is 0. Must range from 0 to 360. CWRotation+CCWRotation cannot exceed 360.
Expression	Analog tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<Orientation>
<Title>Orientation1</Title>
<Expression> <![CDATA[aTag001]]> </Expression>
<X>0</X> <Y>0</Y>
<CWMax>100.0</CWMax>
<CWRotation>360.0</CWRotation>
<CCWMax>0.0</CCWMax>
<CCWRotation>0.0</CCWRotation>
</Orientation>
```

Minimal example:

```
<Orientation>
<Expression>aTag001</Expression>
</Orientation>
```

Disable

The following elements can be specified for a disable animation link.

Elements	Description
Title	Object name. Optional.
State	{On, off}. Default is On. Required if the optional schema is used.

Elements	Description
Expression	Discrete tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<Disable>
<Title>Disable1</Title>
<Expression>dTag001</Expression>
<State>On</State>
</Disable>
```

Minimal example:

```
<Disable>
<Expression>dTag001</Expression>
<State>On</State>
</Disable>
```

Static tooltip

The following elements can be specified for a static tooltip animation link.

Elements	Description
Title	Object name. Optional.
Message	Static text message. Required. Object is not created if invalid or missing.

Example:

```
<TooltipStatic>
<Title>TooltipStatic1</Title>
<Message>
<![CDATA[ Click here to win a million dollars!
]]>
</Message>
</TooltipStatic>
```

Dynamic tooltip

The following elements can be specified for a dynamic tooltip animation link.

Elements	Description
Title	Object name. Optional.
Expression	Expression element. Required. Object is not created if the expression is invalid or missing.

Example:

```
<TooltipTag>
<Title>TooltipTag1</Title>
<Expression>
<![CDATA[sTag001]]>
</Expression>
</TooltipTag>
```

Discrete value display

The following elements can be specified for a discrete value animation link.

Elements	Description
Title	Object name. Optional.
OnMessage	The string to display when the value of the animation link is True. Default text is On.
OffMessage	The string to display .when the value of the animation link is False. Default text is Off.
Expression	Discrete tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<ValueDisplayDiscrete>
<Title>ValueDisplayDiscrete1</Title>
<Expression>
<![CDATA[dTag001]]>
</Expression>
<OnMessage>
<![CDATA[Pump is On]]>
</OnMessage>
<OffMessage>
<![CDATA[Pump is Off]]>
</OffMessage>
</ValueDisplayDiscrete>
```

Minimal example:

```
<ValueDisplayDiscrete>
<Expression>dTag001</Expression>
</ValueDisplayDiscrete>
```

Analog value display

The following elements can be specified for an analog value animation link.

Elements	Description
Title	Object name. Optional.
Expression	Analog tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<ValueDisplayAnalog>
<Title>ValueDisplayAnalog1</Title>
<Expression>
<![CDATA[aTag001]]>
</Expression>
</ValueDisplayAnalog>
```

String value display

The following elements can be specified for a string value animation link.

Elements	Description
Title	Object name. Optional.
Expression	Message tag or expression. Required. Object is not created if the expression is invalid or missing.

Example:

```
<ValueDisplayString>
<Title>ValueDisplayString1</Title>
<Expression>
<![CDATA[sTag001]]>
</Expression>
</ValueDisplayString>
```

Pushbutton action scripts

Pushbutton action script elements are declared within a button action script element, which is 0 declared within an animation links element.

Example:

```
<AnimationLinks>
<ButtonActionScripts>
<OnLeftDown>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnLeftDown>
</ButtonActionScripts>
</AnimationLinks>
```

On Left Key Down/On Key Down

Do not use both OnLeftDown and OnKeyDown animation links in the same object.

Example:

```
<OnLeftDown>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnLeftDown>
```

While Left Key Down/While Key Down

The following elements can be specified for an WhileLeftDown or WhileKeyDown animation link.

Elements	Description
Text	Script text. Required.
Frequency	Script execution frequency in milliseconds. Range from 1 to 360000. Required if optional schema is used. Default is 1000.

Do not use WhileLeftDown and WhileKeyDown animation links in the same object.

Example:

```
<WhileLeftDown>
<Text>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</Text>
<Frequency>1000</Frequency>
</WhileLeftDown>
```

On Left Key Up/On Key Up

You place the text of a script in the OnLeftUp element. Do not use OnLeftUp and OnKeyUp animation links in the same object.

Example:

```
<OnLeftUp>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnLeftUp>
```

On Left Key Double Click

You place the text of a script in the OnLeftDoubleClick animation link.

Example:

```
<OnLeftDoubleClick>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnLeftDoubleClick>
```

On Right Key Down/On Right Down

You place the text of a script in the OnRightDown element. Do not use OnRightDown and OnRightKeyDown animation links in the same object.

Example:

```
<OnRightDown>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnRightDown>
```

While Right Key Down/While Right Down

The following elements can be specified for an WhileRightKeyDown or WhileRightDown animation link.

Elements	Description
Text	Script text. Required
Frequency	Script execution frequency in milliseconds. A number between 1 and 360000. Default is 1000.

Do not use both WhileRightDown and WhileRightKeyDown animation links in the same object.

Example:

```
<WhileRightDown>
<Text>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</Text>
<Frequency>1000</Frequency>
</WhileRightDown>
```

On Right Key Up/On Right Up

You place the text of a script in the OnRightUp and OnRightKeyUp elements. Do not use both OnRightUp and OnRightKeyUp animation links in the same object.

Example:

```
<OnRightUp>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnRightUp>
```

On Right Key Double Click

You place the text of a script in the OnRightDoubleClick element.

Example:

```
<OnRightDoubleClick>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnRightDoubleClick>
```

On Middle Key Down/On Middle Down

You put the text of a script in the OnMiddleKeyDown element. Do not use OnMiddleDown and OnMiddleKeyDown animation links in the same object.

Example:

```
<OnMiddleDown>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnMiddleDown>
```

While Middle Key Down/While Middle Down

The following elements can be specified for an WhileMiddleKeyDown or WhileMiddleDown animation link.

Elements	Description
Text	Script text. Required
Frequency	Script execution frequency in milliseconds. A number between 1 and 360000. Required if optional schema is used. Default is 1000.

Do not use WhileMiddleDown and WhileMiddleKeyDown animation links in the same object.

Example:

```
<WhileMiddleDown>
<Text>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</Text>
<Frequency>1000</Frequency>
</WhileMiddleDown>
```

On Middle Key Up/On Middle Up

You place the text of a script in the OnMiddleKeyUp and OnMiddleUp animation links. Do not use OnMiddleUp and OnMiddleKeyUp animation links in the same object.

Example:

```
<OnMiddleUp>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnMiddleUp>
```

On Middle Key Double Click

You place the text of a script in the OnMiddleDoubleClick element.

Example:

```
<OnMiddleDoubleClick>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</OnMiddleDoubleClick>
```

On Mouse Over

The following elements can be specified for an OnMouseOver animation link.

Elements	Description
Text	Script text. Required.
Timeout	Timeout in milliseconds. Required if optional schema is used. Default is 250.

Example:

```
<OnMouseOver>
<Text>
<![CDATA[
First line of script text
Second line of script text
N-th line of script text
]]>
</Text>
<Timeout>250</Timeout>
</OnMouseOver>
```

Left Key equivalent

The following elements can be specified for a Left Key equivalent animation link.

Elements	Description
KeyCode	Virtual key code name or empty string. Required. An empty string means no assignment occurs.
KeyFlags	Virtual key flag combination or empty string. Required. An empty string means no assignment occurs. Disabled if no KeyCode is specified.

Example for "CTRL+B":

```
<LeftKey>
<KeyCode>B</KeyCode>
<KeyFlags>Ctrl</KeyFlags>
</LeftKey>
```

Example of a No Key Flag Modifier for "L":

```
<LeftKey>
<KeyCode>L</KeyCode>
</LeftKey>
```

Example of a Both Key Flag Modifiers for CTRL+SHIFT+F7:

```
<LeftKey>
<KeyCode>F7</KeyCode>
<KeyFlags>CtrlShift</KeyFlags>
</LeftKey>
```

Right Key equivalent - not supported

The right key equivalent is not supported by the InTouch XML import functionality.

The following elements can be specified for a right key equivalent animation link.

Elements	Description
KeyCode	Virtual key code name or empty string. Required. An empty string means no assignment occurs.
KeyFlags	Virtual key flag combination or empty string. Required. An empty string means no assignment occurs. Disabled if no KeyCode is specified.

Example:

```
<RightKey>
<KeyCode>B</KeyCode>
<KeyFlags>Ctrl</KeyFlags>
</RightKey>
```

Middle Key equivalent - not supported

The middle key equivalent is not supported by the InTouch XML import functionality.

The following elements can be specified for a middle key equivalent animation link.

Elements	Description
KeyCode	Virtual key code name or empty string. Required. An empty string means no assignment occurs.
KeyFlags	Virtual key flag combination or empty string. Required. An empty string means no assignment occurs. Disabled if no KeyCode is specified.

```
<MiddleKey>
<KeyCode>B</KeyCode>
<KeyFlags>Ctrl</KeyFlags>
</MiddleKey>
```

Unsupported InTouch features

Wizards and ActiveX controls cannot be imported using the XML import functionality.

You cannot import tags or the tag database using the XML import functionality. You can create a file containing tag definitions and use the DBLoad utility to import them to an application's Tagname Dictionary. For more information about using DBLoad, see *AVEVA™ InTouch HMI Application Maintenance*. For information about running DBLoad from the command prompt, see Run DBLoad from the command prompt.

Work with Industrial Graphics in CONNECT

CONNECT is the common cloud repository allowing you to to manage common contents, such as, Industrial Graphics, Controls, and Widgets in the Cloud.

Graphics can be downloaded and uploaded on demand. Graphics are stored in 'Stores' in CONNECT; there are three types of stores – global, tenant and user specific. Within each store users can configure multiple drives. You must have an CONNECT user account to manage graphics in the cloud. Each drive can be configured with different access levels for different users.

Users will have read-only or read-write access depending on the access granted by the administrator.

- Users with Read-Only access can view graphics in the cloud. They can download the graphics to the cloud, but these graphics would be available as read-only. They cannot modify or upload the graphics.
- Users with Read-Write access can view, download, modify, and upload graphics to the cloud.

Multiple users can access the graphics in the same drive, but only one user can edit or save a graphic at one time.

Log into CONNECT

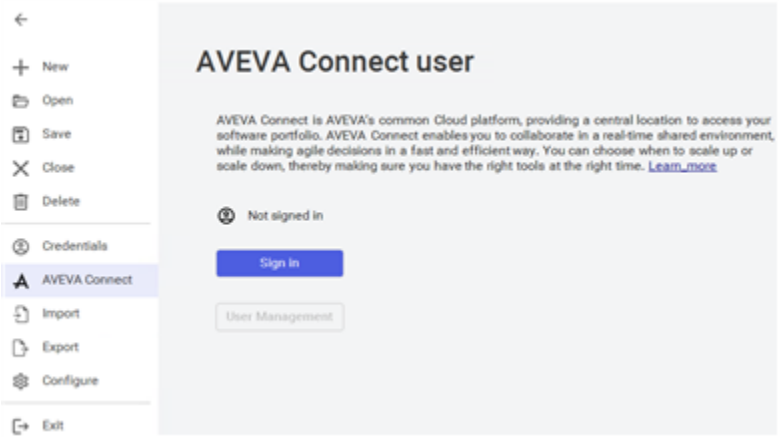
You must have a CONNECT account and authorization from your administration to access CONNECT.

1. If you are using InTouch, launch WindowMaker, then, from the **File** menu, select **CONNECT**.

   If you are using the System Platform IDE, select the **Galaxy** menu, then select **CONNECT**.

   Note: The **CONNECT** option will not be available if you have selected **connected experience** mode under the **Licensing Mode** tab of the Configurator.

   The CONNECT user configuration screen appears.



2. Select **Sign In**.
3. Enter your CONNECT email address.
4. Enter your password.

   If the credentials are verified, you are signed in.

   Note: You must have the Content Contributor role in order to access the CONNECT Industrial Graphics drive.

This is required for both Operations Control connected experience and non-connected experience use.

5. Select the **AVEVA Cloud Integration Studio solution.** For example: Pym Technologies.
6. The AVEVA Drive appears as a separate repository within the Visualization folder.

Note: Once you have selected an Integration Studio solution, it is cached in the browser memory. To switch the Integration Studio solution, you must clear the browser cache.

Navigate between shared drives

Once you are logged into a specific Store you can move between different drives and upload and download graphics.

1. On the **View** menu, in the **Cloud drive** group, select **Shared Drive**.
2. Select a drive from the available options.

   The Visualization folder will refresh to display the selected drive.

Note: You cannot move a graphic which is locked by another user.

Upload/download graphics to the cloud

The AVEVA Drive functions similar to your local Visualization folder. You can upload or create folders on the AVEVA Drive to organize your graphics. With Read-Write access, you can download graphics to the local Visualization folder and use it for your applications. When you download or upload graphics, the contents will be overwritten according to the selected overwrite option

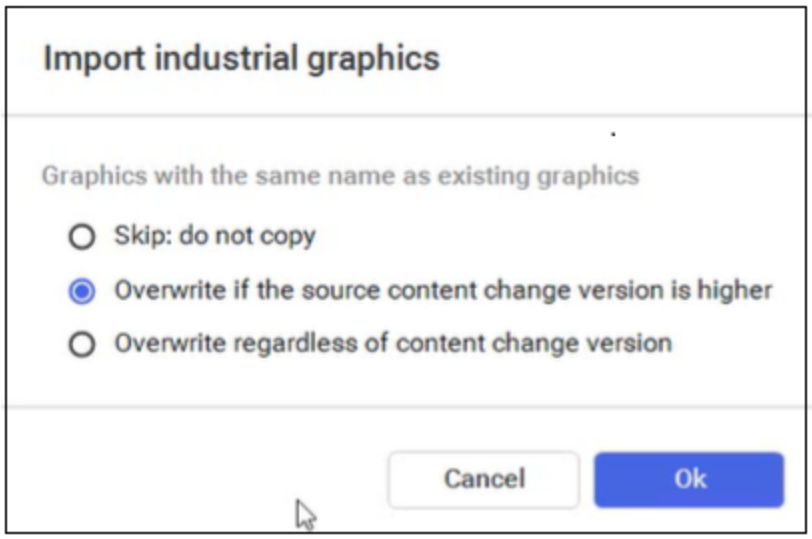
- If the contents already exist in the destination drive, the contents will remain in the same toolset location. Otherwise, the contents will be added in the same toolset path as the source

For the selected embedded contents

- If the embedded contents already exist in the destination drive, the contents will remain in the same toolset location in destination.
- If the embedded content resides in the same toolset location as its parent content or within the same folder hierarchy, then the embedded contents will be added in the same toolset path as the parent. Otherwise, the embedded contents will be added in the same toolset path as the source.

Upload graphics to CONNECT

1. Drag the graphic(s) from the local Visualization folder and drop it to the AVEVA Drive folder.
2. If the graphic already exists in the drive then a dialog appears, requesting to overwrite or skip the upload.



3. Select the appropriate option, and select **OK**.
4. The **Upload Content** dialog appears and displays the progress of the operation. Select **View Details** to view the progress.
5. Select **Close**.

You can also upload folder(s) that contains multiple graphics.

Download graphics to the local visualization folder

Users with Read-Write access can download graphics from the AVEVA Drive to their local Visualization folder.

1. Drag the graphic(s) or folder from the AVEVA Drive and drop it to the local Visualization folder.

   The **Download Content** dialog box requests confirmation of the download.

   - Select **Yes** to continue with the download.
   - Select **No** to cancel the operation.
   - Select **View Details** to view the list of controls and namespaces used. Any errors during downloading are also displayed.
2. If the graphic already exists, the Import industrial graphic dialog box appears.

3. Select the appropriate option.
4. The graphic(s) are downloaded to the local Visualization folder. Embedded graphics are also copied.

Upload/download support for various cloud content

The upload/download support of various cloud content is listed in the table below.

Cloud Content	Upload Support	Download Support
Embedded Client Control and HTML5 Widget	Upload will package the embedded graphics contents including Client Control and HTML5 Widgets as aaPKG format and copy/overwrite to the Cloud.	Download will extract the embedded graphics contents including Client Control and HTML5 Widgets and import/overwrite the control into the local application
Embedded application object graphic	Not Supported	Not Supported
Custom script library	Not Supported	Not Supported
Application Style Library	Not Supported	Not Supported

Manage graphics in CONNECT

You can also create, rename, update, duplicate and delete graphics directly on AVEVA Drive, just like managing local graphics.

- Right-click the AVEVA Drive or any of the folders to create new graphic or toolset.
- Double-click to edit graphics using the graphic editor.
- Right-click and select the corresponding options to rename, duplicate or delete the graphic, and update thumbnails.

  The context menu options that are not supported for the Cloud appear greyed out. For example, Set Web Client Home Symbol, Set Web Client Root Folder, Export Symbol, and Export Localization.

  Note: Updating thumbnails for embedded graphics directly in the AVEVA Drive causes a lot of cloud traffic. Hence, we recommend that you update the thumbnails locally.

Manage graphics with multiple users

Multiple users can use the graphics available on AVEVA Drive at a time. However, a graphic can be edited by only one user at a time. When a graphic in the Cloud repository is being edited by one user, the graphic is checked-out and locked, preventing other users from making any edits. All other users can only view the read-only unmodified graphic when it is checked-out. The latest changes reflect only when it is checked-in the modifying user.

- A lock icon against the graphic icon in the folder indicates that the graphic is checked out by the logged in user.

- An edit icon against the graphic icon in the folder indicates that the graphic is being edited by another user.

After the first user saves and checks in the graphic, the second user can check it out and make changes to it. Once the graphic is saved and checked-in, all users can view the modified graphic.

About cloud graphic versions

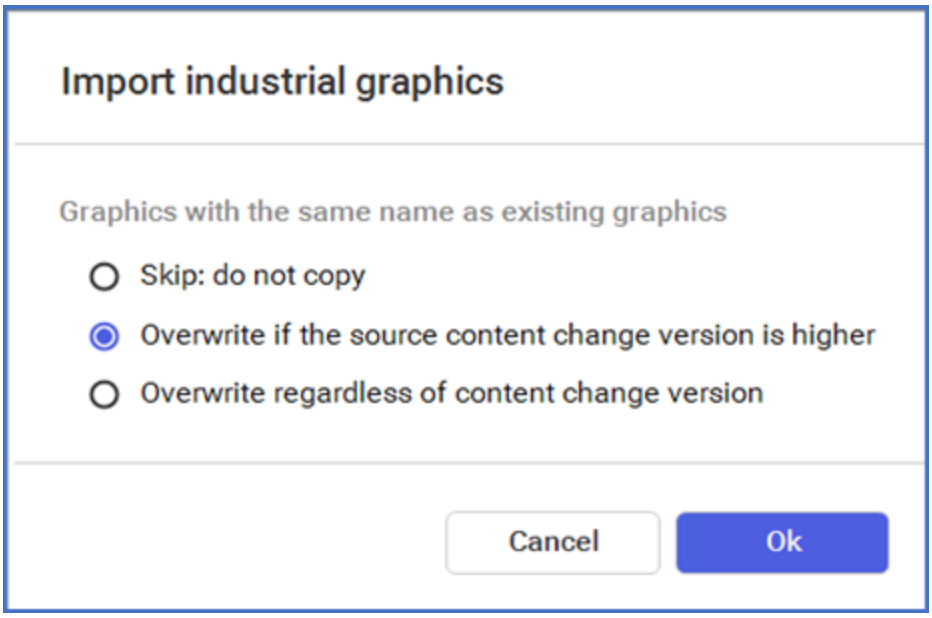
The AVEVA Drive graphics supports Industrial Graphics, Client Controls, and Widgets in the Cloud, published from different products with different versions of the graphic subsystems.

Key concepts of cloud graphic version support

- Industrial Graphics are stored locally and in the Cloud in a specific format that supports all the features and capabilities available in the Graphic subsystem.
- When new features and functionality are added to the Graphic subsystem, the graphics need to be saved and checked-in to be available for other users.
- Different versions of the Industrial Graphics subsystem are identified by a unique, internal version number, not visible to the user. At the time of their release, each product will consume the most current version of the Industrial Graphics subsystem.
- The version number of the Industrial Graphics subsystem is included as part of each stored Industrial Graphic, and is used by the Industrial Graphic subsystem to ensure only a product with a matching or higher version of the Industrial Graphic subsystem is allowed to edit, embed, or download that Industrial Graphic.
- All users connecting to the same Cloud repository will see the same set of graphics. The ability to edit, embed, or download a Cloud graphic will depend on the version of the Industrial Graphics subsystem in the product through which the user is connecting.
- The Industrial Graphics saved to the Cloud repository by a product with a newer version of the Industrial Graphic subsystem will appear disabled in the Visualization folder of the products with older version accessing the same Cloud. The user from the old product version will not be able to edit, embed, or download the graphic repository.
- Just like the local Visualization folder, graphic names must be unique within any single Cloud repository. This holds true even if the graphics are stored by products with different versions of the Industrial Graphics subsystem.

Overwrite graphic contents in the cloud

If the graphic content being uploaded or downloaded already exists, the system prompts to overwrite the graphic.



The overwriting depends on the graphic subsystem versions of the source and the destination, and the change log version of the graphic.

- Option 1 will skip the overwrite regardless of the graphic subsystem versions or the change log version of the source and the destination.
- If the graphic subsystem version is different between source and destination, both options 2 and 3 will overwrite the graphic, regardless of change log version.
- If the graphic subsystem version is same between source and destination:
  - Option 2 will overwrite only if source has higher change log version.
  - Option 3 will overwrite regardless of the change log version.

Overwrite custom client controls and widgets in the cloud

The guidelines of Cloud Graphic Version Management do not apply to the Custom Clients Controls and Widgets at the time of this release. Client Controls and Widgets are supported for both download or upload operation. However, overwrite is not supported. If a Custom Client Controls or Widgets is already available at the target location, irrespective of the client control internal version, the overwrite will be skipped. To overwrite the source repository client control, you must first delete the client control in the source repository. After that, upload/ download the client control will be successful.

Note: If a new Custom Client Controls or Widget is uploaded to the Cloud, all other users connected to the same Cloud repository must restart the WindowMaker for the latest Custom Client Controls or Widget to be available.

Cloud graphic versions - scenarios and examples

This section provides a non-exclusive list of high level scenarios and expectation for Cloud graphic version handling.

Let's consider two products P1_Old and P2_New, of different versions, which connect to the same tenant in AVEVA Cloud.

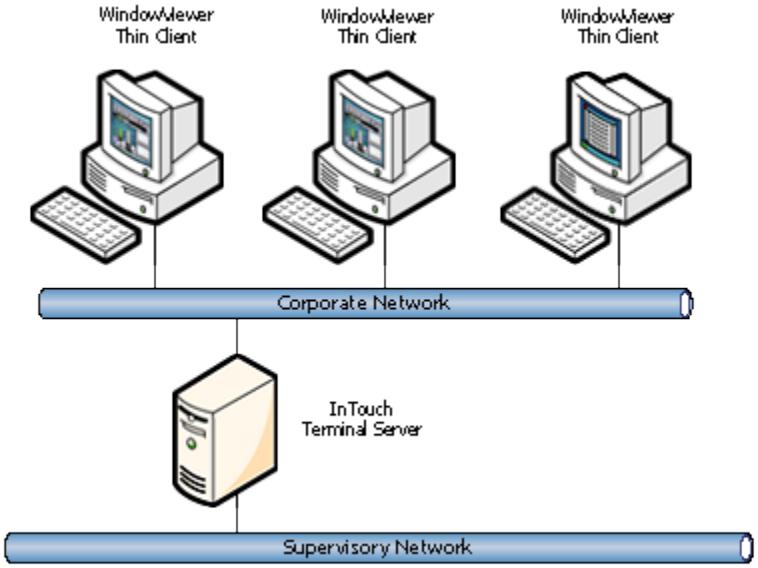
- P1_Old has the older version of Graphic Subsystem and P2_New has the newer version of Graphic Subsystem.
- To begin with, the cloud repository has graphics S1 and S2 uploaded by P1_Old.

Scenario / Steps	Expectation
Scenario 1: P2_New creates S3. P1_Old attempts to edit S3.	S3 is saved in newer graphic version and hence it cannot be edit, embed, or download in older version. P1_Old receives a message stating that S3 is saved in newer version and cannot be opened.
Scenario 2: P2_New opens S2. P2_New saves S2 in Cloud. Then, P1_Old attempts to open S2.	S2 created in older graphic version can be opened and saved in the newer graphic version. S2 is saved in newer graphic version and hence it cannot be edit, embed, or download in older version. P1_Old receives a message stating that S3 is saved in newer version and cannot be opened.
Scenario 3: P1_Old saves S1 and S2, where S2 is embedded in S1 P2_New saves and updates only S2. P1_Old attempts to edit S1.	S1 is opened because it is still the same version in P1_Old. The embedded S2 will fail to load.
Scenario 4: P2_New creates a new S4 in Cloud. P2_New embeds S1, S2, and S3 in S4.	S1 is in older graphic version but still can be loaded in S4 (newer version). S2 and S3 can be embedded because they are the same graphic version as S4.
Scenario 5: P1_Old creates a new S5 in Cloud. P1_Old embeds S1, S2, and S3 in S5.	S1 can be embedded because they are the same graphic version. S2 and S3 cannot be embedded due to newer version.
Scenario 6: P1_Old download S3.	S3 cannot be downloaded because it is in newer graphic version.
Scenario 7: Cloud repository has a graphic S3 which is in newer graphic version. P1_Old attempts to upload a graphic with same name S3.	By selecting Option 3, the graphic can be overwritten.
Scenario 8: P2_New has a graphic S8. P1_Old creates locally S8, S9, and embed S8 in S9. P1_Old uploads S9 to Cloud.	S9 is uploaded to Cloud. S8 is prevented upload because a newer version exists in the Cloud. By selecting Option 3, the content can be overwritten.

Deploy

Deploy and work with Terminal Services and Remote Desktop Services

Terminal Services is a configurable service included in the Microsoft Windows Server operating systems that runs Windows-based applications centrally from a server. In Terminal Services, client computers access the server node, where multiple instances of InTouch software applications run simultaneously.



The Terminal Services environment has three main parts:

- **Terminal Services Server**. The server manages the computing resources for each client session and provides client users with their own unique environment. The server receives and processes all keystrokes and mouse actions performed at the remote client and directs all display output for both the operating system and applications to the appropriate client. All Terminal Services application processing occurs on the server.
- **Remote Desktop Protocol (RDP).** A Remote Desktop Protocol (RDP) client application passes the input data, such as keystrokes and mouse movements, to the server.
- **Client**. The Terminal Services client performs no local application processing; it just shows the application output. You access Terminal Services from a client by running the **Terminal Services Client** command on the Windows Program menu. When you connect to the Terminal server, the client environment looks the same as the Windows server. The fact that the application is not running locally is completely transparent.

For more information about Terminal Services, including features and benefits, see your Microsoft documentation.

Plan for Terminal Server applications

Important: We recommend that you install applications on a test server before you deploy them in your production environment.

Before you install Terminal Services

- Identify the client applications (for example, InTouch) that you need to install on the server.
- Identify the hardware requirements for your clients.
- Determine the server configuration required to support clients.
- Identify the licenses required for Terminal Services as well as other applications that you will be running.
- Understand how some aspects of InTouch applications run under Terminal Services, such as alarms, security, I/O, and scripts.

Deploy InTouch applications in a Terminal Services environment

When deploying InTouch applications in a Terminal Services environment, a separate InTouch application should be deployed for each node.

Alarms in a Terminal Services environment

By using the Distributed Alarm System with Terminal Services for InTouch, alarm clients running on different terminal sessions can select what alarm to show and how to present it.

Alarm Providers identify themselves by a name that uniquely identifies their application, and the instance of their application. This information is made available to the Distributed Alarm System when the Alarm Provider or the Alarm Consumer registers with the Distributed Alarm System.

The node on which an Alarm Provider is running is identified by a name that uniquely identifies the computer node in the system. This information is made available to the Distributed Alarm System when an instance of it starts up on the computer node.

When an alarm event is logged, the node and complete Alarm Provider name identify the source of the alarm.

When an alarm is acknowledged in a Terminal Services environment, the Operator Node that gets recorded is the name of the client computer running the Terminal Services session used by the operator. If the node name of the computer cannot be retrieved, its IP address is used instead.

Note: Alarm Providers are not supported on Terminal sessions. They are only supported on the Terminal Console.

Security in a Terminal Services environment

Use application security to secure your InTouch application, IndustrialSQL Server, and other sensitive information systems.

- Use the $Operator system tag to secure your application. You can then control operator access to specific functions by linking those functions to internal tags.

  For more information about using the $Operator system tag, see About Securing InTouch.
- Replace the GetNodeName() function with the newer TseGetClientId() function to identify the client

computer. When using Terminal Services, the GetNodeName() function returns the name of the terminal server, not the name of the client computer.

Use security auditing to monitor intrusion attempts. If you suspect that your system is under any sort of attack, then you can enable logging for an array of auditable events. By default, security logging/auditing is disabled because it usually requires excessive processing resources.

Caution: Security auditing requires significant resources. Enable auditing when you evaluate your pilot server to accurately estimate your InTouch application hardware requirements.

I/O in a Terminal Services environment

The InTouch HMI cannot start I/O Servers in a Terminal Services environment. Depending on the sequence that view sessions start, you may need to use the IOReinitialize() function. All servers (I/O devices or view applications) must be running before starting an application that reads values from these servers.

Avoid receiving an "initializing I/O" error message when WindowViewer starts

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.
   The WindowViewer configuration screen appears.
3. On the **Preferences** tab, in the **I/O** section, clear the **Start Local Servers** check box.

Script execution in a Terminal Services environment

Because all applications running in Terminal Services use a single timing reference (server clock), scripts may not run during periods of excessive CPU loading. Abnormal CPU loading can be caused by excessive video processing or when several applications have the same script triggers defined (such as an End-of-Shift event). It is possible, therefore, that if the server is busy processing scripts from many clients, it may not start a script on another client during the interval when the timer would normally start the script. This can prevent the script from running on the client.

To ensure scripts run correctly, combine scripts with common triggers and move them to a single application, such as a tag server. This is one of the primary reasons for pilot deployment. Pilot deployment gives you an opportunity to conduct stress testing to determine if your hardware selection is adequate.

Log on to a terminal session properly to run InTouch

Each session must be logged on with a unique account. This can be done manually or Terminal Services can be configured to enforce unique logons.

Note: Running with the same logon account on multiple sessions can cause corruption and other unexpected

results.

Alarm query syntax in a Terminal Service environment

The alarm query syntax for a session's alarms is:

```
\\ServerNodename\InTouch!$System
```

The alarm query syntax for console alarms includes a colon (:) after the node name; for example:

```
\\ServerNodename:\InTouch!$system
```

Miscellaneous limitations in a Terminal Services environment

The following table describes the limitations and suggested solutions to run applications on a terminal server.

Feature	Supported?	Comment
WindowViewer	Yes	WindowViewer is not supported running as a service under Terminal Services.
DDE to an I/O Device or MS Office (for example, Excel)	No	Use a tag server (console or separate computer). This includes DDE QuickScripts: WWExecute(), WWpoke() and WWRequest()
DDE from MS Office (for example, Hot-link configured in Excel)	Yes	Excel and the InTouch HMI must be running in the same session.
Historical Trending	Yes	Use a tag server or NAD to log values. Multiple sessions may read the same historical files, but only a console can write to historical files.
InTouch Alarm Logger	Yes	--
MEM OLE Automation	Yes	--
Printing Alarms	No	--
Retentive tags	Yes	Must use NAD.
SQL Access (ODBC)	Yes	Database should be on a separate computer.

Feature	Supported?	Comment
SuiteLink to an I/O Device or another InTouch application.	Yes	When communicating to another view session, include the Terminal Server node name and append the IP address of the desired session to the application name. For example, view10.103.25.6. I/O Servers are not supported in client sessions.

Retrieve information about the InTouch client session using scripts

You can use the following InTouch QuickScript functions for Terminal Services.

- TseGetClientId() function
- TseGetClientNodeName() function
- TseQueryRunningOnConsole() function
- TseQueryRunningOnClient() function

TseGetClientId() function

Returns a string version of the client ID (the TCP/IP address of the client) if the View application is running on a Terminal Server client. This client ID is used internally to generate SuiteLink server names and logger file names. Otherwise, the TseGetClientId() function returns an empty string.

Syntax

```
MessageResult=TseGetClientId();
```

Example

The client IP address 10.103.202.1 is saved to the MsgTag tag.

```
MsgTag=TseGetClientID();
```

TseGetClientNodeName() function

Returns the client node name if the View application is running on a Terminal Server client assigned a name that can be identified by Windows. Otherwise, the TseGetClientNodeName() function returns an empty string.

Syntax

```
MessageResult=TseGetClientNodeName();
```

Example

The client node name is returned as the value assigned to the MsgTag tag.

```
MsgTag=TseGetClientNodeName();
```

TseQueryRunningOnConsole() function

The TseQueryRunningOnConsole() function can be run from a script to indicate whether the View application is running on a Terminal Services console.

Syntax

```
Result=TseQueryRunningOnConsole();
```

Return value

Returns a non-zero integer value if the View application is running on a Terminal Services console. Otherwise, the TseQueryRunningOnConsole() function returns a zero.

Example

IntTag is set to 1 if WindowViewer is running on a Terminal Services console.

```
IntTag=TseQueryRunningOnConsole();
```

TseQueryRunningOnClient() function

Returns a non-zero integer value if the View application is running on a Terminal Services client. Otherwise, it returns a zero.

Syntax

```
Result=TseQueryRunningOnClient();
```

Return value

Returns 0 if View is not running on a Terminal Services client.

Example

IntTag is set to 1 if WindowViewer is running on a Terminal Services client.

```
IntTag=TseQueryRunningOnClient;
```

Remote Desktop Services overview

Remote Desktop Services, formerly Terminal Services, is a server role in Windows Server® 2008 R2 and later versions that provides technologies that enable users to access Windows-based programs that are installed on a Remote Desktop Session Host (RD Session Host) server, or to access the full Windows desktop. With Remote Desktop Services, users can access an RD Session Host server from within a corporate network or from the Internet.

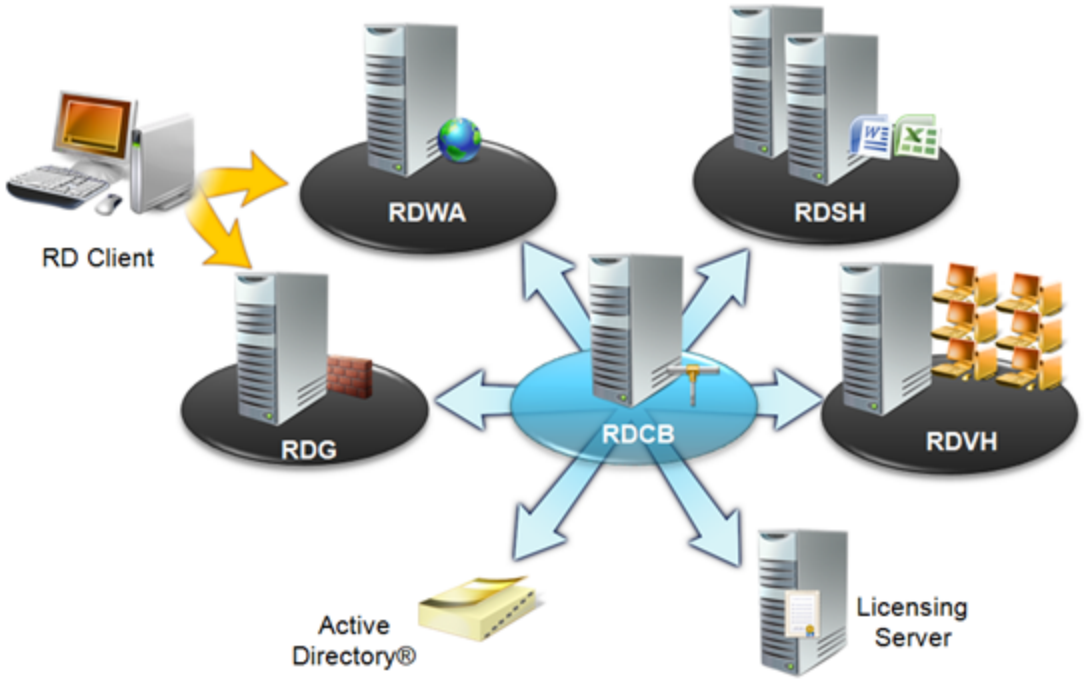
When a user accesses a program on an RD Session Host server, the program runs on the server. Each user sees only their individual session. The session is managed transparently by the server operating system and is independent of any other client session. Additionally, you can configure Remote Desktop Services to use Hyper-V™ to either assign virtual machines to users or have Remote Desktop Services dynamically assign an available virtual machine to a user upon connection.

For more information about Remote Desktop Services, see the Remote Desktop Services page on the Windows Server 2008 R2 TechCenter (http://go.microsoft.com/fwlink/?LinkId=138055).

Remote Desktop Services role services

Remote Desktop Services is a server role that consists of several role services. In Windows Server 2008 R2 and later versions, Remote Desktop Services consists of the following role services:

- **RD Session Host**: Remote Desktop Session Host (RD Session Host), formerly Terminal Server, enables a server to host Windows-based programs or the full Windows desktop. Users can connect to an RD Session Host server to run programs, to save files, and to use network resources on that server.
- **RD Web Access**: Remote Desktop Web Access (RD Web Access), formerly TS Web Access, enables users to access RemoteApp and Desktop Connection through the Start menu on a computer that is running Windows 7 or through a Web browser. RemoteApp and Desktop Connection provides a customized view of RemoteApp programs and virtual desktops to users.
- **RD Licensing**: Remote Desktop Licensing (RD Licensing), formerly TS Licensing, manages the Remote Desktop Services client access licenses (RDS CALs) that are required for each device or user to connect to an RD Session Host server. You use RD Licensing to install, issue, and track the availability of RDS CALs on a Remote Desktop license server.
- **RD Gateway**: Remote Desktop Gateway (RD Gateway), formerly TS Gateway, enables authorized remote users to connect to resources on an internal corporate network, from any Internet-connected device.
- **RD Connection Broker**: Remote Desktop Connection Broker (RD Connection Broker), formerly TS Session Broker, supports session load balancing and session reconnection in a load-balanced RD Session Host server farm. RD Connection Broker is also used to provide users access to RemoteApp programs and virtual desktops through RemoteApp and Desktop Connection.
- **RD Virtualization Host**: Remote Desktop Virtualization Host (RD Virtualization Host) integrates with Hyper-V to host virtual machines and provide them to users as virtual desktops. You can assign a unique virtual desktop to each user in your organization, or provide them shared access to a pool of virtual desktops.



Secure your Remote Desktop Services (RDS) connections

Safeguard against attacks with the following security practices

1. Use strong passwords

   Use a strong password on all accounts with access to Remote Desktop.

2. Update your software

   Make sure you are running the latest versions of both the client and server software by enabling and auditing automatic Microsoft Updates.

3. Set an account lockout policy

   By setting your computer to lock an account for a period of time after a number of incorrect guesses, you will help prevent "brute-force" attack.

4. Use Two-factor authentication

   RD Gateways support smartcard two-factor authentication.

5. Change the listening port for Remote Desktop

   Prevents network attacks and worms that attempt to access the default Remote Desktop port (TCP 3389).

6. Use RD Gateways

   RD Gateway restricts access to Remote Desktop ports while supporting remote connections through a single "Gateway" server. When using an RD Gateway server, all Remote Desktop services on your desktop and workstations are routed through the RD Gateway. The RD Gateway server listens for Remote Desktop requests over HTTPS (port 443), and connects the client to the Remote Desktop service on the target machine. Refer to the steps here: http://technet.microsoft.com/en-us/library/cc770601.aspx

7. Configure Network Level Authentication for Remote Desktop Services Connections

   Network Level Authentication requires that the user be authenticated to the RD Session Host server before a

session is created. Network Level Authentication increasing availability of the RD Session Host server (reduces the risk of denial-of-service attacks of the RD Session Host server). https://technet.microsoft.com/en-us/library/hh831778.aspx

8. Configure Server Authentication and Encryption Levels

   By default, Terminal Services sessions use native Remote Desktop Protocol (RDP) encryption. However, RDP does not provide authentication to verify the identity of a terminal server. You can enhance the security of Terminal Services sessions by using Transport Layer Security (TLS) 1.0 for server authentication and to encrypt terminal server communications. The RDS and the client computer must be correctly configured for TLS to provide enhanced security. By default, RDS connections between the client and server are encrypted at the highest level of security available (128-bit), ensuring integrity and confidentiality of the data transmitted.

Windows Server 2016 Remote Desktop Services best practices

For the Windows Server 2016 environment you can implement the below best practices for Remote Desktop Services

- Use Multi-Factor Authentication

  Leverage the power of Active Directory with Multi-Factor Authentication to enforce high security protection. Refer to the Microsoft documentation here: https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/rds-plan-mfa

- Secure data storage with User Profile Disks (UPDs)

  User Profile Disks (UPDs) allow user data, customizations, and application settings to follow a user within a single collection. A UPD is a per-user, per-collection VHD file saved in a central share that is mounted to a user's session when they sign in - the UPD is treated as a local drive for the duration of that session. Refer to the Microsoft documentation here: https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/rds-plan-secure-data-storage

Distribute applications

Distributed applications typically have a central development station, central data storage, and client stations. You use InTouch Network Application Development (NAD) to build and maintain distributed applications. NAD allows many client stations to maintain a copy of a single application without restricting the development of that application. Using an individual copy of the application provides Viewer redundancy. Client stations are automatically notified when the application changes. You can create single computer, client-based, and server-based InTouch applications.

You can also manage and deploy InTouch applications using the System Platform IDE.

Supported InTouch architectures

Supported InTouch network architectures are:

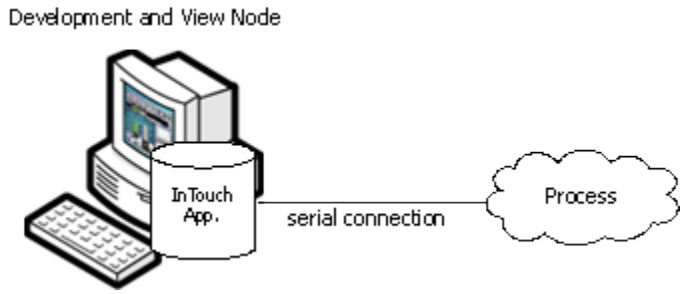
- Single computer
- Client-based

- Server-based
- NAD

Single computer architecture

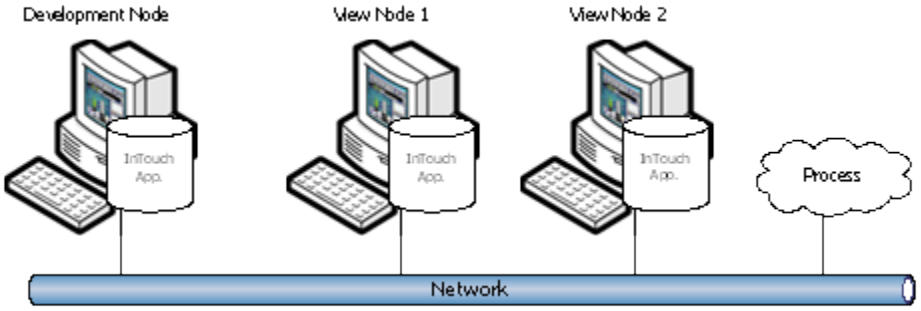
A single computer application typically consists of one non-networked computer that functions as the primary operator interface. This computer is connected to the industrial process with a direct connection, such as a serial cable.

In this architecture, you develop the InTouch application on the single computer. You can copy the application to another computer to modify it and then copy it back to the original computer.



Client-based architecture

In a client-based architecture, there is a unique copy of one InTouch application for each computer running WindowViewer (View node) or in a unique location on a network server. In the following example, an application is developed and tested on the development node and then copied to each View node.



There is inherent redundancy because each node can be self-sufficient, and there is no limit to the number of View nodes you can use.

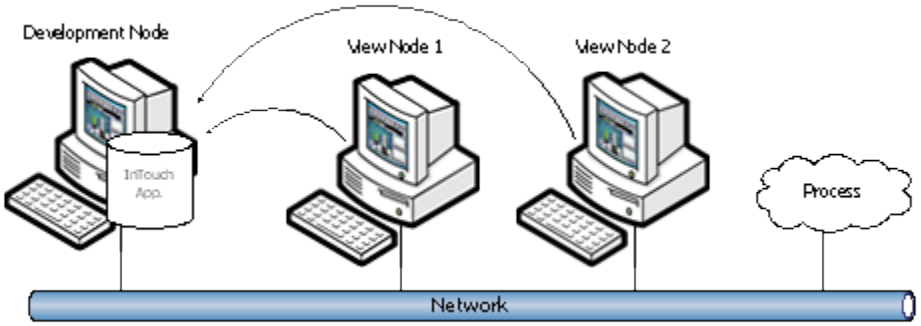
Each View node must have an identical copy of the application and identical access to any network data sources, such as I/O Servers or the IndustrialSQL Server. However, each View node maintains a separate conversation with the shared server, which can result in increased network loading.

You can modify and test the application on the development node without affecting the running process. However, you must distribute the modified application to the View nodes. You must shut down each View node locally, copy the new application to it, and then restart.

Server-based architecture

A server-based architecture distributes a common InTouch application to several View nodes. In the following

figure, two View nodes access the same application from the development node.



For each View node:

- A logical drive must be mapped to the shared network drive of the development node.
- The shared application must be registered with the InTouch program.
- The computer must have identical access to any data sources referred to by the application. There are also ways to define the data source locations by using a combination of scripts to identify the node name and change each data location based on that name.

In this architecture, there is a single application to maintain. View nodes are automatically updated when the application changes and WindowViewer restarts.

Disadvantages of this architecture are:

- Development of application is restricted
- No redundancy if the development node goes down
- All nodes must have the same screen resolution

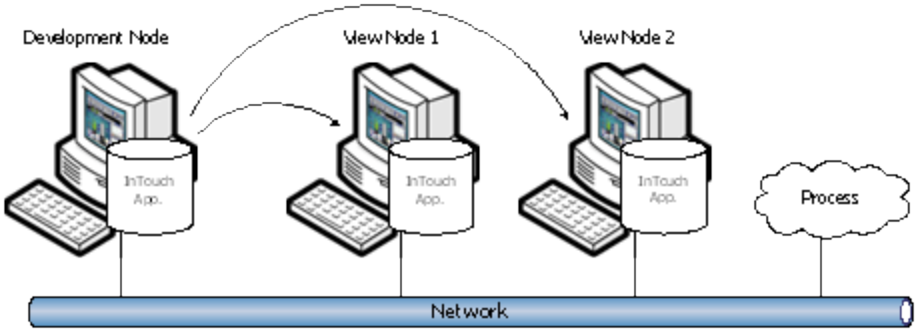
Network Application Development (NAD)

In the Network Application Development (NAD) architecture, you maintain a master copy of an application on a central network location, which is usually the development node. Each View node copies the application to a user-defined location and runs it.

When you notify clients of application changes (using the Notify Clients command on the WindowMaker Special menu), a flag is set in the application directory, which is then read by the View nodes.

You can configure how you want application changes handled for the View nodes. These range from ignoring the changes to automatically shutting down and restarting the View node, which reloads the master application.

In the following figure, the two View nodes have the master application registered from the development node, but actually run it locally on their computers.



Note: If you configure your application to write historical data to the master application node's application directory, all NAD nodes attempt to write their historical data to the master application. To avoid this, on each NAD node, configure historical data to write to a local directory, not the master application node.

If you are distributing a large, complex application to numerous nodes, slow system response time may be apparent on the initial download. Updates, however, are optimized. Application transfer may be a problem for slow networks or over serial connections.

Also, be aware of other network constraints, such as the user of routers that filter out certain types of network traffic and addresses.

Plan networked applications

Regardless of the architecture you choose when building your InTouch application, it is important to consider:

- Access to I/O data sources.
- Access to shared files.
- Where data is logged.
- Any special network requirements.

I/O data access for networked applications

The InTouch HMI uses Access Names to reference real-time I/O data. Each Access Name equates to an I/O address, which consists of a node name, an application, and a topic. In a distributed application, I/O references can be set as global addresses to a network I/O Server or local addresses to a local I/O Server.

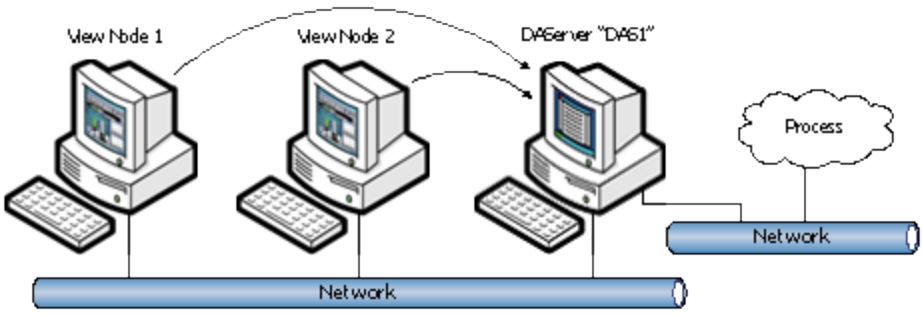
Note: InTouchView is restricted to the single Galaxy Access Name. You cannot create other Access Names for InTouchView. For more information about the restrictions of InTouchView, see View applications at runtime.

The View node must have the same access to data sources as the development node.

Use global I/O addresses

Global addresses to I/O data allow all View nodes to share a common network-based I/O Server. This eliminates the need for multiple I/O Servers, but is less fault-tolerant and can result in lower overall performance.

In the following figure, two View nodes are running a copy of the same application. Both View nodes refer to the same I/O data source. Because each application uses a fully qualified I/O address for the data source, all references point to the same I/O Server.



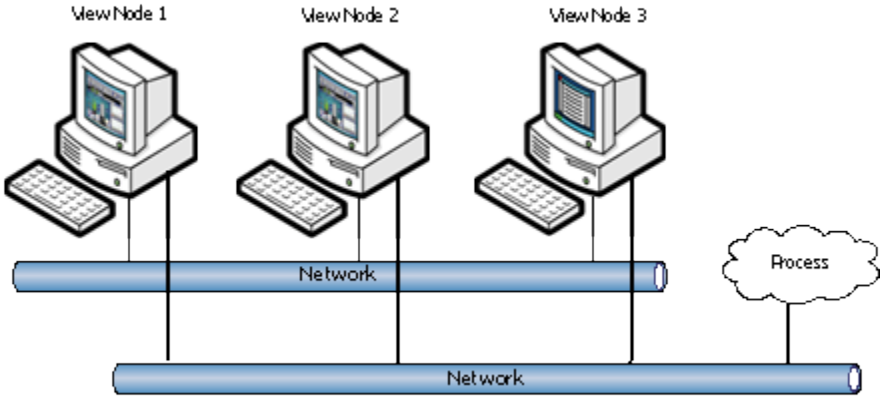
You can set up an InTouch application to identify an element of data stored on another node by using a three-part addressing convention in an Access Name. The Access Name addressing convention includes the node name, application name, and topic name where the remote data is located. An InTouch application obtains remote data using the Access Name in combination with an item name. For more information about defining an Access Name for a remote I/O Server, see Data access with I/O in *AVEVA™ InTouch HMI Application Development.*

Note: When you create Access Names in WindowMaker, if the Access Name uses the SuiteLink protocol, the software prevents Access Names from accessing the same node, application and topic. Do not use the IOSetAccessName() function to redirect Access Names to duplicate ones during run time or else the redirected Access Name will not work.

Use local I/O addresses

Local addresses to I/O data are used when each View node has its own I/O Server. This architecture provides fault-tolerant operation, as each View node can continue to run independently if the network goes down.

In the following figure, two View nodes run copies of the same application that refer to their own I/O data source. Because each application uses a local I/O address for the data source, each reference points to the local I/O Server.



Using a local I/O Server significantly increases the load on the process connection network. For example, three nodes triples the traffic created by one node, as each node's requests must be separately processed.

For more information about defining an Access Name for a local I/O Server, see Data access with I/O in *AVEVA™ InTouch HMI Application Development.*

SuiteLink

The SuiteLink communications protocol is based on the TCP/IP protocol. Use SuiteLink for your high-speed

industrial applications, as it provides these features:

- Value Time Quality (VTQ), in which a timestamp and quality indicator are associated with all data values delivered to VTQ-aware clients. The InTouch HMI is a VTQ-aware client whose tag data is delivered with a VTQ indicator.
- Extensive diagnostics of the data throughput, the server loading, computer resource consumption, and network transport are made accessible through the Microsoft Windows operating system performance monitor.
- Consistent high data volumes can be maintained between applications regardless if the applications are on a single node or distributed over a large number of nodes.

SuiteLink is not a replacement for DDE, FastDDE, or NetDDE. Each connection between a client and a server depends on your network requirements.

Access to shared files

In a distributed application, file references can be set up as

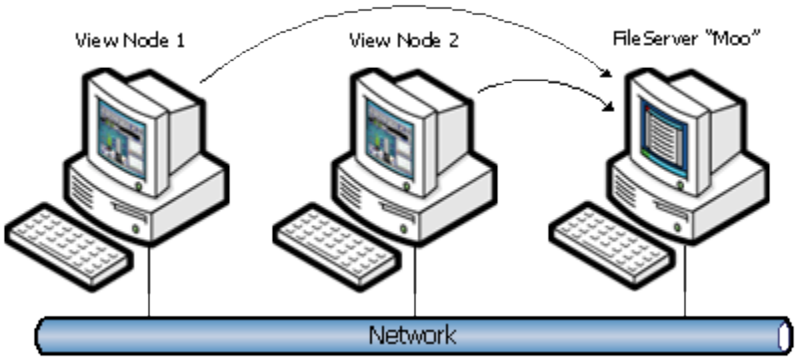
- Global addresses to a network file server.
- Local addresses to local files.

The View node must have the same access to data sources as the development node.

Use global addresses to file data

You can set up global addresses to file data so that all View nodes share a common network-based set of files. This provides single-source maintenance of the files, but it is less fault-tolerant than local copies.

In the following figure, two View nodes are each running a copy of the same application, but reference the same recipe file. Because each application uses a drive letter mapped to a fully-qualified network path for the file, all references point to the same file.



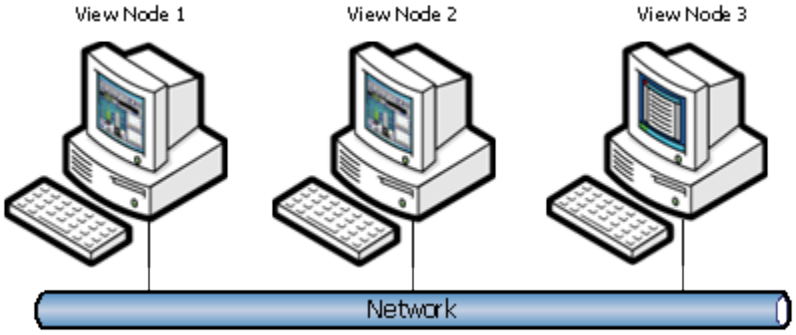
Set up a shared file

1. Map a network drive to the shared path containing the referenced files. For example, G:\Directory\Recipe.csv, where "G:\" is the mapped drive letter that refers to \\Moo\Share. You must map this same drive on every View node.
2. In scripts, reference the shared path. For example:

```
RecipeSelectRecipe("G:\Directory\Recipe.csv", "review", "RecipeName");
```

Use local addresses to file data

You can use local addresses to file data when each View node has its own copy of the file. In the following figure, three View nodes are each running a copy of the same application and reference the local copy of a recipe file.



In this example, the local address is:

```
C:\Directory\Recipe.csv
```

where "C:\" is the local drive.

In scripts, reference the local path. For example:

```
RecipeSelectRecipe("C:\Directory\Recipe.csv", "review", "RecipeName");
```

This architecture is fault-tolerant. However, you must copy any file changes to all the View nodes.

Any file access should be "Read Only" and modification to the local file should not be permitted.

Access to shared files through UNC

You can use a Universal Naming Convention (UNC) address anywhere that you would normally enter a file path, such as for application directory entries, configuration items, and distributed alarms. If you use UNC names, you do not need to create mapped drives.

A UNC address is in the form of \\Node\Share\Path, where:

- Node is the name of the computer that contains the file share.
- Share is the logical name assigned to the shared folder on that computer.
- Path is the normal path to that file with respect to the share.

Note: If you are using SuiteLink, the node name is limited to 15 characters.

Before you can access a file through UNC, you must create a file share on the computer you want to access. For more information, see your Windows documentation.

For example, assume that you have a computer with the network name of "EngineRm" that you have shared the root drive "C:\" with the share name of "Root". To set up a UNC path to the "C:\IT\Apps\Boiler" application you must use the following UNC:

\\EngineRm\Root\IT\Apps\Boiler

If the "Boiler" directory itself was shared as "Boiler," the UNC could be shortened to:

\\EnginerRm\Boiler

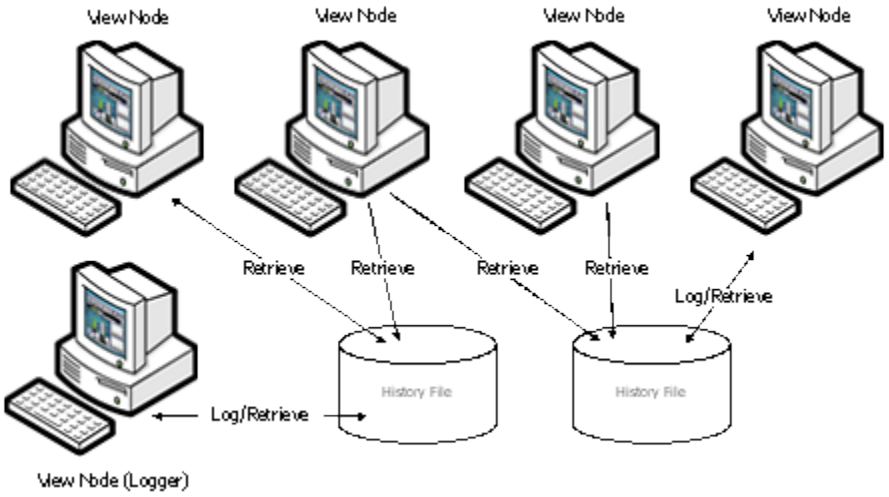
No path is required if the share is a path specified in the PATH environment variable.

Note: If you need to write to a file referred to by a UNC address, the share must be a read/write share, even on a local node. If you create a share that is password-protected, you will not be able to access the share with a UNC unless you first set up a network drive mapping. You can set up a drive mapping from the remote node by using Windows Explorer.

Log data in a distributed environment

You can use the InTouch distributed history system to retrieve historical data from any InTouch application on the network. This system also allows for remote retrieval of data from multiple history databases simultaneously. These databases are called history providers.

Only one InTouch node can log to a distributed history file. However, an unlimited number of InTouch nodes can view the contents of the file.



A remote node retrieving data from a history file may not see data for the last hour of data (based on the logger node's time). Remote trends can only view data that has been written to the logging node's disk.

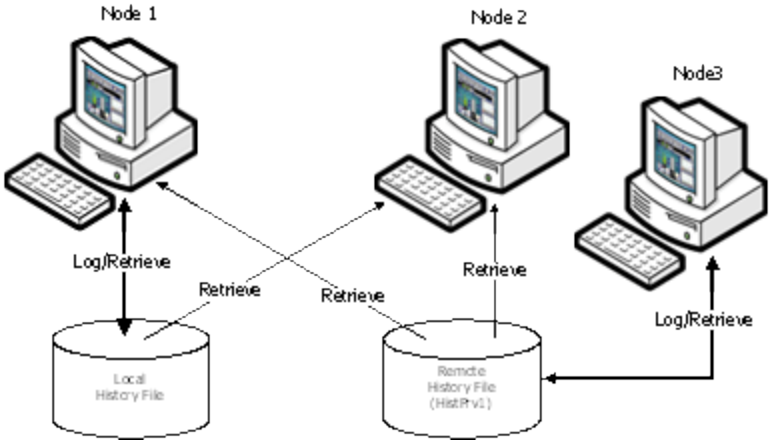
Data for each tag checked for 'Log Data' is automatically written to disk after 22 samples for that tag have been collected. If the HTUpdateToCurrentTime() function is called, data is written to disk regardless of the number of samples collected. By default, data is written to disk once an hour. You can change this interval by adding the following line to the INTOUCH.ini file:

```
ForceLogging=X;
```

Where X is minutes and can be set to any interval between 5 and 120.

Note: The NetDDE Helper service must be running when you use the distributed history system.

The following figure shows the configuration of a typical distributed history system using Network Application Development (NAD) to distribute the application.



Nodes 1 and 2 contain copies of the same InTouch application; however, the application is configured to allow only Node 1 to log to a local history file, whereas either node can retrieve from the local history file or the remote history file. Node 3 is also logging to and retrieving from the remote history file location. Node 3, the history provider, is assigned the name HistPrv1. Node 1 is both a development and run-time station, while Node 2 is just a run-time station.

Create this type of application

1. Create a history provider list. See Configure remote history providers.
2. Create and configure a historical trend object. For more information, see Trending tag data in *AVEVA™ InTouch HMI Application Development*.
3. Configure the application for distributed logging. See Configure distributed historical logging.
4. Distribute the application. See Configure an InTouch application for NAD.

You can distribute your application manually or by using NAD. When you distribute your application, the historical provider list file is distributed as part of the application.

After you have distributed your application, you can run the View nodes and retrieve both local tags and tags from a remote history provider. While the application runs on all the View nodes, only the logging node logs to the historical log file; other nodes can only read from it.

Configure remote history providers

You must specify a name and network location for each remote history provider that you want to use with the InTouch HMI. You can use either a remote InTouch history provider or a remote IndustrialSQL Server history provider.

Note: A remote history provider cannot be configured for an InTouchView application. For more information about the limitations of InTouchView applications, see InTouchView applications.

While the local InTouch application is considered a history provider, you do not need to define it for your application.

If you reference an undefined history provider in an application, WindowViewer ignores the reference and an error message is written to the Logger.

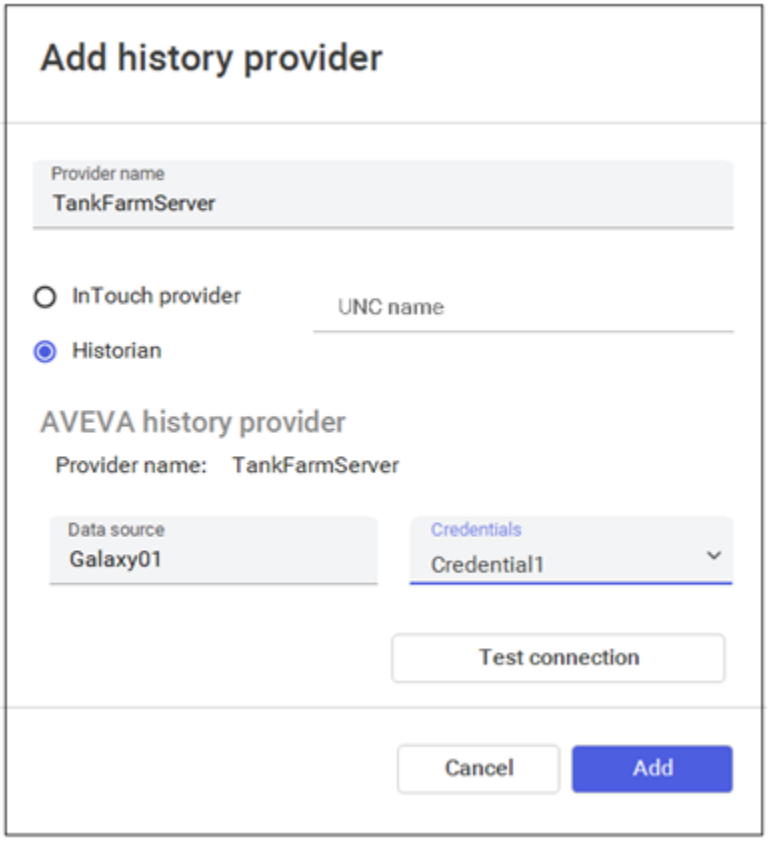
The HistData utility cannot retrieve historical information from a Historian provider.

Configure a history provider

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **Distributed Name Manager**.

   The Distributed Name Manager configuration screen appears, with separate tabs for the **Distributed Alarms** and the **Distributed History**.
3. On the **Distributed History** tab, select the **+** icon to add a history provider, or press Alt+A.

   The **Add history provider** dialog appears.



4. In the **Provider name** box, type the name you want to use for the new historical provider.

   A provider name can be 16 alphanumeric characters or fewer.
5. Configure an InTouch history provider
   a. Select **InTouch provider**.
   b. In the **UNC name** box, type the UNC path to the InTouch application directory and then select **Add**.

      The UNC path format is:

      ```
      \\node_name\volume_name\directory\
      ```

      If the UNC location is password-protected, you must first establish a node connection using Windows Explorer.
6. Configure an AVEVA history provider
   a. Select **Historian**.
   b. In the **Data source** box, type the node name of the server where the Historian Server is installed.
   c. From the **Credentials** drop-down list, select the credentials for authentication.

Note: For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credential Manager in AVEVA™ InTouch HMI Application Development.

7. Select **Test connection** to verify the connection to the Historian Server database. A message appears indicating whether the connection to the database is successful or not.
8. Select **Add** to close the dialog box.

   The Historian Server node appears in the History Providers list.

Dynamically configure remote history providers

At run time, you can also dynamically configure a historical trend's remote history provider by creating a script that specifies the remote history provider tag references in the HTSetPenName() function. For example:

```
HTSetPenName("HistTrendTag", 1, "HistPrv1.Boiler1");
```

Where a 1 specifies the trend pen that plots the specified remote history provider tag.

The run-time **Historical Trend Setup** dialog box and .Pen dotfield are not supported for remote history providers.

Configure distributed historical logging

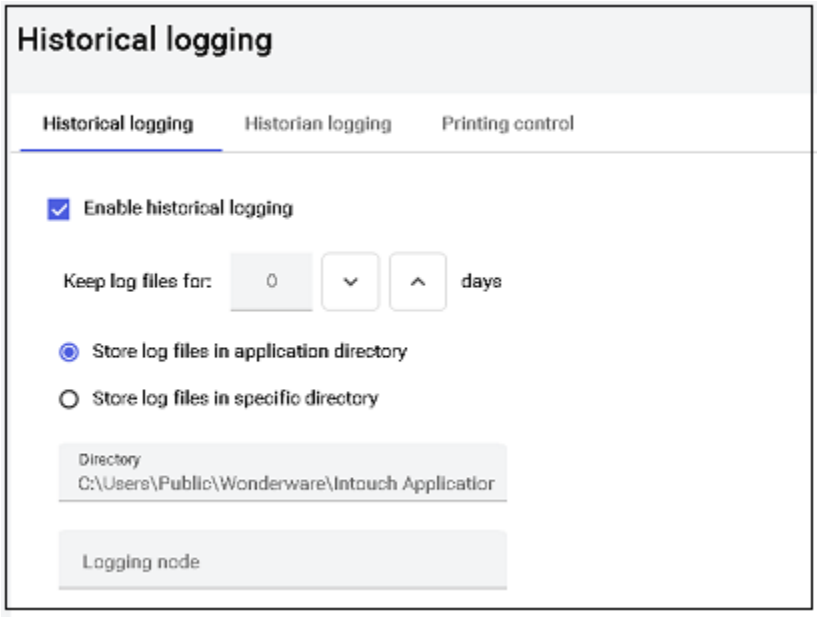
Only one InTouch node can log to the history file. However, multiple InTouch nodes can view the file.

Note: Historical logging cannot be configured for an InTouchView application. For more information about the limitations of InTouchView applications, see InTouchView applications.

Configure distributed historical logging

1. On the **File** menu, select **Configure**, and then select **Historical Logging**.

   The **Historical Logging** configuration screen appears.



2. Select the **Enable historical logging** check box to turn on global tag logging.

3. Select **Store Log Files in Specific Directory**, and then in the input box, type the path of the location where the log files are stored.

   You must type a valid Universal Naming Convention (UNC) path. For example, `\\Node\Share\Path`

   If you are using NAD, make sure the path points to a folder other than the application folder.
4. In the **Name of Logging Node** box, type the name of the node that will be logging to the history log file.

   This setting only allows the node named here to log to the file.
5. Select **OK**.

Note: When an application with the **Enable Historical Logging** option selected is distributed to a WindowViewer node, that node checks this option to determine if it should log or not. If **Enable Historical Logging** is selected, the possible settings are: **Field equals name of Node - Logging enabled Field does not equal name of Node - Logging disabled.**

Considerations for special networks

If you are working on a slow network and the InTouch HMI takes a long time to start or save information, modify the `win.ini` settings on the NAD client:

```
ViewNadClearNADCopyDirectory=0
ViewNADCopyApplicationOnStartup=1
ViewNADOnApplicationChanged=3 ( or 4)
ViewNADThreadPriority=2
```

For the ViewNADOnApplicationChanged parameter, a setting of 3 corresponds to the **Load changes into** WindowViewer option on the **Node Properties** dialog box in the InTouch Application Manager. A setting of 4 corresponds to the **Prompt user to load changes into** WindowViewer option. These settings allow the application to continue to run while NAD downloads take place in parallel, on a separate execution thread.

When NAD performs an update to an application, it copies only the changed files from the master. NAD does not copy the SmartSymbol design-time dictionary files for run-time language switching.

Configure an InTouch application for NAD

Network Application Development or NAD is an architecture that combines the best of the client-based and server-based architectures. NAD provides automatic notification of application changes and can automatically distribute updated applications to View nodes

When configuring an application for NAD, you must specify the folder that you want WindowViewer to copy the master application to.

- If this is the development node, you can type a local folder path, such as c:\InTouch\NAD. You can also type a networked remote UNC path, such as \\node\share\path. This is convenient for file server-based networks where most file storage is kept in a central location.
- If this is a client node (run-time only), you typically use a local folder path.

We recommend that you use a local folder whenever possible to prevent network delays and failures from affecting the operation of WindowViewer.

Caution: Do not use a root folder or a UNC pathname that points to a root folder. The View node recursively deletes all files and subfolders in the specified destination application folder before copying the master application directory. Therefore, never use the path of the master application folder or a UNC to the master

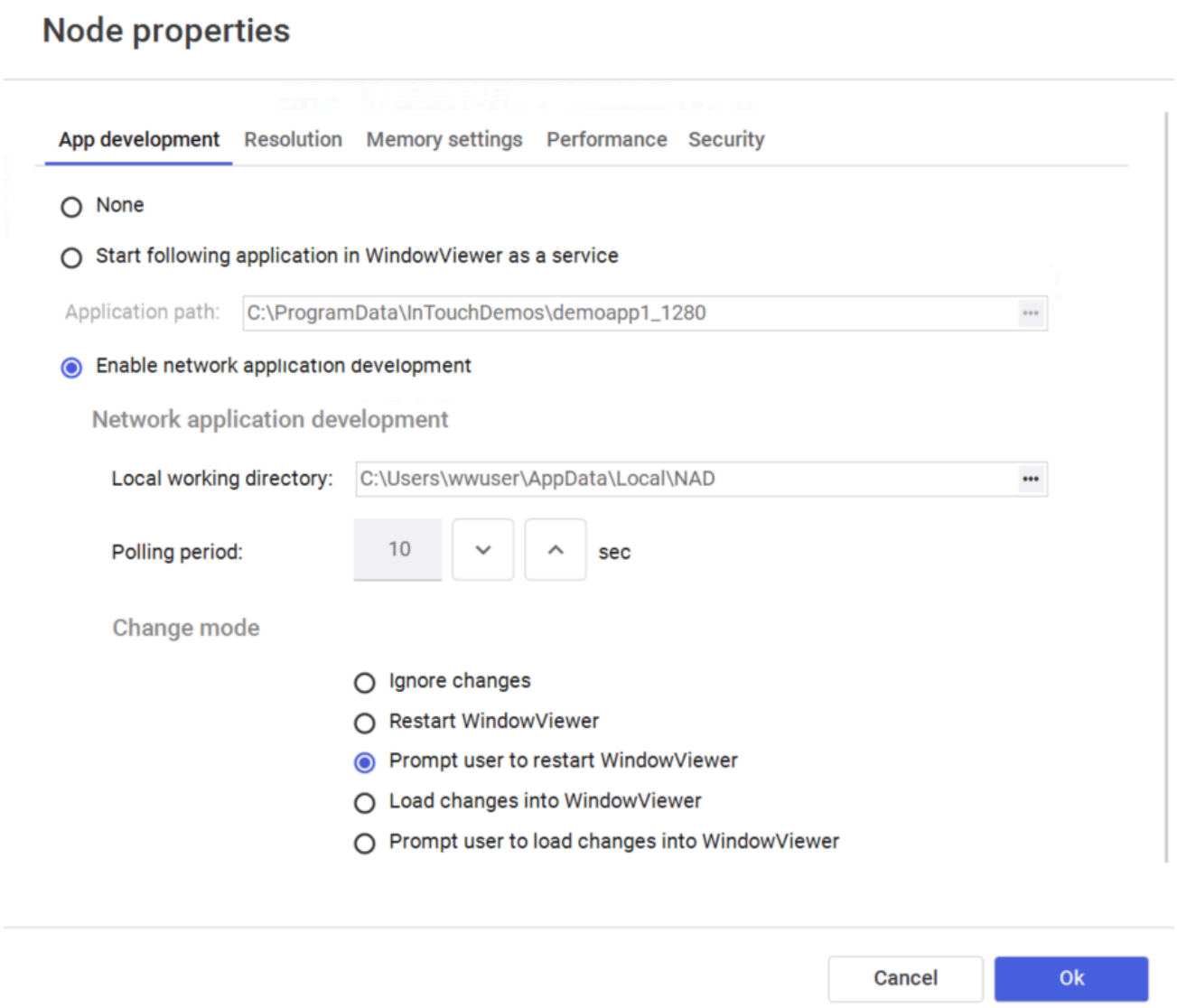
application folder.

If you do not specify a folder, WindowViewer automatically creates a local subfolder named NAD in the folder from which WindowViewer is launched. The NAD folder should be considered a temporary folder and no other files should be saved to it except those copied by NAD itself.

Configure an application for NAD

1. Start Application Manager.
2. On the **Tools** tab, select **Node Properties**.

   The **Node Properties** screen appears.



3. Select the **Enable Network Application Development** radio button.
4. In the **Local working directory** box, type the path to the folder that you want WindowViewer to copy the master application.
5. In the **Polling period (sec)** box, type the interval, in seconds, at which the View node checks the development node for updates.

- Be careful that you do not set this value too small. If WindowViewer checks for master application changes too often, it can interfere with servicing the running application.

6. In the **Change Mode** area, select the option that determines the action WindowViewer takes when the master application changes.
   - Select **Ignore changes** to have the WindowViewer node ignore any changes made on the development node.
   - Select **Restart WindowViewer** to have the WindowViewer node copy over the updated master application (if configured to do so) and then restart itself.
   - Select **Prompt user to Restart WindowViewer** to show the operator a message that the application has changed. The operator can either restart WindowViewer with the application updates or continue using the current application.
   - Select **Load Changes into WindowViewer** to dynamically load in WindowViewer the changes made in the development node. This may affect performance for large updates.

     Note: It is recommended that you use the **Load Changes into** WindowViewer option only if the application changes are minor and few in number. Examples of minor changes include changes made within an existing window, resizing of graphic toolbar elements, adding new graphic toolbar elements, and reference substitutions. When making changes that require that WindowViewer be restarted, such as adding new tags, adding new windows, or changing the configuration—or if in doubt—use one of the Restart options instead.
   - Select **Prompt user to load changes into WindowViewer** to show the operator a message that the application has changed. The message prompts the operator to load the changes.
7. Select **OK**.

Perform an automatic NAD update

You can start an automatic NAD update during application development.

When you run the **Notify Clients** command, a flag is set to notify all remote View nodes that the master application has changed. Clients can automatically start an update process based on the **Change Mode** option defined for each node.

The first time a standalone application (with embedded Industrial graphics) is opened on a View node, graphics may not appear and errors are logged in the Logger. To avoid this, run the **Notify Clients** command from the master node and the Industrial graphics will be loaded on the View node based on the **Change Mode** option.

Perform an automatic update

1. Open the application in WindowMaker.
2. On the **Properties** menu, in the **Clients** group:
   a. Select **Notify now** to notify clients immediately.
   b. Select **Notify on-close**, to be reminded to notify NAD clients, when WindowMaker is closed.

Note: If the **Notify on close** option is selected, every time WindowMaker is closed it will verify if there are any changes from the last notification. If there are any changes, a dialog box with the prompt ‘Do you want to notify the NAD clients?’ will appear. Select **Yes** to notify the clients, select **No** to ignore the changes.

Perform a manual NAD update

You can write scripts that allow operators to manually start a NAD update on the View nodes in which they work.

To manually update an application with NAD, you must set the **Change Mode** option to **Ignore Changes** in the **Node Properties** dialog box. For more information, see Configure an InTouch application for NAD.

Use the following system tags and functions in your script to perform a manual NAD update:

- $ApplicationChanged System Tag
- $ApplicationVersion System Tag
- RestartWindowViewer() Function
- ReloadWindowViewer() Function

$ApplicationChanged system tag

Signals that the master application has changed in a Network Application Development (NAD) architecture.

Category

application

Usage

```
$ApplicationChanged
```

Remarks

This system tag changes to 1 every time the update signal is generated by selecting **Notify Clients** on the WindowMaker **Special** menu. $ApplicationChanged is reset to 0 when the application is updated. This tag can be used to generate a message that informs the operator that the master application has changed.

You can also use the $ApplicationChanged system tag in a data change script to build a node update notification script. This script can launch your own dialog boxes or stop running processes. Then, you could use the ReloadWindowViewer() function to start the update process.

Data Type

Discrete (read only)

Example

Using the following statement in the tagname box of a data change script causes the body of the script to run. The script body could show a window informing the user to restart WindowViewer for the change to take effect.

```
$ApplicationChanged
```

See Also

$ApplicationVersion

$ApplicationVersion system tag

Contains the current version number of the application. This number changes with every change that can be saved or undone.

Category

application

Usage

```
$ApplicationVersion
```

Remarks

The value associated with the $ApplicationVersion system tag is set to the current version of the InTouch application. The version changes with every change to the application that can be saved or undone. This tag can be used to generate a message that informs the operator that the master application has changed.

Data Type

Real (read only)

Example

If used in an analog display link, this system tag shows the current version of the application that is running within WindowViewer.

```
$ApplicationVersion
```

See Also

$ApplicationChanged

RestartWindowViewer() function

Shuts down WindowViewer, copies the updated master application (if configured to do so), and then restarts WindowViewer.

Category

system

Syntax

```
RestartWindowViewer();
```

Remarks

This function is used to update an application when the automatic update Network Application Development (NAD) functions are not used.

Use the $ApplicationChanged system tag to determine when a NAD update has occurred.

You use the **Notify Clients** command to initiate a NAD update. However, the operator may want to delay the update until a later time. You can use this function with a button action script so that the operator can restart WindowViewer when it is convenient.

You could instead use the ReloadWindowViewer() function, which updates the View node without shutting down WindowViewer.

See Also

$ApplicationChanged, ReloadWindowViewer()

ReloadWindowViewer() function

Dynamically updates WindowViewer with the updated master NAD application without any interruption in service.

Category

system

Syntax

```
ReloadWindowViewer();
```

Allows the user control over reloading WindowViewer.

Remarks

Use this function to update an application when the automatic update Network Application Development (NAD) functions are not used.

Use the $ApplicationChanged system tag to determine when a NAD update has occurred.

You use the **Notify Clients** command to initiate a NAD update. However, the operator may want to delay the update until a later time. You can use this function with a button action script so that the operator can reload the application in WindowViewer when it is convenient.

See Also

$ApplicationChanged

Application editing locks

To prevent multiple developers from trying to edit an application, WindowMaker locks an application during the edit session. If you try to open a locked application, an error message is shown. The name of the node editing the application is included in the message.

If WindowMaker is abnormally shut down with an application loaded, the appedit.lok file may not be deleted. You can manually remove the lock by deleting the appedit.lok file from the application directory.

Changes to an application during a NAD update

When the WindowViewer node updates an application, it makes every attempt to retain the attributes (read-only, system, hidden, and so on) of the master application during the copy process.

WindowViewer also copies all files and subfolders of the master application, except for these files: *.WVW, *.DAT, *.LGH, *.IDX, *.LOG, *.LOK, *.FSM, *.STG, *.DBK, *.CBK, *.HBK, *.KBK, *.LBK, *.NBK, *.OBK, *.TBK, *.WBK, *.XBK, *.$$$, RETENTIV.X, RETENTIV.D, RETENTIV.A, RETENTIV.S, RETENTIV.H, RETENTIV.T, SSD_, WM.INI, DB.INI, LINKDEFS.INI, TBOX.INI, GROUP.DEF, and ITOCX.CFG.

Note: WindowViewer recursively deletes all files and sub folders in the destination application folder except those required for run-time language switching. This folder should be considered a temporary folder. No other files should be placed in it.

The NAD client starts an update by creating a local list of files and sub-directories that appear in the client application directory. As it looks for updates in the list of master files, the NAD client removes the corresponding client file for each master file from the local list. The remaining entries in the local list are obsolete files and sub-directories that should be deleted from the application.

All downloaded files are copied to a temporary sub-directory called NAD_Temp. Files are only copied from NAD_Temp to the application directory if all of the new and updated files are copied successfully within the re-try limits. If the NAD client has to abandon an update, the running application is not corrupted by the partial introduction of new or updated files.

If contact with the NAD master fails after all new and updated files have been downloaded, the update can still be completed by copying the updates from NAD_Temp and deleting the obsolete files. This ensures that files are not erased simply because a lost connection makes it impossible to confirm their existence on the master application.

NAD can detect whether additional changes have been made to the master application during application download. If such a situation arises, NAD abandons the download of the application. If you run the **Notify Clients** command after the latest update, NAD automatically begins downloading the latest application files at the next polling period. Otherwise, it waits until the next **Notify Clients** command issued before an application download takes place.

Scale the application resolution at run time

You can use Dynamic Resolution Conversion (DRC) so that the distributed applications you create can run on

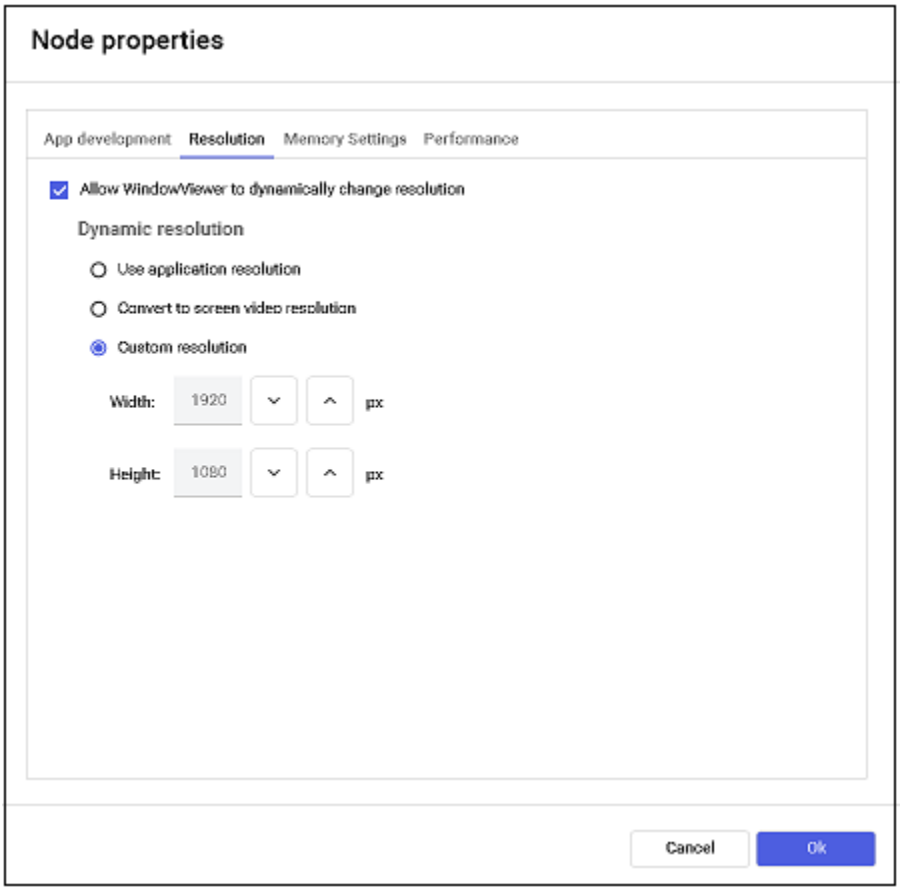
different screen resolutions.

Each View node can scale the application appropriately, including scaling to a custom resolution. This scaling takes place while WindowViewer compiles the application and does not require WindowMaker. Because each View node can use a different DRC setting, each View node must have its own settings configured.

Caution: If you do not use DRC to scale the application, WindowViewer only runs the application if the node's screen resolution is identical to the screen resolution of the application development node. If the resolutions are different, WindowViewer prompts the operator to run WindowMaker to convert the application to the node's resolution. Use caution when doing this if you have set up a UNC path to the master application directory, as this will only modify the original application.

Configure an application for DRC

1. Start Application Manager.
2. On the **Tools** menu, select **Node Properties**. The Node Properties dialog box appears.
3. Select the **Resolution** tab.



4. Select the **Allow WindowViewer to dynamically change resolution** check box if you want WindowViewer to locally scale the master application.
5. In the **Dynamic Resolution** area, select one of the following:
   - Select **Use application resolution** if you want WindowViewer to run the application at the resolution it was developed for and ignore the node's resolution. For example, if the application was developed at 800x600 and the node's resolution is 1024x768, WindowViewer does not dynamically scale the application. Instead, the application resolution remains at 800x600.

- Select **Convert to screen video resolution** if you want WindowViewer to run the application at the node's resolution and ignore the resolution the application was developed at. For example, if the node is running at 800x600 and the application was developed at 1280x1024, WindowViewer dynamically scales the application to fit the node's 800x600 resolution.
  - If the target resolution is different from the screen resolution when the application was created, then WindowViewer will scale to the current screen resolution from the original application resolution instead. The original application resolution is the screen resolution when the application was created regardless of the target resolution settings. For example, if the application was developed at 1920x1080 with a target resolution of 1280x1024 and the view node is running the application at resolution of 800x600, WindowViewer will dynamically scale the application to use the original application resolution of 1920x1080. For more information, see Original application resolution.
- Select **Custom resolution** if you want WindowViewer to run the application at a specific resolution you specify in the **Width (X)** and **Height (Y)** (must be integer values) boxes. The application's resolution and the node's resolution are both ignored. For example, if **Width (X)** and **Height (Y)** are set to 512 and 384, respectively, the application is dynamically scaled to fit in a 512x384-pixel area on the node's screen.
  - If the target resolution is different from the screen resolution when the application was created, then WindowViewer will scale to the current screen resolution from the original application resolution instead. The original application resolution is the screen resolution when the application was created regardless of the target resolution settings.

6. Select **OK**.

Lock the application resolution

You can configure the WindowMaker properties to lock the size of InTouch application windows. This allows you to convert applications to a different resolution without scaling the windows and graphics.

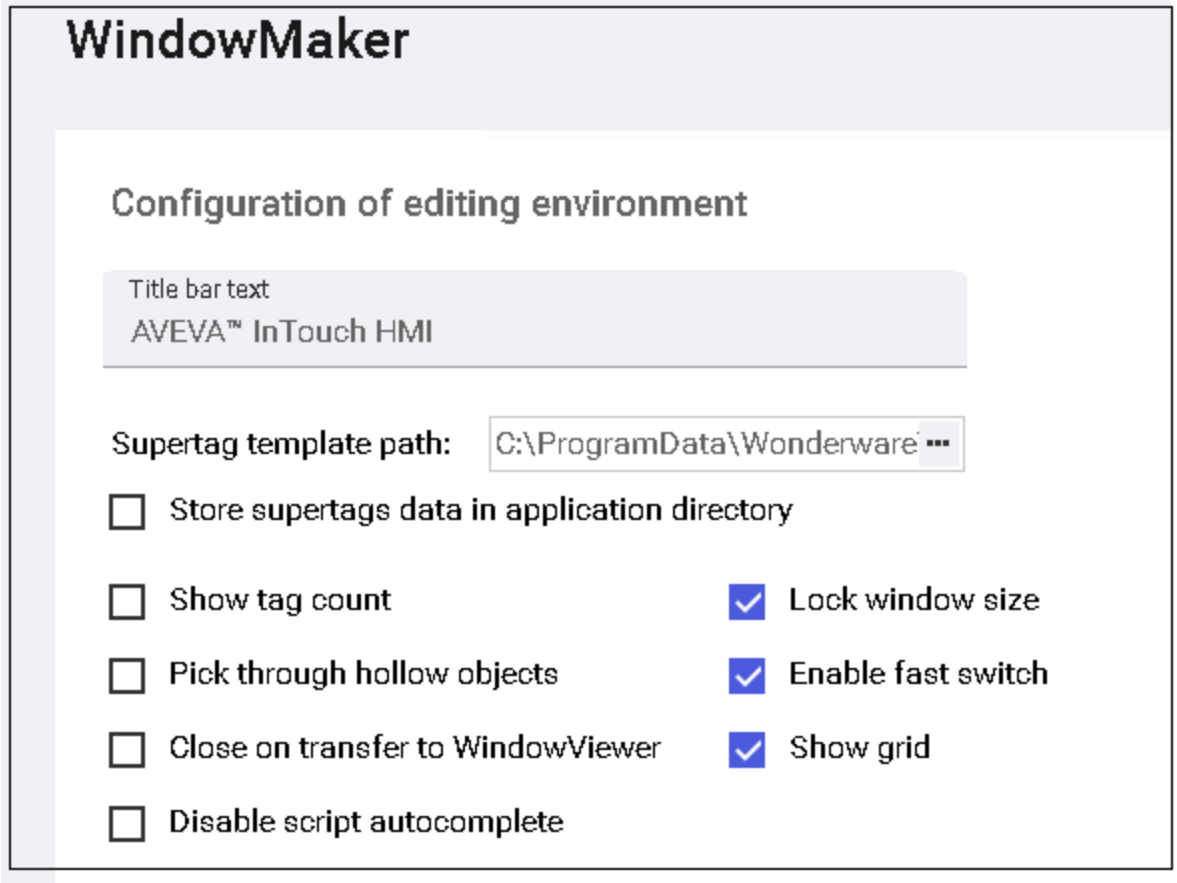
If you select this option, the next time you open an application in a computer with a different resolution, the system prompts you to specify whether you want to convert the application to the new resolution without scaling the windows and graphic.

You can lock the application resolution from inside WindowMaker or from the Application Manager.

Lock the application resolution from WindowMaker

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **WindowMaker**.

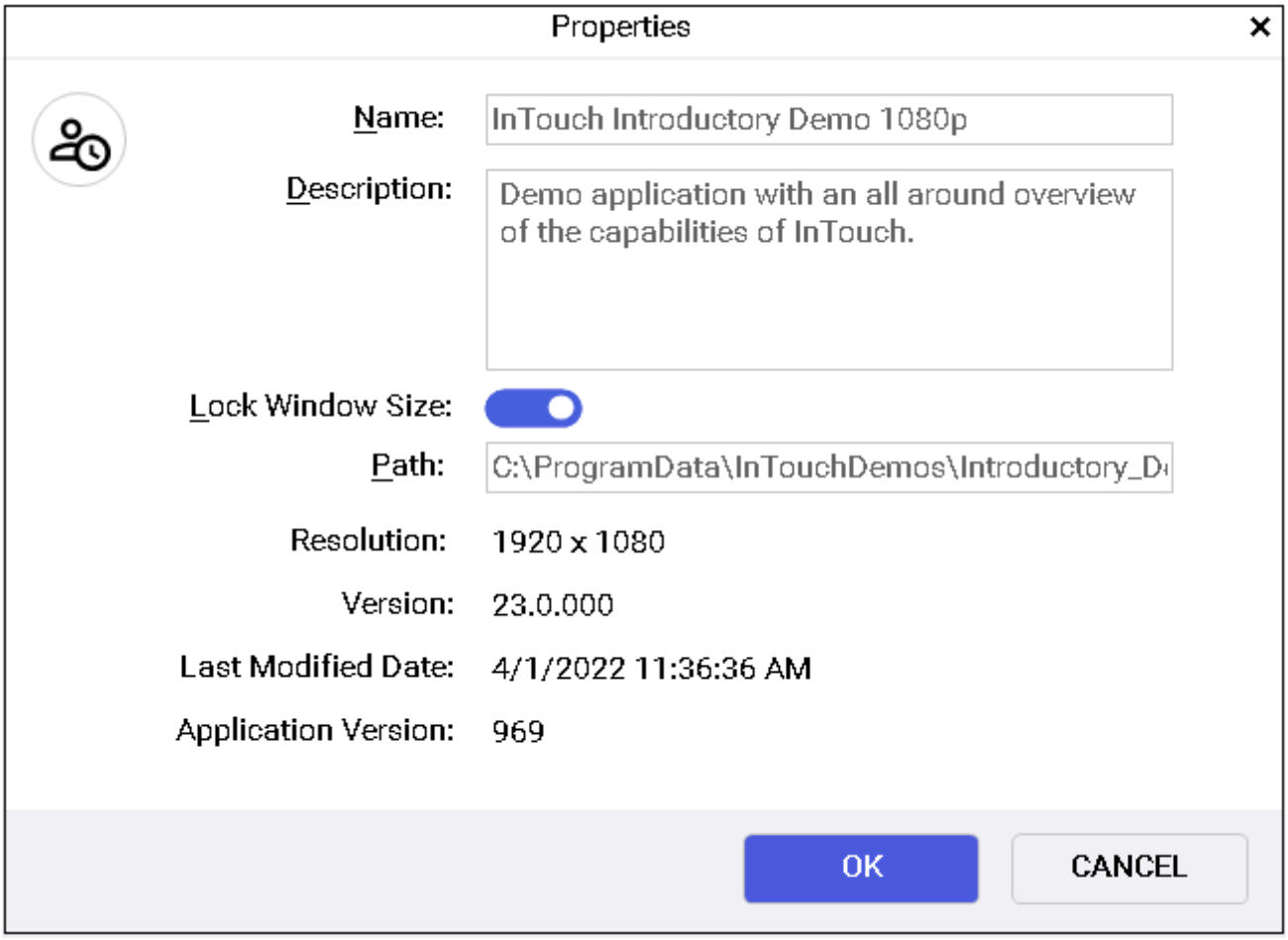
   The **WindowMaker configuration** screen appears.



3. Select the **Lock Window Size** check box. By default, the check box is not selected.
4. Select **Save**.

Lock the application resolution from Application Manager

1. Open Application Manager.
2. Select the application you want to configure.
3. Select **File** on the menu bar, then select **Properties**.
   The **Properties** dialog box appears.
4. Select the **Lock Window Size** switch. By default, the check box is not selected.



5. Select **OK**.

Publish applications to remote nodes

Using Application Publisher, you can create a compressed, self-extracting package file that contains all relevant files and setup procedures to install an InTouch application on another computer. You use Application Publisher to publish standalone InTouch applications. You publish managed InTouch applications using the System Platform IDE.

You have two options to publish applications:

- **Run-time only.** A run-time only package includes the files needed to run the application, but not to edit the application.
- **Design-time and run-time.** A design-time and run-time package includes all files needed to edit and run the application. Some run-time files, such as compiled *.wvw files, are excluded because they can be re-created from the design-time files.

You can post published applications to a web server where they can be downloaded and installed. The following package information is shown for posted applications:

- Package description
- Publisher name

- Published file name (executable)
- Application resolution

For example:

Description	Dairy Processing Application
Publisher	Navin Johnson
File Name	Dairy.exe / Video Resolution…(1024x768)

Description	Dairy Processing Application
Publisher	Navin Johnson
File Name	Dairy_2.exe / Video Resolution…(800x600)

Contents of a published file

The following table lists the included folders, files, and excluded files for all published stand-alone InTouch applications.

Included Folders	Included Files	Excluded Files
Main application folder	All	Backup files. These files have the .?bk file name extension.
	Files with these extensions: .win, .dat, .lgh, .idx, .log, .fsm, .stg, .$$$	Subfolders not in the Special Directories list
	retentiv.x retentiv.d retentiv.a retentiv..s (two dots) retentiv.h wm.ini db.ini linkdefs.ini tbox.ini group.def itocx.cfg	The appedit.lok file, which indicates that the application is open in WindowMaker.

Included Folders	Included Files	Excluded Files
	Any files with names of the form SSD_*.xml.	Compiled window files with the file name extension .wvw.
Dictionary subfolders for run-time language switching	All files with the .xml extension.	
Symbol subfolders	All files and subfolders.	
	wiz.ini file, if there are wizards installed.	
	Copy of the wizard executable.	
	.dll files, .wdo files .wdf files	

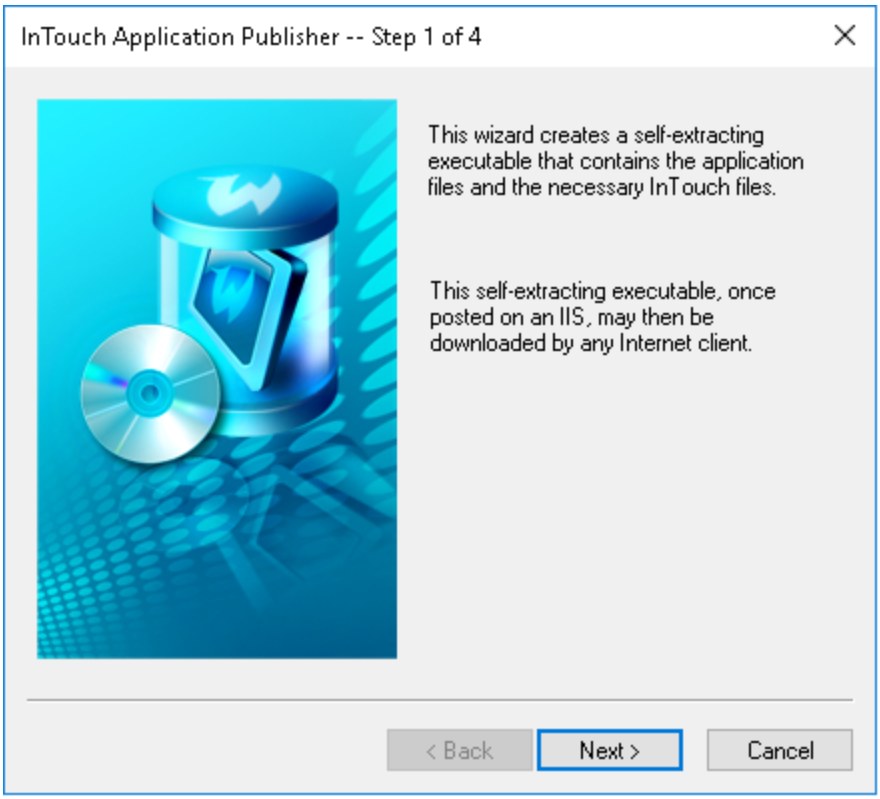
For a run-time only application, all files with a file names of SSD_*.xml are excluded.

Publish a standalone InTouch application

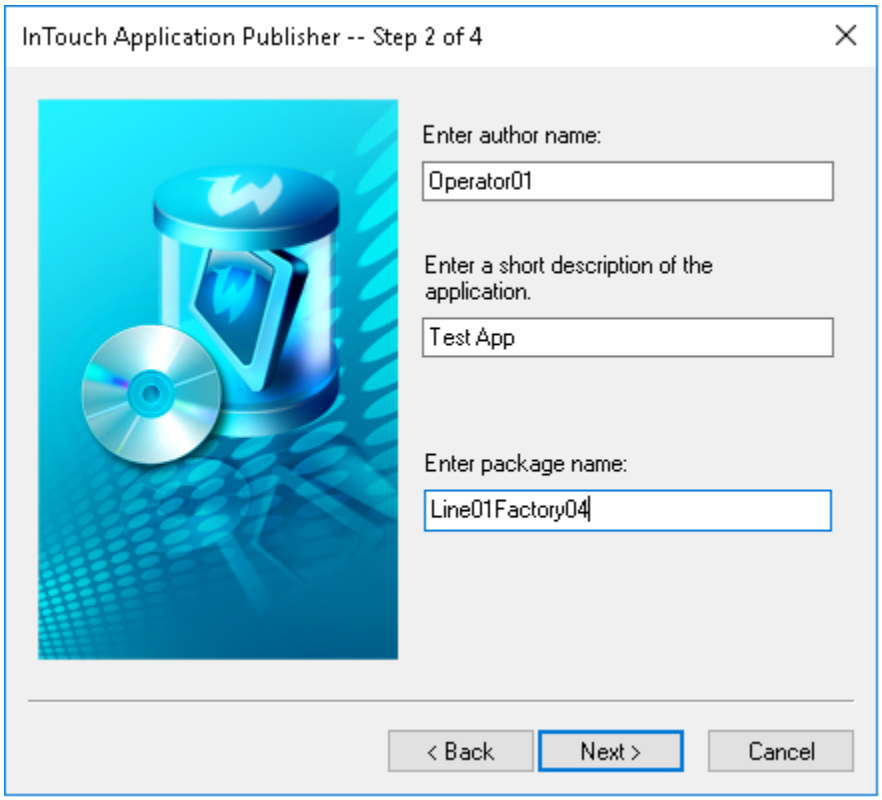
Use the Application Publisher to publish a standalone InTouch application. If you want the published application to run at a specific screen resolution, set the original application to that resolution before you publish it. To publish a managed InTouch application, use the System Platform IDE.

Publish a standalone InTouch application

1. Start the Application Publisher.
   a. Open WindowMaker.
   b. Expand the **Tools** pane, and expand **Applications**.
   c. Double-click **Application Publisher**.
   The InTouch **Application Publisher – Step 1 of 4** dialog box appears.



2. Select **Next**. The InTouch **Application Publisher – Step 2 of 4** dialog box appears.

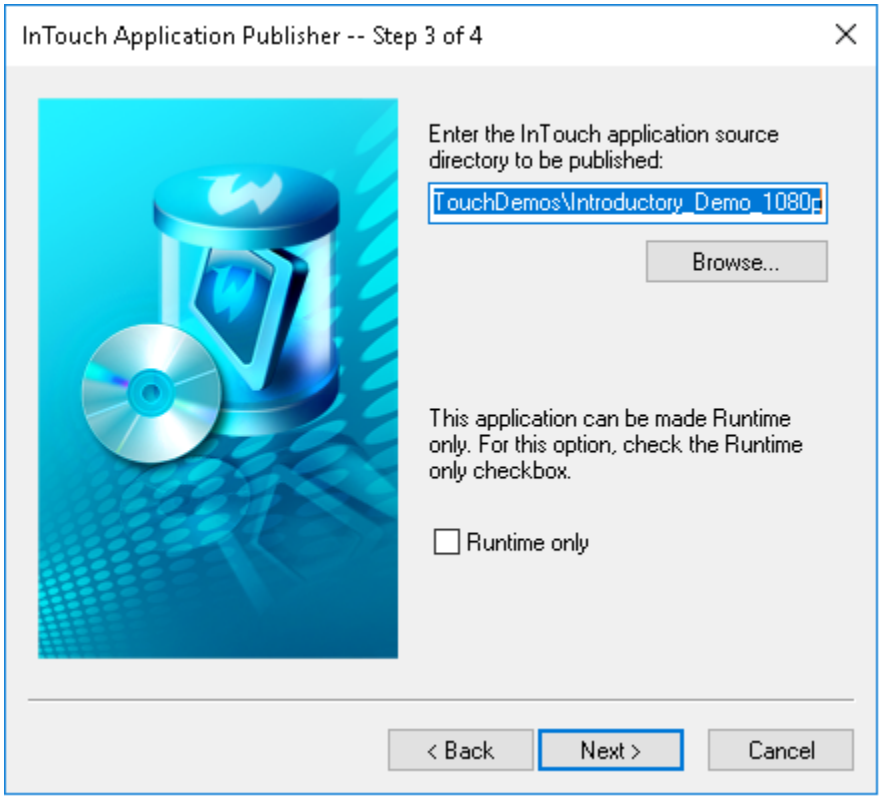


3. Configure the package details.
   - In the **Enter author name** box, type the name of the person to contact regarding the application. The

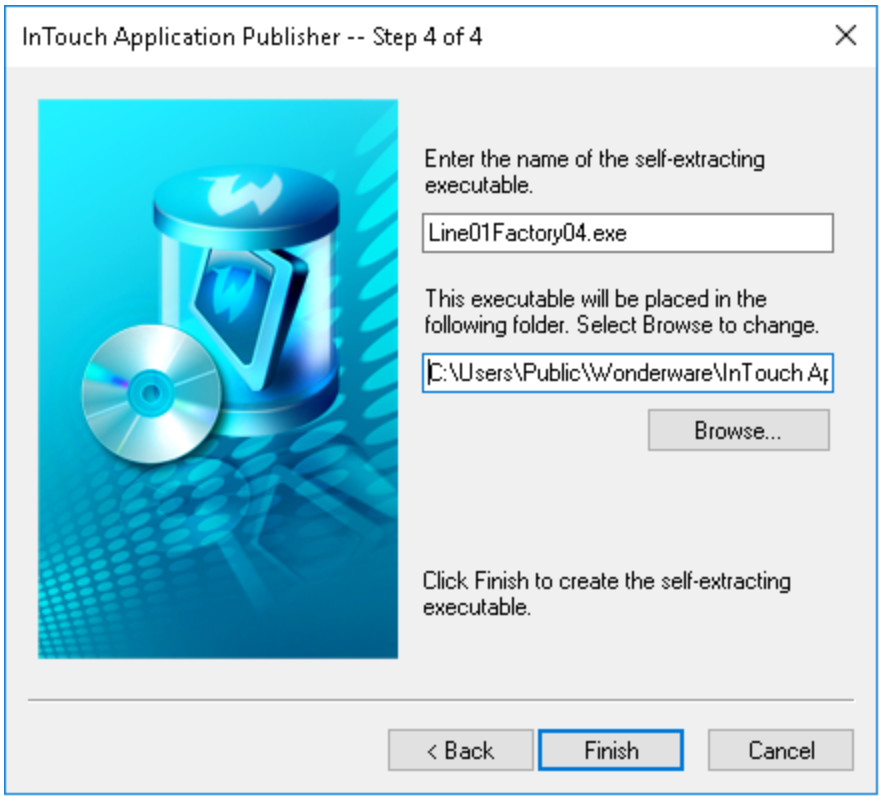
name limit is 256 characters.

- In the **Description** box, type a description of the application. The limit is 256 characters.
- In the **Package Name** box, type a unique name for the published application package. The limit is 32 characters. If you use the name of an existing package, the existing package is overwritten.

4. Select **Next**. The InTouch **Application Publisher – Step 3 of 4** dialog box appears.



5. Configure the publishing details.
   - In the box, type the path to the InTouch application folder. The default path is the WindowMaker application folder.
   - Select the **Runtime only** check box to exclude the development WindowMaker files in the published file.
6. Select **Next**. The InTouch **Application Publisher – Step 4 of 4** dialog box appears.



7. Configure the details for the executable application package.
   - In the first box, verify the executable name in the first box is correct. By default, the executable name is the same as the package name.
   - In the second box, type the path to the folder in which to save the executable file, or select **Browse** to select a different folder. By default, the executable is saved in your current temporary folder.
8. Select **Finish**.

Publish managed InTouch applications

You can publish a managed InTouch application. The published InTouch application is no longer associated with the InTouchViewApp template.

The published application cannot be edited within the IDE or imported into another InTouchViewApp template. In other words, you cannot manage it with the IDE or republish it.

The published InTouch application can still communicate with the Galaxy through any embedded Industrial graphic. You can, for example, write data back to the Galaxy or visualize Galaxy data.

You can edit the Industrial graphic using basic InTouch operations such as copying, cutting, pasting, duplicating, moving, resizing, flipping, rotating, and configuring with InTouch animation links.

However, the Industrial graphics cannot be modified, nor can new Industrial graphics be embedded into the InTouch application. These operations are only allowed with managed InTouch applications.

You can do this in environments that do not support the processing requirements of ArchestrA. For example, in remote plant operations or in small networks.

Publish a managed InTouch application

You can publish a managed InTouch application from the InTouchViewApp object that is associated with it.

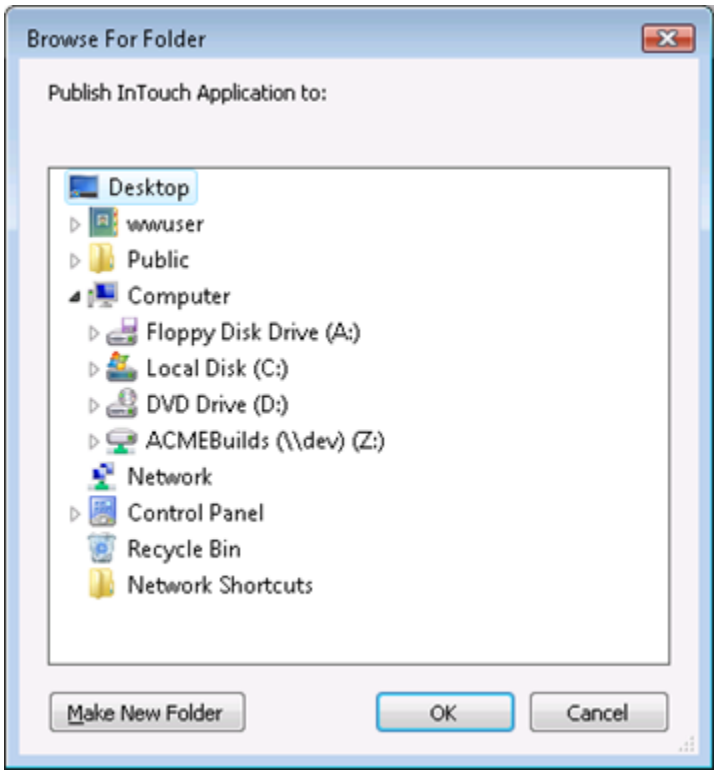
The export consists of a folder containing information about the object and the managed InTouch application.

This is different than exporting the InTouchViewApp object itself. For more information, see Import and export an InTouchViewApp object.

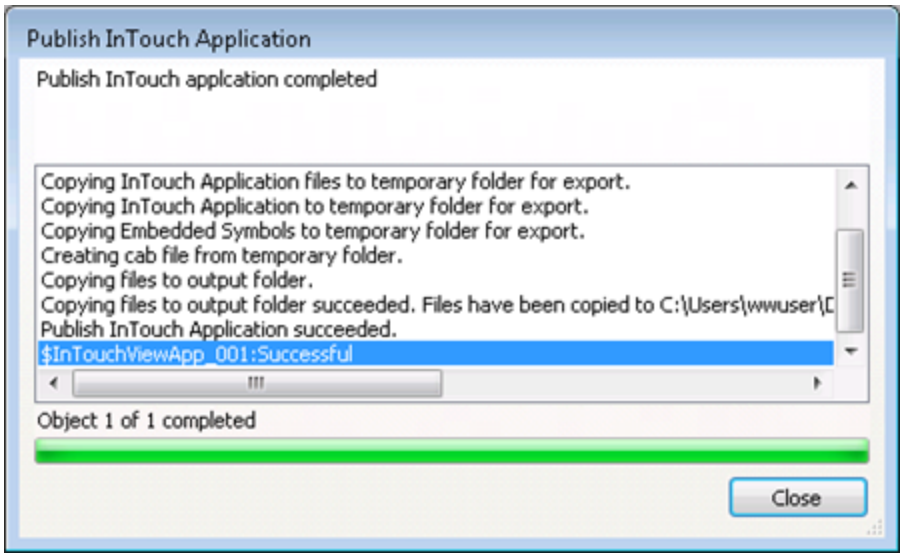
The published InTouch application cannot be reimported into an InTouchViewApp object. A published InTouch application cannot be edited.

Publish a managed InTouch application

1. Open the System Platform IDE.
2. Locate the InTouchViewApp object that contains the managed InTouch application you want to publish.
3. Right-click the object, and then select **Publish InTouch Application**. The **Browse For Folder** dialog box appears.



4. Specify the folder to publish to the InTouch application to. Do any of the following:
   - Browse to an existing folder.
   - Select **Make New Folder** to create a new folder or folder structure.
5. Select **OK**. The **Publish InTouch Application** progress dialog box appears.



6. When publishing is complete, select **Close**. A directory containing the new published InTouch application is created in the selected folder. You can now copy it to any run-time node.

Publish applications to Insight

Using the Insight Publisher you can publish an application to the Insight website. You can use the Application Manager or WindowMaker.

Publish applications to Insight

1. Open Application Manager.
2. On the **Tools** menu, in the **Tools** group, select **Insight Manager**.

   The **Insight Publisher** window appears.

InTouch WindowMaker, you can use the **AVEVA Insight Publisher** from the **Tools** pane under **Applications**.

You can select one of the following options and proceed with the onscreen instructions.

- **Publish** - Create a new Insight data source from an existing InTouch application.
- **Import** - Import an Excel spreadsheet listing items from an OPC, MQTT, or OI Server.
- **Authorize** - Create a data source.

For more information, refer to the Historian Documentation.

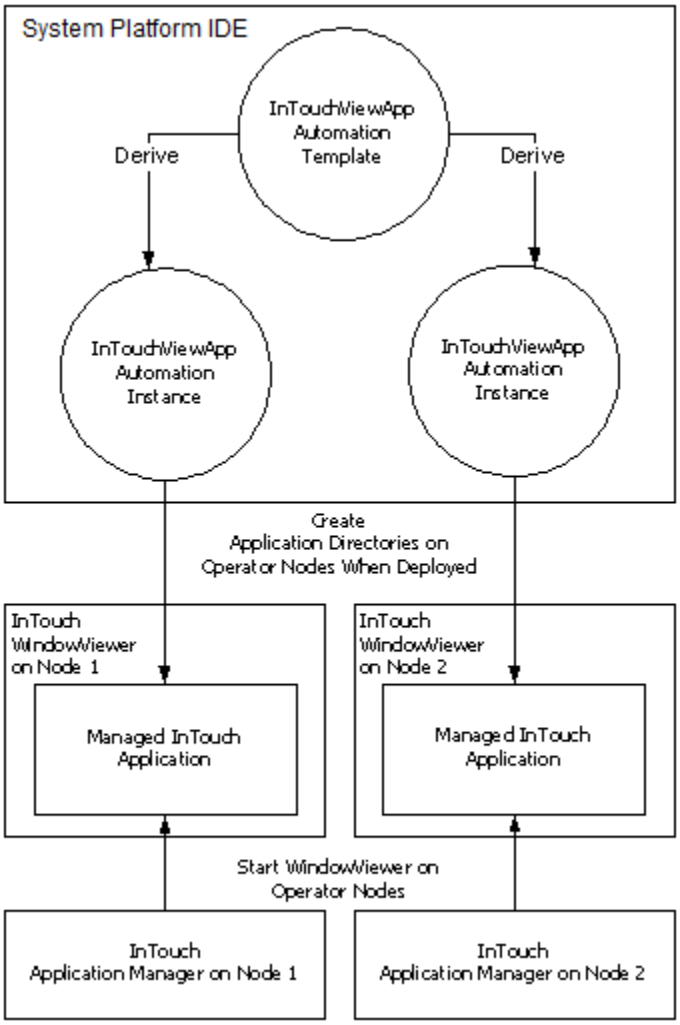
Note: You will need an Insight Account to publish the application.

Use managed InTouch applications at run time

You can run managed InTouch applications on remote nodes by deploying InTouchViewApp instances to those nodes.

You can also deploy changes of the InTouch application and contained Industrial Graphics to these nodes and select whether to accept or decline the changes for each node.

The following graphic shows how managed InTouch applications are deployed to run-time nodes.



Deploy a managed InTouch application

You can deploy a managed InTouch application from the System Platform IDE to the local node or a remote node. After you deploy the application, you can run it in WindowViewer on the remote nodes.

Note: WindowViewer can run only one application at a time. If a platform is deployed on a local node, the configured styles of the Galaxy will take precedence over any configured styles in any other standalone or managed applications.

Deploy a InTouchViewApp object for the first time

The first time you deploy an InTouchViewApp object, the associated InTouch application is copied to the node of the platform that hosts the object. This is called the operator node.

Deploy a managed InTouch application

1. Open the System Platform IDE.

2. Select the instance of the InTouchViewApp for which you want to deploy the managed InTouch application.
3. On the **Object** menu, select **Deploy**. The **Deploy** dialog box appears.
4. Select **OK**. The complete InTouch application is copied to the operator node.

Deploy changes to a managed InTouch application

Note: The **Override WindowViewer default configurations** option has not been tested on a non-English language Operating System. Web Client does not support this option.

You can change a managed InTouch application by:

- Changing the references, content, or size of an Industrial Graphic that is used in a managed InTouch application.
- Changing the managed InTouch application itself by opening WindowMaker from the InTouchViewApp template.

In both cases, when you save the changes, the changes are propagated from the updated template to the derived instances. The changes are identified by the Pending Changes icon.

The changes are not immediately reflected in a running WindowViewer session. The operator of each node can select to accept or decline the changes. For more information, see Accept new application versions at the operator node.

Deploy changes to a managed InTouch application

1. Open the System Platform IDE.
2. Select the instance of the InTouchViewApp for which you want to deploy the changes of a managed InTouch application.
3. On the **Object** menu, select **Deploy**. The **Deploy** dialog box appears.
4. Select **OK**. The changes are copied to the operator node.

Start a managed InTouch application

From the operator node, you can start Application Manager and select the managed InTouch application you want to run.

You can also set the resolution in Application Manager to run the managed InTouch application in different resolutions.

Start the managed InTouch application

1. On the node the InTouchViewApp object is deployed to, start the InTouch Application Manager.
2. In the application list, select the managed InTouch application you want to run in WindowViewer.
3. Select the WindowViewer icon. The application starts in WindowViewer after a short while.

Set the dynamic resolution conversion settings

1. Open the InTouch Application Manager.
2. On the **Tools** menu, select **Node Properties**. The **Node Properties** dialog box appears.

3. Select the **Resolution** tab.
4. Select the **Allow** WindowViewer **to dynamically change resolution** check box.

   If **Allow** WindowViewer **to dynamically change resolution** is unchecked, the managed application runs with the resolution in which it was developed.
5. Configure how you want to run the application. Do any of the following:
   - Select **Use application resolution** to run the managed application at the same resolution as it was developed in.
   - Select **Convert to screen video resolution** to convert the managed application so that it can run with the screen resolution.
   - Select **Custom resolution** to convert the managed application to a specified resolution.
6. Select **OK**.

Control the WindowViewer restart wait period when deploying managed applications

You can control the delay that occurs before WindowViewer is restarted when managed applications that require a restart are deployed. Delays are implemented to ensure the proper operation of the alarm subsystem.

The WindowViewer restart delay is controlled by the following parameters in the win.ini file, which is located in the local C:\Windows folder. All parameter values are in seconds.

- `ViewManagedRestartWaitPeriod=0`

  This parameter controls the WindowViewer restart wait period. The default is 0, or no wait period.
- `ViewNADShutdownWaitPeriod=30`

  For applications managed using Network Application Development (NAD), this parameter controls the delay before NAD is shutdown. The default is 30 seconds.
- `ViewNADRestartWaitPeriod=90`

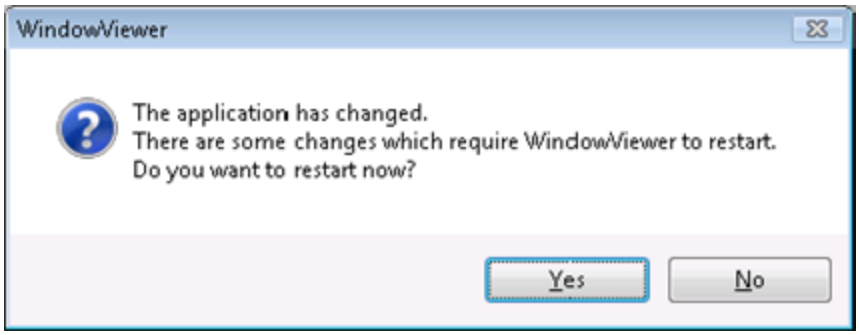
  For applications managed using NAD, this parameter controls the delay before NAD is restarted. The default is 90 seconds.

Accept new application versions at the operator node

If the managed InTouch application is changed and its associated InTouchViewApp instance is deployed, you can select to accept or decline the changes at the run-time node.

A message appears prompting you if you want to accept the changes to the managed InTouch application:

- When you start WindowViewer from the InTouch Application Manager.
- While WindowViewer is running.



Depending on the nature of the change, you may be prompted to restart the WindowViewer application, or just reload it. You can also set the behavior of WindowViewer for application changes, such as:

- How WindowViewer accepts or rejects these changes.
- How frequently WindowViewer should remind you to reload or restart WindowViewer when changes are pending.

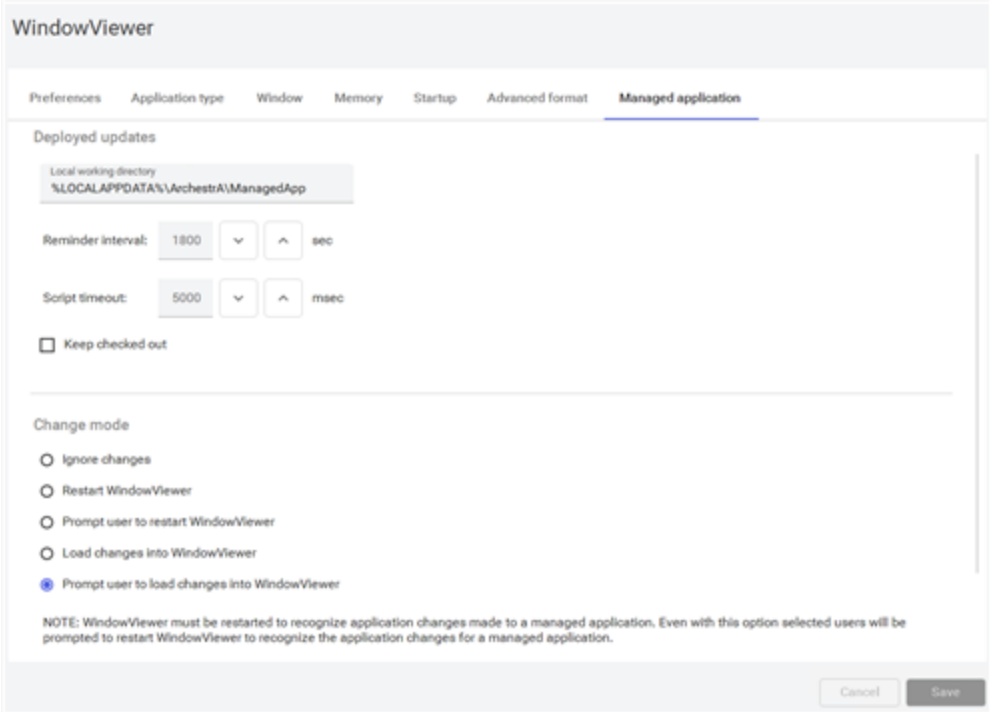
Accept new application versions at the operator node

- Select **Yes**. The changes to the managed InTouch application are copied to the operator node and WindowViewer restarts or reloads.

Set the behavior of WindowViewer for application changes

1. Open the managed InTouch application in WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **WindowViewer**.

   The **WindowViewer configuration** screen appears.
3. Select the **Managed Application** tab.



4. In the **Change Mode** area, configure how WindowViewer responds when changes are deployed. Do any of the following:
   - Select **Ignore Changes** to have WindowViewer ignore any deployed changes. You can manually configure the RestartWindowViewer() and ReloadWindowViewer() script functions to accept the changes depending on the $ApplicationChanged system tag.
   - Select **Restart WindowViewer** to have WindowViewer restart automatically.
   - Select **Prompt user to restart WindowViewer** to have WindowViewer prompt you to restart WindowViewer.
   - Select **Load changes into WindowViewer** to have WindowViewer load the changes automatically.
   - Select **Prompt user to load changes into WindowViewer** to have WindowViewer prompt you to load changes into WindowViewer.

     Note: If you select the **Load changes into..** or **Prompt user to load..** options, WindowViewer must be restarted to recognize changes to a managed application. Even with these options selected, you will be prompted to restart WindowViewer.
5. In the **Reminder Interval (sec)** box, type how often, in seconds, the user is reminded to load or restart the changes into WindowViewer. This option is only available when the applicable change mode has been set. Set the interval to 0 to not remind the user again.
6. Select **OK**.

Set the default behavior for WindowViewer

1. Open the managed InTouch application in WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **WindowViewer**. The **WindowViewer** configuration screen appears.
3. Select the **Managed Application** tab.
4. Select **Restore Defaults**. The settings are reset to their default values.

5. Select **OK**.

Run ArchestrA scripts in embedded industrial graphics

When you run a managed InTouch application in WindowViewer, any ArchestrA scripts associated with elements or symbol scripts themselves run as expected.

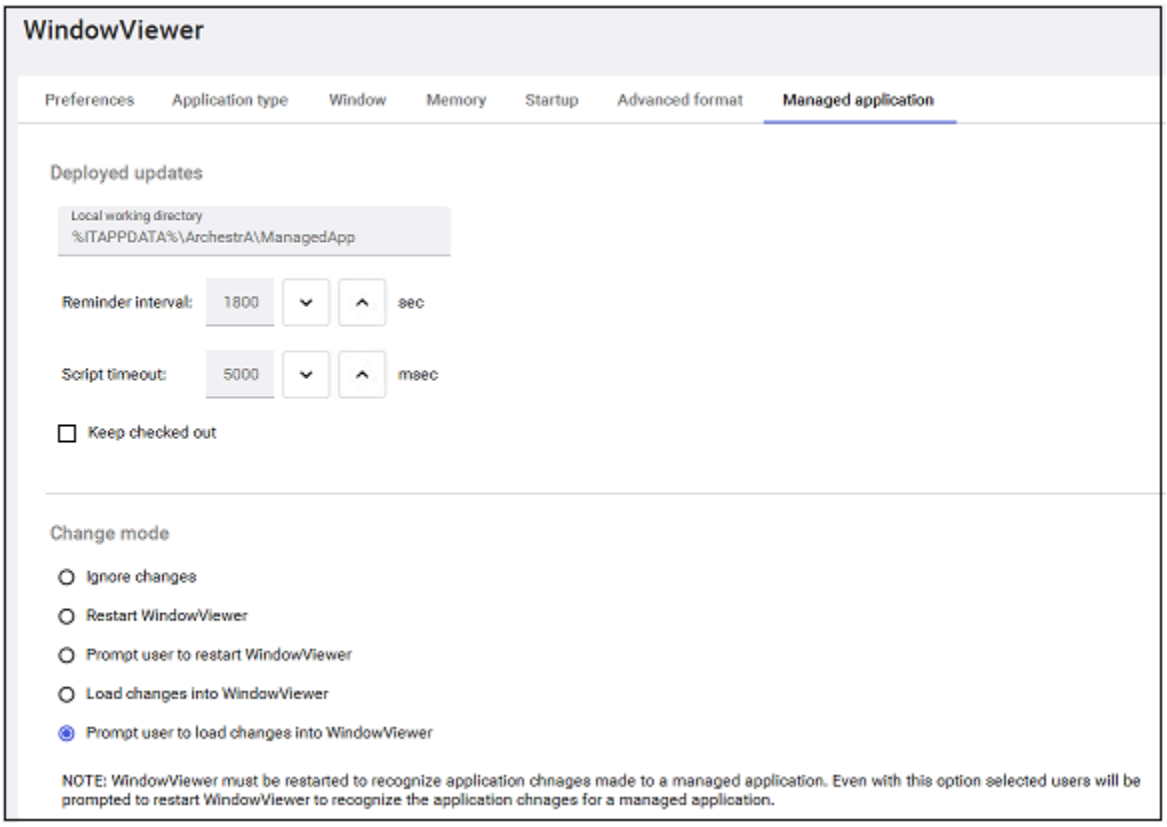
However, some scripts contained in symbols can run for a long time and stop you from interacting with other InTouch elements.

To prevent this, you can set a script time-out that is applicable for all scripts in the managed InTouch application. A script time out stops script execution and returns the control to the operator.

By default, scripts time out after 5 seconds.

Set the script time-out

1. Open a managed InTouch application in WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **WindowViewer**. The **WindowViewer** configuration screen appears.
3. Select the **Managed Application** tab.



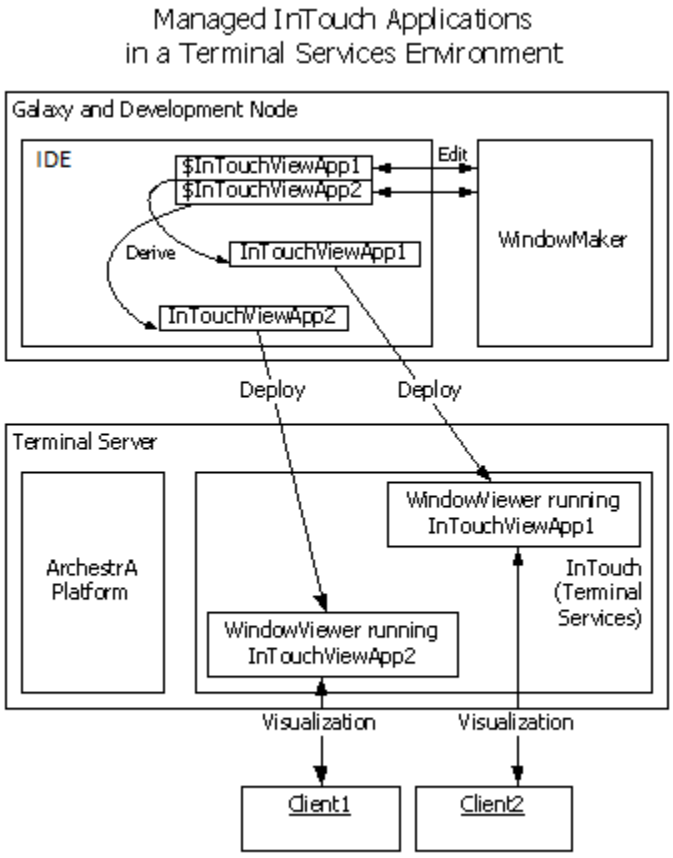
4. In the **Script timeout (msec)** box, type a value in milliseconds.
5. Select **OK**.

Deploy the InTouchViewApp object in a Terminal Services environment

You can run managed InTouch applications in a Terminal Services environment. The main advantage of this architecture is that you can run multiple InTouch applications on one computer at the same time.

Run managed InTouch applications in a Terminal Services environment

- Use InTouch Terminal Services Edition.
- Deploy each InTouchViewApp instance with its own ViewEngine host, if you have several InTouchViewApplication instances on the terminal server node.
- Run each managed InTouch application in its own terminal services client session.



Note: Only one InTouchViewApp object needs to be deployed on the terminal server node to make the application available to multiple terminal session clients. There is no need to deploy a separate InTouchViewApp object on the terminal server node for each client that will use the application.

Use InTouch HMI applications in AVEVA Edge

AVEVA Edge provides users the option to deploy and reuse InTouch HMI applications. Using the Application Manager, you can download applications to AVEVA Edge or upload them to CONNECT.

The downloaded zip files are compressed versions of InTouch HMI applications, suitable for deploying to edge

devices and containers on account of their low file sizes. The same zip can also be uploaded to CONNECT, where it can be downloaded to other devices.

Note: If the AVEVA Edge related options are not enabled, launch WindowMaker.

Download the AVEVA Edge zip file

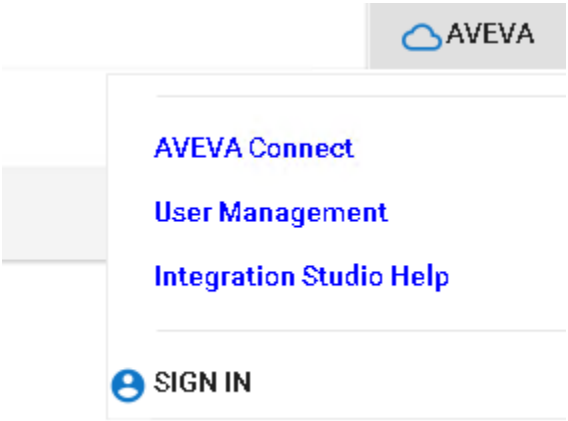
1. Select the application.
2. Select the  icon.

   A compressed (.zip) file will be prepared and ready for download.
3. Save the file.
4. Use the AVEVA Edge Management portal and **Add a New Device**. Upload the zip file under the **Project Configuration** option.

   For further deployment and pairing instructions, see the AVEVA Edge Help.
5. Users can use the InTouch HMI Web Client to verify if the application is running correctly. For more information on the Web Client, see *Viewing Application Graphics in a Web Browser*.

Upload the AVEVA Edge zip file

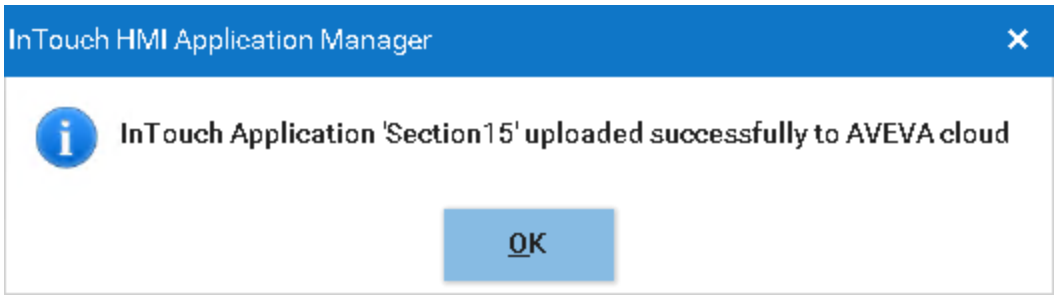
1. Select the **AVEVA** button on the far right of the Application Manager window.



2. Select **SIGN IN**.
3. Enter your CONNECT credentials.
4. After you are logged in, select the application you want to upload.

   Optionally you can change the Drive, but you cannot move applications between drives.
5. Select  .

   A success message is displayed after the application is uploaded to CONNECT.

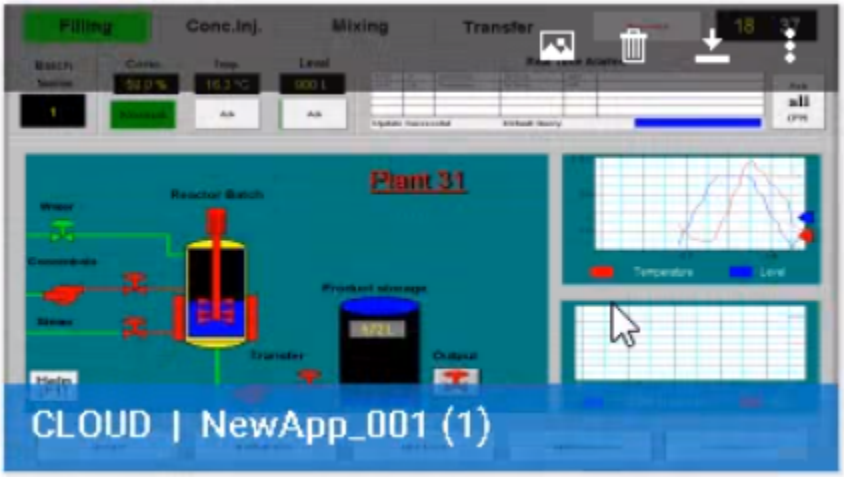


The uploaded application is displayed in the application list, prefixed with *Cloud | <Application Name>*.

The cloud application has the following options:

- Thumbnail
- Delete
- Download

  All other options are disabled.



Once the application is uploaded to the Shared Drive it is accessible to all users. Users can then download and delete the application from the cloud store.

Configure users for Edge device

Users that need to access the InTouch Web Client from the Edge device must be configured in Application Manager. The same users must be present in the appropriate user groups for web client. You can export or import the user list to a csv file. The template for the file is as follows:

```
UserName,Password,IsWrite
TestUser,TestPass#1,1
```

Add Users

1. Select  .

   The User Information dialog box appears.
2. Enter the username and password.

3. The password must comply with the following rules:

   At least 6 characters long

   1 lower case character

   1 upper case character

   1 number

   1 special character
4. Select the access type.
5. Select **Save**.
6. Select  to add new users.
7. Select  to remove users.
8. After you have configured all the users, select  to export the configured list.
9. In AVEVA Edge Management, upload the list, during the 'Add a device' procedure under the User Configuration option.

   After the edge device is configured and paired, the list of users will be created on the edge device. These users will be able to access the web client and view the application graphics.

Import an existing user list

If you have a list of existing users to be included, prepare a .csv file in the required format.

1. Select  and provide the path to the .csv file.
2. The entries in the file will be included in the list of users.
3. Use the AVEVA Edge Management portal and **Add a New Device**. Upload the .csv file under the **User Configuration** option.

   For further deployment and pairing instructions, see the AVEVA Edge Help.

Operate

View applications at runtime

You use WindowViewer to run your InTouch applications. Applications that are designed specifically for use in an Application Server environment are called InTouchView applications. You can use the InTouch Web Client to view Industrial graphics in any HTML5 supported web browser. These applications run in WindowViewer, but the Application Server provides most of the HMI functionality.

About viewing applications at runtime

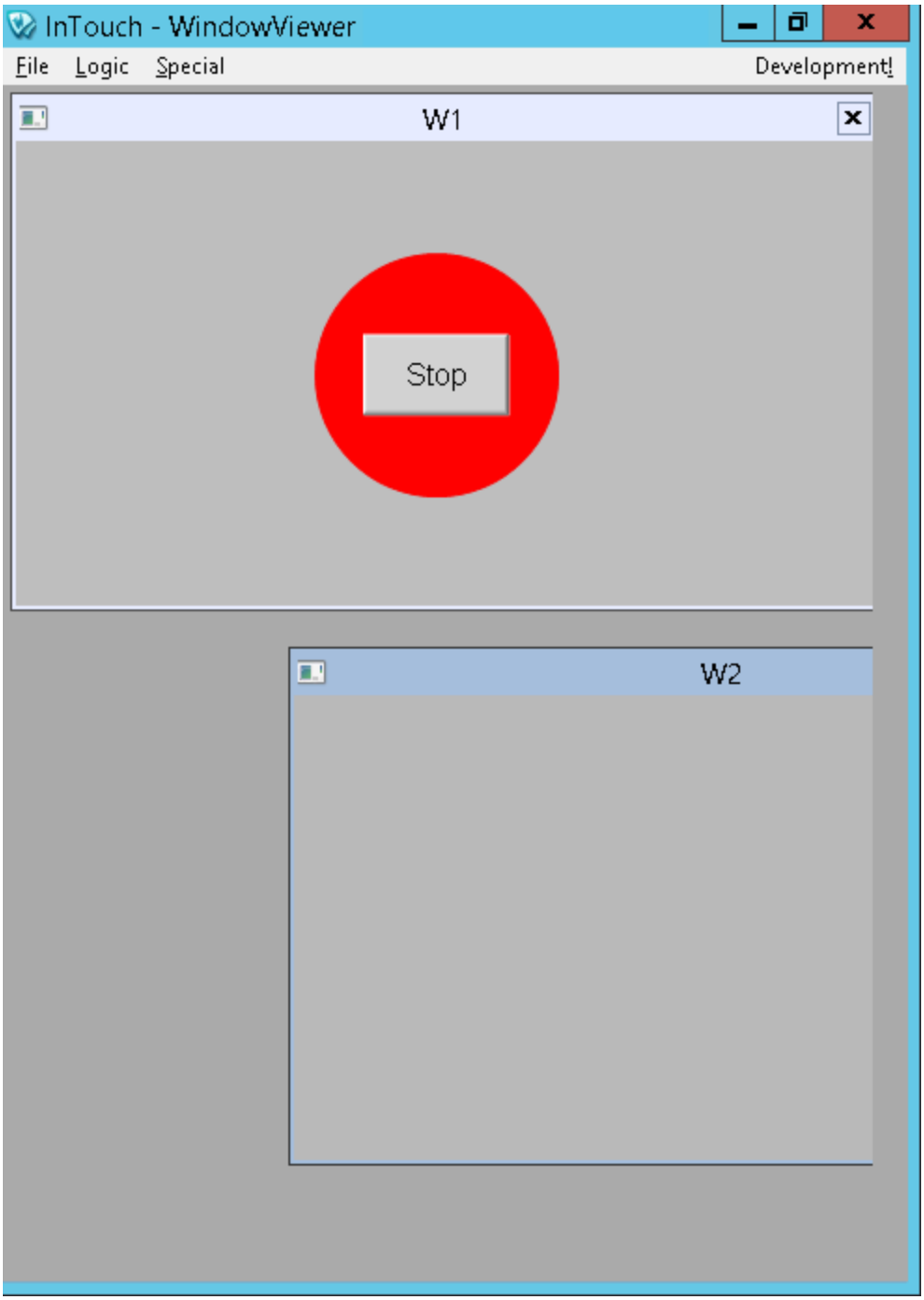
You use WindowViewer to run your InTouch applications. Applications that are designed specifically for use in an Application Server environment are called InTouchView applications. You can use the InTouch Web Client to view Industrial graphics in any HTML5 supported web browser. These applications run in WindowViewer, but the Application Server provides most of the HMI functionality.

View applications at runtime in a different target resolution size

If you have specified an application target resolution that is different than your screen resolution, WindowViewer displays the application at the specified target resolution. Only windows, window controls and graphics developed within the canvas boundary that outlines the target resolution size will display at run time. The target resolution size is automatically adjusted at run time to account for WindowViewer's menu and title bar controls.

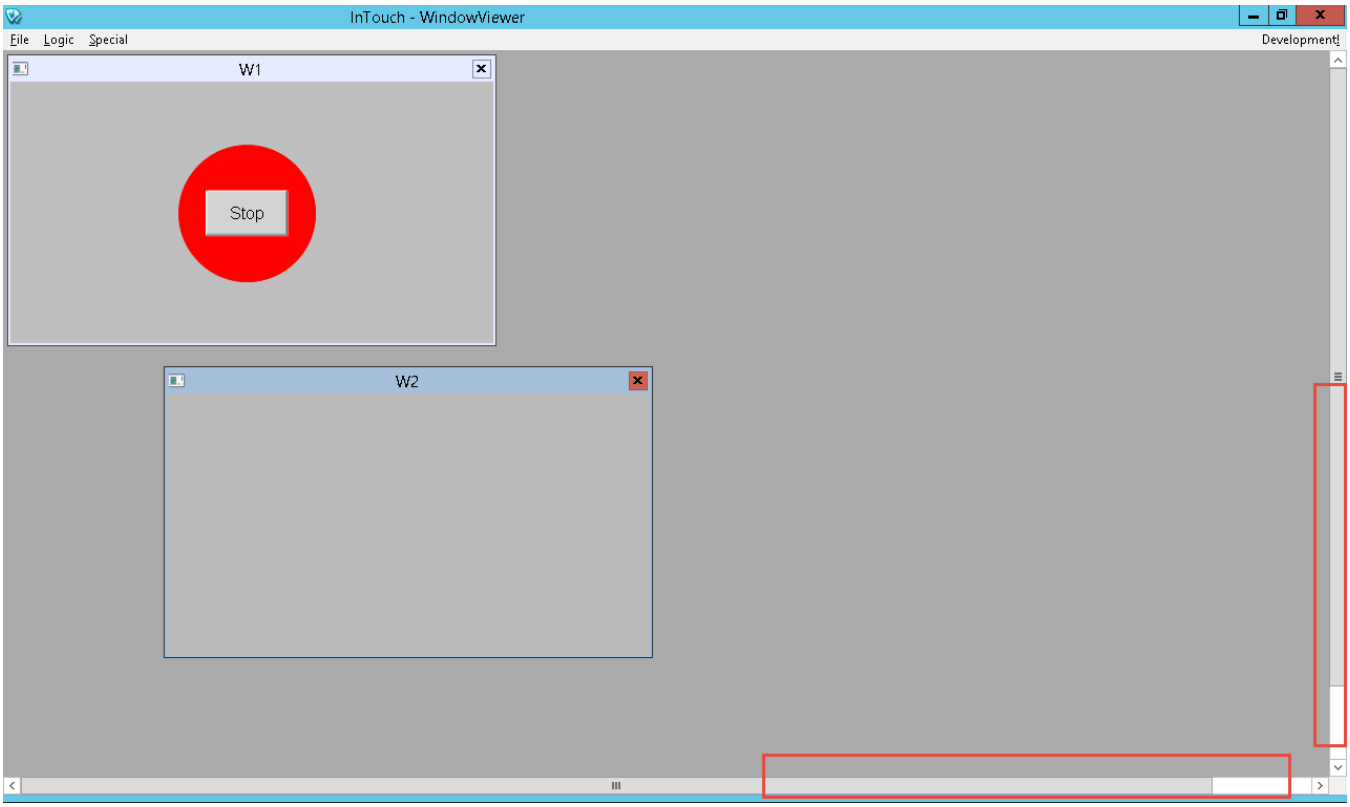
The aspect ratio of embedded controls will be maintained at run time.

For example:



Note: If the specified target resolution is less than the screen resolution, the width and height of the runtime window cannot be adjusted beyond the specified target resolution. Maximizing the WindowViewer window will enlarge it only to the maximum of the target resolution.

If the specified target resolution is greater than the screen resolution, vertical and horizontal scroll bars will display at run time. Windows will scroll accordingly. Popup windows and popup graphics from ShowSymbol animations and ShowGraphic scripts will not be scrolled.



Error messages and popup dialogs will display in the center of the application at its target resolution, not the center of the screen. The same applies to keyboards and keypads.

About the InTouch Web Client

To zoom with mouse gestures:The InTouch Web Client feature allows you to view Industrial graphics used within an InTouch HMI application on any HTML5 supported web browser. A built-in Web Server provides web browsers access to application graphics, from any Microsoft Windows client or server operating system without the use of Remote Desktop Protocol (RDP) or Internet Information Services (IIS) for Microsoft Windows® Server. You can view application graphics in a web browser for both standalone and managed applications. Standalone application windows must be converted to Industrial Graphics before they can be viewed on the Web Client.

For more information on the InTouch Web Client, see Viewing *Application Graphics in a Web Browser*. The Web Client is installed as part of the System Platform installation. For information on configuring the Web Client, refer to the System Platform Installation Guide.

Original application resolution

The original application resolution is the screen resolution when the application was created regardless of the target resolution settings.

The original application resolution is updated only under the following conditions:

- At the time of application creation
- When conversion is applied on the application

If the application is later switched back to screen resolution, then application conversion will occur if, the current screen resolution is different from the original application resolution. Else, no conversion will occur. If a target resolution is used when the application is created it will impact the behavior of the Dynamic Resolution Conversion.

Work with keyboard, mouse, and touch gestures to pan and zoom at runtime

Frame windows allow you to pan and zoom on Industrial Graphics at run time. This functionality is enabled by the **InteractionMode** property in WindowMaker.

Zoom at runtime

You can zoom in and out on the frame contents at run time. Be sure the frame has the correct pan and zoom property enabled. You cannot zoom in above 5000% or below 100%.

You can edit a symbol's **ZoomPercent** property to change the visibility of an symbol or element at run time. For example, add the following to a Visibility animation to allow you to dynamically change the symbol according to the zoom percent level.

```
ZoomPercent => 200
```

Note: You can write to this property at run time.

When **ZoomPercent** is set for a symbol, the symbol will be zoomed to the set percent at the center of the viewable area.

When **ZoomPercent** is set for a symbol’s element, the symbol will be zoomed to the set percent but will center on the element.

The following script is an example of **ZoomPercent** set for an element:

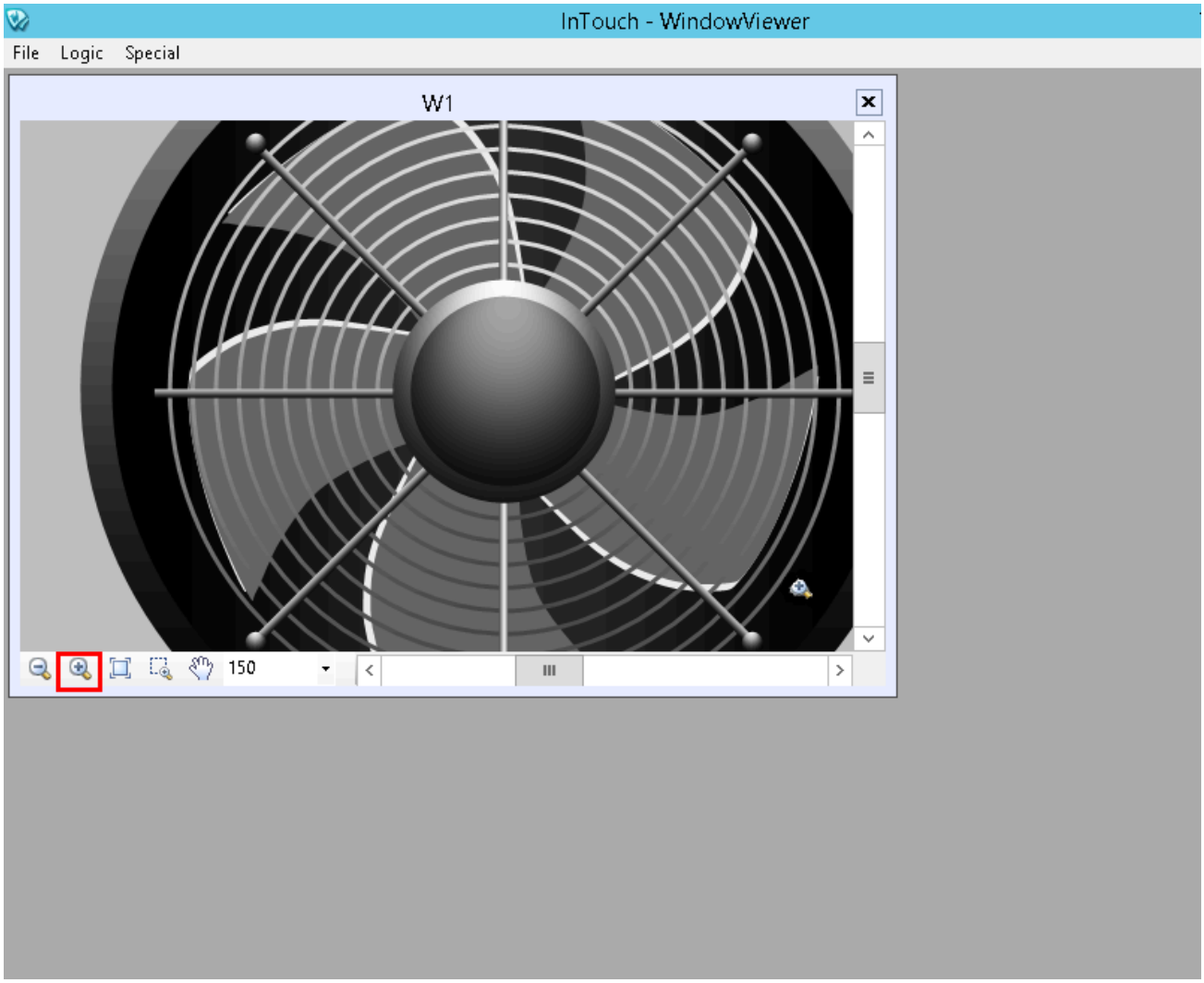
```
TextBox1.ZoomPercent = 500
```

Zoom with mouse gestures

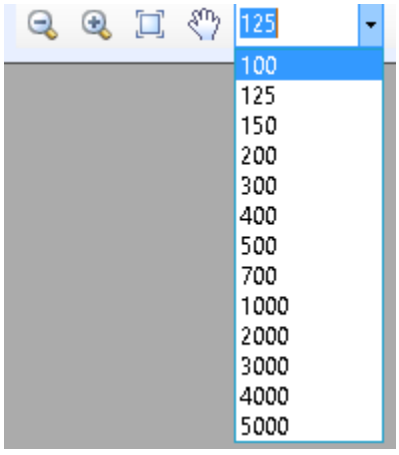
Do the following:

- Press the "Ctrl" key and scroll up with the mouse wheel to zoom in on the frame contents. The contents will zoom in from the current position of the mouse pointer.

  Note: You cannot zoom in on the frame contents if your mouse pointer is outside the frame.
- Scroll down with the mouse wheel to zoom out on the frame contents.
- Select the zoom in and zoom out icons from the Pan and Zoom Control Toolbar. You must then left select the contents of the frame to zoom in.



- Double left mouse select on the frame contents to restore the zoom level to 100%.
- Use the Zoom Level combobox to select a predefined zoom level.



- Select the Rubber Band Zoom icon from the Pan and Zoom Control Toolbar to select a specific area to zoom in on.

Zoom with keyboard gestures

Do the following:

- Press "Ctrl" and "+" keys together to zoom in
- Press "Ctrl" and "-" keys together to zoom out

Zoom with touch gestures

Do the following:

- Place two fingers on the screen and expand them to zoom in
- Place two fingers on the screen and contract them to zoom out
- A double-tap will restore the zoom level back to 100%

Zoom limitations

The following limitations apply to run time zoom functionality:

- Zooming for the Windows Common Controls and client controls is only supported within 500%. These controls include:
  - Customized controls: radio button group, checkbox, edit box, combo box, calendar, datetime picker, and listbox
  - Embedded Alarm Client and Trend Client Controls
  - Third party client controls
- Windows Controls can override mouse, keyboard and touch input
- InTouch will not override custom fonts in Windows Controls

Pan at runtime

Configure the applicable symbol properties to enable pan functionality within a frame at run time. You can pan at run time using mouse and touch gestures. Panning with keyboard gestures is not supported.The zoom level on the graphic must be greater than 100% to pan at run time.

Pan using mouse gestures

Do either of the following:

- Hold down the center mouse wheel.

  The pan hand is displayed
- Select the pan hand from the Pan and Zoom Control toolbar. Hold down the left mouse button and move the pan hand to pan the display.

  The display will pan until the mouse button is released.

Pan using touch gestures

1. Press one finger on the frame content.

2. Move your finger across the screen to pan as needed.

Note: For both panning methods, the horizontal and vertical scroll bars will adjust in accordance with the pan directions.

Pan and zoom simultaneously using touch gestures

1. Place two fingers on the screen and move your fingers downwards, to the left and right to adjust the center of the zoomed content.
2. Use one finger to pan over the frame content.
3. Place your second finger on the frame content to zoom at the same time.

Panning limitations

The following limitations apply:

- panning is not supported using keyboard gestures.
- Window Controls can override mouse, keyboard and touch input. As a result, panning may be disabled over areas with Window Controls.

Animation support for touch gestures

All action scripts configured for touch support should function the same no matter the zoom level. All interaction and visualization animations behave the same while frame content is zoomed in as when frame content is at its standard view. Interaction animations will function properly for touch.

Note: Industrial Graphic pop ups shown by the **ShowSymbol** animation or **ShowSymbol** script function will have pan and zoom enabled by default. However, you cannot disable this configuration.

The following table lists action scripts commonly configured for touch support:

Action Script	Touch Triggered
On Left Down	Touch down
While Left Down	Touch down and slide
On Left Up	Touch up
On Left Double Click	Double tap
On Right Click	Touch down and hold
On Right Up	Touch down and hold for square and execute on release
On Right Double Click	Not supported
While Right Down	Touch down and hold square and slide a bit while pressing

On Center Click	Not supported
While Center Down	Not supported
On Center Up	Not supported
On Center Double Click	Not supported

Note: When touch interaction is used on an area with animation, the animation will take precedence over panning actions and panning actions will be ignored. If one finger retains touch interaction, any subsequent touch points will be ignored.

To enable panning in this scenario,select the **Pan** icon in the **Pan and Zoom Toolbar**.

Touch gesture limitations

Some limitations apply to touch gesture functionality for run time panning and zooming. The below functionality is not supported:

- Scaling fonts for Windows Common Controls

  Note: Additionally, a symbol with an embedded windows control uses different mechanisms for scaling than a symbol without a control. A symbol with an embedded control has a maximum zoom limitation of 500%, while a symbol without a control can zoom up to 5000%. The advantage of a symbol without a control is smoother scaling. An additional limitation of using symbols with embedded controls is, while zooming is in progress, the control will flicker. This is particularly visible while zooming with touch gestures.

Use the ShowGraphic() function with frame windows

You can run the ShowGraphic() script function to change the symbol associated with a frame window at run time. Specify the frame window name and associated symbol as in the following example:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "InTouch:FrameWindow01";
graphicInfo.GraphicName = "Symbol_002";
ShowGraphic( graphicInfo );
```

"Symbol_002" will display in "FrameWindow01" at run time regardless of the frame window's design time configuration. You can use a ShowGraphic script to host different symbols in the same window.

Limitation:

Graphic caching occurs only for the symbol currently shown in the frame window at the time the window is closed and cached. Replaced symbols will not be cached.

Set windows to appear at runtime

"Home" windows are windows that appear in WindowViewer when the user starts WindowViewer directly, either from an icon or a menu command.

Home windows do not appear if you use the **Runtime** fast switch to start WindowViewer.

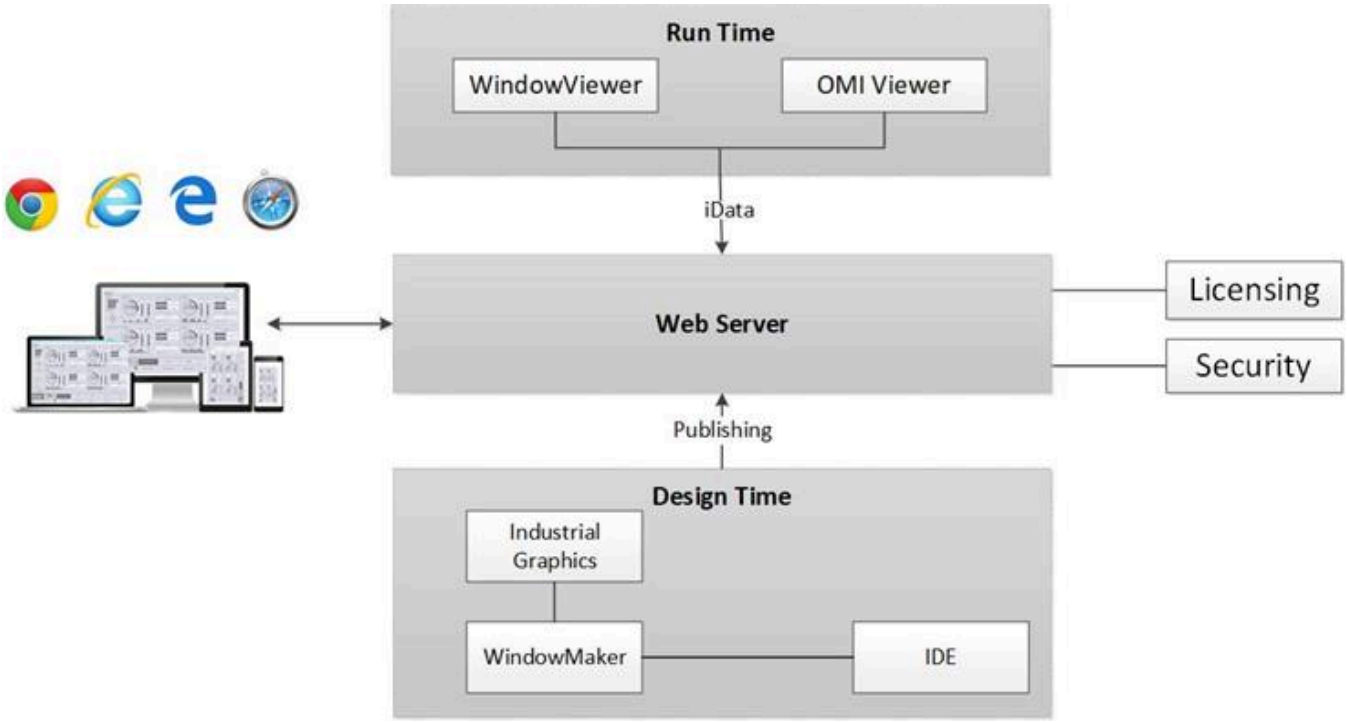
You can show home windows at any time during run time by using the ShowHome() function in a script.

Set home windows

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **WindowViewer**. The **WindowViewer configuration** screen appears.
3. Click the **Startup** tab.
4. Select the window or windows to open when WindowViewer starts.
5. Click **OK**.

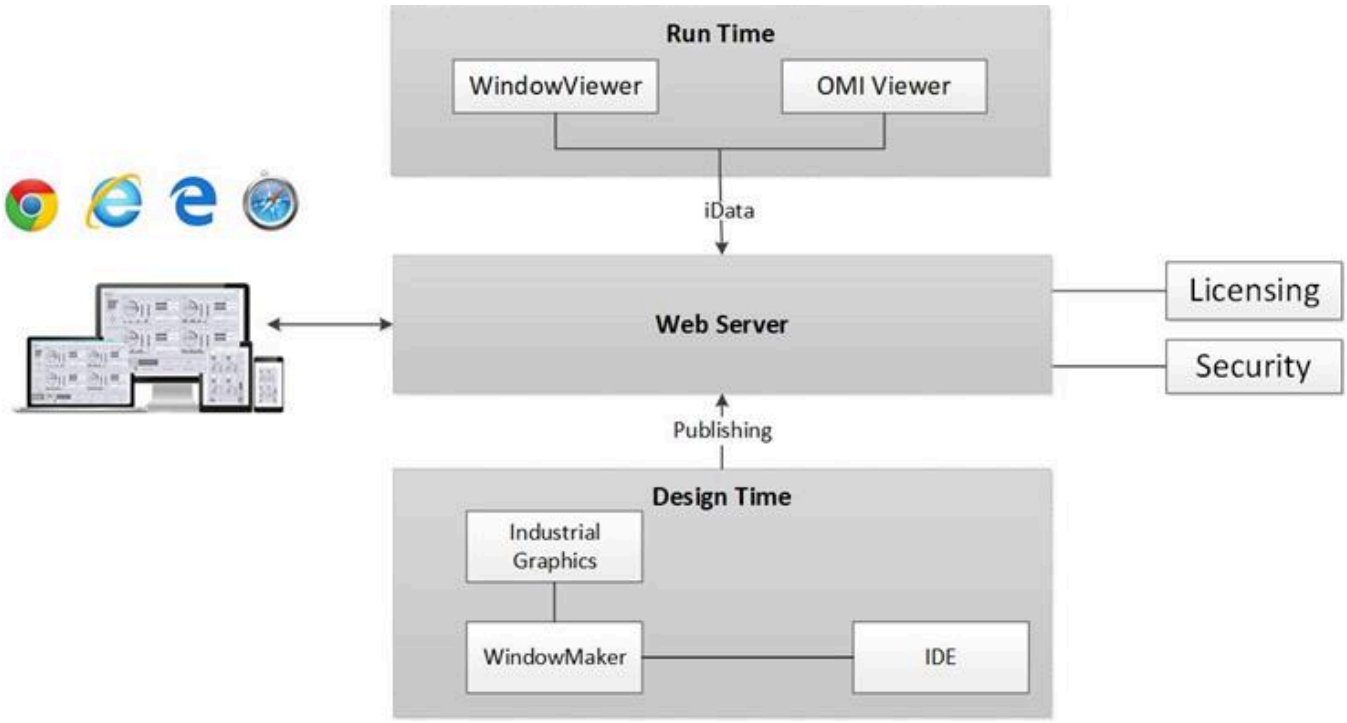
View application graphics in a Web Client

Web Client allows you to view AVEVA InTouch HMI applications and AVEVA OMI ViewApp on any HTML5 supported web browser. Web Client is also available on mobile platforms. A built-in Web Server provides web browsers access to applications, from any Microsoft Windows client or server operating system without the use of Remote Desktop Protocol (RDP) or Internet Information Services (IIS) for Microsoft Windows® Server.



Use the Web Client

Web Client allows you to view AVEVA InTouch HMI applications and AVEVA OMI ViewApp on any HTML5 supported web browser. Web Client is also available on mobile platforms. A built-in Web Server provides web browsers access to applications, from any Microsoft Windows client or server operating system without the use of Remote Desktop Protocol (RDP) or Internet Information Services (IIS) for Microsoft Windows® Server.



Design an InTouch Web Client application

Designing InTouch Web Client applications involves the following stages:

Configure (Optional)

1. Configure the security protocol.
   a. Connect to a System Management Server using the System Platform Configurator - choose between a remote or local server.
2. Configure AVEVA Identity Manager.
3. Configure the reverse proxy server.

Design

1. Create and/or edit an application.
2. Identify the root folder that will contain all the graphics you want displayed on the web client.
3. Identify and set the Web Client Root folder.
4. Identify and set the Web Client Home symbol.
5. Fast switch to InTouch WindowViewer.
   - WindowViewer is required to run to access InTouch tags, but not application server object attribute references.
6. Use the Web Client fast switch button or launch a browser and navigate to http://localhost/InTouch.

Iterative Development

1. The home symbol is displayed on the web page.
2. Use the Navigation overlay to navigate to a symbol/window.
3. Make any graphic related changes in WindowMaker, the changes are automatically refreshed on the browser.

   Changes in Quality and Status Style, Element Style, and Formatting Styles will be propagated only after making the following graphics changes or re-launching WindowViewer.

   - Content of symbol is updated and saved
   - Symbol is created, imported, or deleted
   - Symbol is moved to different toolset folder
   - Root folder or home symbol assignment is updated
4. You can continue to build the application and test the output in the browser.

Deploy

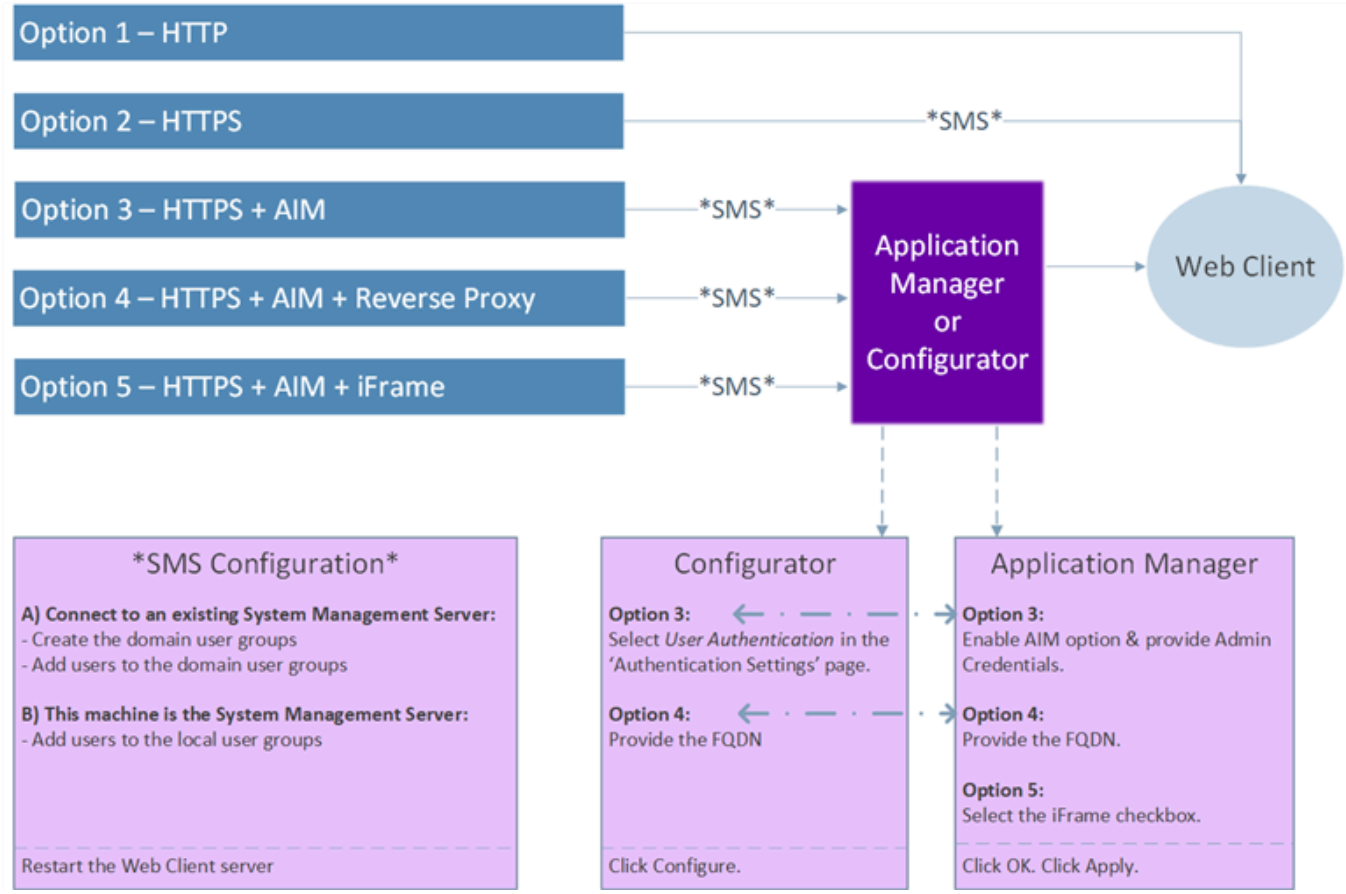
1. Assign users to one of the Web Client related user groups (local or domain) or opt for anonymous access.
2. Publish or deploy the application.
3. Run the application in InTouch WindowViewer. WindowViewer must be running, to receive live data from InTouch tags.

   After the application is published or deployed, it will be available from any computer on the intranet.

   You can point to http://<IPAddress>/InTouch or http://<NodeName>/InTouch, where <IPAddress> or <NodeName> corresponds to the machine where the application is published or deployed.

Secure InTouch Web Client access

Configuring the System Management Server (SMS) allows InTouch Web Client to be used securely with the https protocol. Additionally, after configuring the SMS, InTouch Web Client can be used with the AVEVA Identity Manager (AIM), to support non-windows based security and single sign on. As an administrator there are multiple configuration options available for user authentication and security. Security and user management should work together, see Configure user access in InTouch Web Client for more information.



Option 1: By default, InTouch Web Client can be accessed using the http protocol and windows-based authentication.

Option 2: Use the Configurator to connect to a local or remote System Management Server. Here the InTouch Web Client will use the https protocol with windows-based security (by default). If you connect to a remote server, you must provide credentials for a domain user when accessing the Web Client.

Option 3: After you configure the System Management Server, you can optionally enable the AVEVA Identity Manager (AIM). AIM is a standalone authentication server that exposes an OpenID Connect endpoint. To use AIM, you must register the client to the identity server. You can configure AIM using the System Management Server Configurator or InTouch HMI Application Manager. For more information, see Register with the AVEVA Identity Manager.

In runtime when the InTouch Web Client page loads, if there is no valid security token, then:

- Web Client will re-direct to AIM's login page.
- AIM will check user credential from the Active Directory.
- If the credentials are valid, then Active Directory will provide a security token and return it to Web Client.
- Web Client will then grant access to user with the token.

If a security token already exists, then the user will be granted access. AIM will only support users that can be validated from Active Directory.

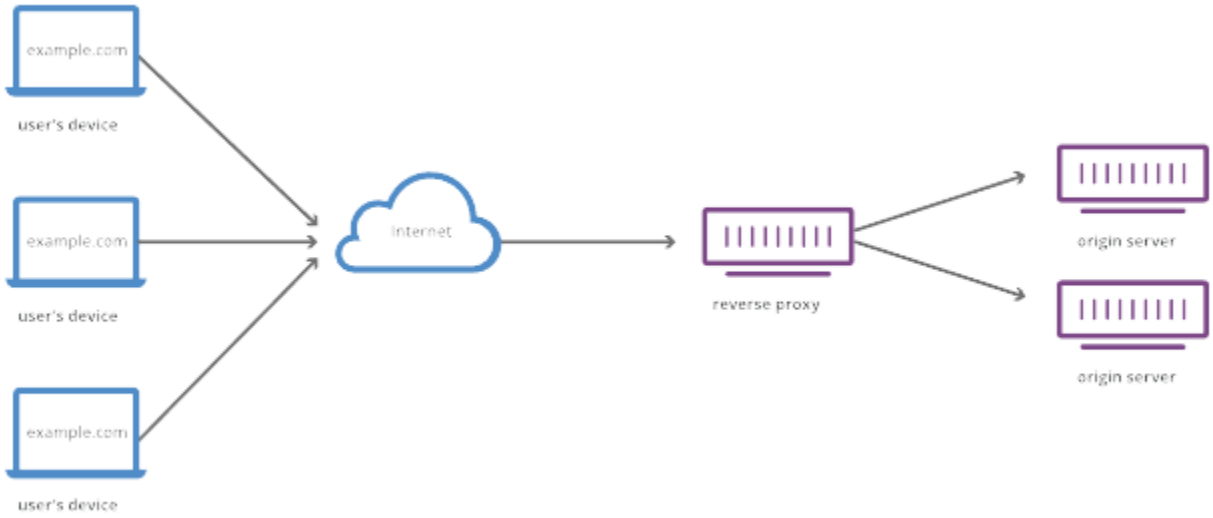
Option 4: The Reverse Proxy option allows the InTouch Web Client to be accessed by users outside the OT network. Provide the FQDN of the reverse proxy server in the secure gateway field in the Configurator or the Web Client tab in InTouch HMI Application Manager.

Option 5: When AIM is used for user authentication, it will prevent graphics from being displayed in an iFrame. Selecting the **Allow web client to be embedded in any website** checkbox will allow users to display graphics within an iFrame in runtime. In runtime, use the Share icon and select the code snippet to insert in the iFrame.

Access InTouch Web Client from outside the OT network

A reverse proxy is a server that positioned in front of one or more web servers, intercepting requests from clients. With a reverse proxy, when clients send requests to the origin server of a website, those requests are intercepted at the network edge by the reverse proxy server. The reverse proxy server will then send requests to and receive responses from the origin server. If any of these web servers are down or in case of the fail-over of one of these web servers, then the reverse proxy will send the requests from clients to the redundant web server.

A forward proxy positioned in front of a client and ensures that no origin server ever communicates directly with that specific client. On the other hand, a reverse proxy sits in front of an origin server and ensures that no client ever communicates directly with that origin server.



Before devices outside the OT network can access the InTouch Web Client, a reverse proxy must be configured on the DMZ server. AVEVA Identity Manager must be enabled for the reverse proxy to function. There are many providers of reverse proxy solutions. Refer to the relevant reverse proxy solution documentation,the information on redundancy support and instructions on setting up the reverse proxy solution in your infrastructure.

Register with the AVEVA Identity Manager

Using the AVEVA Identity Manager (AIM), you can configure the InTouch Web Client to use Single Sign On, instead of the default Windows OS-based authentication.

Note: InTouch Web Client supports ArchestrA-based security with AIM configuration only.

Configure Identity Server

1. In the System Platform Configurator, configure the **Common Platform** > **System Management Server**.
2. In AVEVA Application Manager, register the AIM server with user credentials.

The AIM Registration dialog box can also be used to configure the reverse proxy server:

1. Setup the reverse proxy server.
2. In the System Platform Configurator, configure the **Common Platform > System Management Server**.
3. Provide the Secure Gateway address in the AIM registration dialog box.

You can select a remote or local System Management Server. For more information on configuring the System Management Server, see *System Platform Installation*.

Register with the Identity Server

1. In Application Manager, select the **Web Client** tab.
2. Select **Tools** and then **Identity manager**.

   The **Identity server settings** screen appears.
3. Select the **Use AIM Server as the authentication server** checkbox to enable the use of the AVEVA Identity Manager.

   The **Identity Server** field displays the Identity Server configured in the Configurator.
4. Update the following settings:
   a. **User Name**: Provide the user name to connect to the Identity Server. The user must be part of the Administrators user group.
   b. **Password**: Provide the password for the corresponding user name.
5. To complete the reverse proxy setup, provide the URL of reverse proxy or DMZ server in the **Secure Gateway** field.

   This is an optional setting and only needs to be set if the Web Client is hosted behind a reverse proxy server.
6. Optionally you can also select the **Allow Industrial Graphics to be embedded in any website** checkbox to view the web client within an HTML iframe in runtime.
7. Select **OK**.
8. Select **Apply**.

Note: If the web site name or address is changed, you must configure the AVEVA Identity Manager to register the new web site.

Access the InTouch Web Client using single sign on

If the System Management Server option has been enabled in the Configurator, you can use Single Sign On to access the InTouch Web Client. This will be different from the default Windows based authentication. This method will use the HTTPS protocol and only support domain users (when using a remote SMS) that can be validated from Microsoft Active Directory. If the user selects to use the HTTP protocol, then Web Client will fall back to using Windows based authentication only.

For more information on configuring the System Management Server, see *AVEVA System Platform Installation.*

1. Launch Web Client.

   The login page appears.
2. Enter your username and password.
3. Select **Login**.

   Optionally, you could select on **Windows Integrated** to use the credentials of the user currently logged in.

You will be logged with the Identity Manager credentials.

4. Use **Sign Out** to logout of the session.

Use local tags with AIM

The behavior of Local tags in InTouch Web Client is independent of WindowViewer. The behaviour of local tags will differ if AIM is enabled. If AIM is enabled a unique token is used for every session for every user.

	User 1 – Session 1	User 1 – Session 2	User 2 – Session 1
AIM Disabled	Same tag value	Same tag value	Different tag value
AIM Enabled	Different tag value	Different tag value	Different tag value

Configure user access in InTouch Web Client

Accessing the InTouch Web Client Anonymously

InTouch Web Client provides both anonymous and authenticated user access. By default, anonymous access is disabled. To enable this option, select the **Allow Anonymous Access** checkbox in the Web Client tab in InTouch HMI Application Manager. In runtime, the user does not have to provide any user credentials and will be logged in as a 'Guest' user. The Guest user will have read only access to Web Client.

Configuring Authenticated Users for InTouch Web Client

Both authentication and licensing workflows use user groups to determine which users get access and the type of license acquired for the web session. The user groups can be configured on the local or remote node depending on the SMS configuration. All users needing access must be included in one of the groups depending on their access level.

If Web Client is configured to use a remote System Management Server, create the following domain user groups: aaInTouchUsers and aaInTouchRWUsers on the remote server. Add all relevant users to these groups before deploying the application.

Use virtual account for InTouch Web Client

The Web Client virtual service account has had its privileges reduced to improve security. To continue using the Web Client for an InTouch Application, the InTouch Web Client virtual service account must have read/write access to the InTouch Application Folder. The virtual service account for the InTouch Web Service is called "NT SERVICE\AIGWebServer".

The InTouch Web Client virtual service account will be granted read/write access to:

- all existing InTouch applications.
- all InTouch applications created/imported via Application Manager
- all InTouch applications found via 'Find' operation; only if the users select Yes to the dialog which prompts to grant Web Client virtual account read/write access to the found application.

To manually grant the InTouch Web Client virtual account access to an InTouch application:

1. Navigate to the InTouch Application folder using File Explorer.
2. Right-select the folder and select **Properties**.
3. In the **Properties** dialog, under the **Security** tab, select **Edit**.
4. In the **Group or usernames** section, select **Add**.

   The **Select Users, Computers, Service Accounts, or Groups** dialog appears.
5. Select **Locations**.

   The **Locations** dialog appears.
6. Select the computer name and select **OK**.
7. In the **Enter the object names to select** field, enter NT SERVICE\AIGWebServer.
8. Select **OK**.
9. In the **Permissions for Authenticated Users** section, under the **Allow** column, select the check-boxes: Read & execute, List folder contents, Read, and Write.
10. Select **OK**.

Note: The InTouch Web Client virtual service account will not be granted access to the application folder in a shared location, that is, if the application path begins with "\\", such as the path to a file share.

Enable the InTouch Web Client feature

By default, on installation the InTouch Web Client feature will be disabled. You must enable the InTouch Web Client feature before you can use it. The InTouch Web Client feature will be automatically enabled if you configure one of the following options in the Configurator:

- **Common Platform** > **System Management Server**
- **Application Manager** > **Web Client**

For more information on the Configurator, refer to *System Platform Installation*. To enable or disable the Web Client feature manually, you must have administrative privileges. You can enable the Web Client feature from Application Manager or InTouch WindowMaker.

Enable the InTouch Web Client Feature from Application Manager

1. Launch Application Manager.
2. Select the Web Client Tab.
3. On the **File** menu, in the **Main** group select **Web Client**.
4. Select **Web Client** again to disable.

Enable the InTouch Web Client Feature from WindowMaker

A user with administrative privileges can enable the web client feature from WindowMaker.

1. Launch an application in WindowMaker.
2. Select the **Web Client** fast switch button.

The following dialog box appears: 'InTouch Web Client feature is currently disabled on this system. Do you want to enable it?'.

3. Select **Yes**.

   The InTouch Web client page appears.

If you try to launch the Web Client when it is disabled, you will receive the following error message: **HTTP Error 404.0 – Not Found**.

If a non-administrative user attempts to enable the Web Client feature from WindowMaker, they will be directed to enable the feature from Application Manager.

Customize the InTouch Web Client

The Web Client tab of the AVEVA Application Manager provides options for the user to configure InTouch Web Client related settings.

Update the InTouch Web Client settings

1. Launch the Application Manager and select the **Web Client** tab.

   The **Web Client settings** screen appears.
2. Configure the following settings:
   - *Current application:* Select an application from the list to modify the web client settings.
   - *Graphic refresh rate (ms)*: Set the rate on how frequently the web browser will query the web server for graphic data. The default is 1 second. For more information, see System Platform Installation.
   - *Alarm refresh rate (ms)*: Set the rate on how frequently the web browser will query the web server for alarm data. The default is 1 second. For more information, see System Platform Installation.
   - *Show header*: Select the checkbox to display the Title Bar.
   - *Enable navbar*: Select the checkbox to display the Navigation Bar.

     This setting will be disabled if the **Show header** setting is not selected.
   - *Allow Anonymous Access*: Select the checkbox to allow users access to web client without authentication.
   - *Enable workspaces*: Select the checkbox to enable Workspaces.
3. Under the Advanced settings section, select **Customize** to configure the following settings:
   - *Website name*: Provide a string that will replace the standard URL.
   - *Application title*: Provide a string that will appear as a title for the application in the App Bar.
   - *Website title*: Provide a string that will appear as the title on the title bar of the browser.
   - *Website icon*: Provide an image file that will replace the icon on the title bar of the browser.
     a. Select **Choose file**... to select the icon image.
     b. Select a file.
     c. Select **Open**.
4. Select **Save** to save the advanced settings.
5. After modifying the settings, select **Apply**.
6. To view the InTouch Web Client, select **Launch**.

Note: If the web site name or address is changed, ensure to configure the AVEVA Identity Manager to register the new web site. For more information, see Register with the AVEVA Identity Manager.

For more information on these settings, see Viewing *Application Graphics in a Web Browser*.

Configure graphics for viewing in InTouch Web Client

Using the Graphic Toolbox, you can configure the folder containing the application graphics that will be displayed in InTouch Web Client. Only graphics stored within the folder set as the Web Client Root folder are displayed on the web browser.

Note: The operating system user groups **aaInTouchUsers** and **aaInTouchRWUsers** will be created on the InTouch runtime node during installation. Only users in the aaInTouchUsers or aaInTouchRWUsers user groups will have access to view graphics from an application in the web browser. Add relevant users to the group before configuring the application. By default, the installation user will be added to the groups.

1. Start InTouch WindowMaker.
2. Open an application.

   In the **Industrial Graphic Toolbox**, create or identify the folder that will contain the hierarchy of folders and graphics you want displayed on the web browser.
3. Right-select the folder and select **Set the Web Client Root Folder**.

   The icon thumbnail will change to reflect the setting.

Icon	Description
	When the root folder is set as the Web Client Root folder
	When a folder other than the root folder is set as the Web Client Root folder

   By default, the root of the Graphic Toolbox is set as the Web Client Root folder.

Set the InTouch Web Client home symbol

You can set the default symbol displayed on the InTouch Web Client page when it is launched by setting the Web Client Home Symbol.

1. Set the Web Client Root Folder, see Configure graphics for viewing in InTouch Web Client.

   The Home Symbol must be a child of the Web Client Root folder.
2. Under the Web Client Root folder, identify the home symbol.
3. Right-click the symbol and select **Set Web Client Home Symbol**.

The icon thumbnail ( ) will change to reflect the setting. If no home symbol is set, the Web Client web page will display a blank page when launched. While viewing the Web Client in a browser, select the Home icon on the App bar to navigate to the home symbol.

Understand the behavior of the InTouch Web Client home icon

Standalone applications can be viewed on the web client. Standalone application windows must be converted to Industrial Graphics before they can be viewed on the Web Client. Only Frame windows and Industrial Graphics can be viewed on the web client.

For InTouch applications, in WindowMaker you can use the Home Windows tab to set up windows that will appear by default in WindowViewer, see Set windows to appear at runtime. Similarly, you can set a graphic to be the default symbol that appears on the web client when it is launched, see Set the InTouch Web Client home symbol.

If you have used the InTouch ShowHome script function, when the windows is converted to a graphic, the script is converted to the ShowHome() function and will display the corresponding graphic as a Home symbol in WindowViewer.

These settings will impact, which window or graphic appear on the InTouch Web Client by default when it is launched. For an InTouch HMI application; there are two possibilities:

1. If Home Windows are configured

   Even if the user has configured Home Symbols, on all occasions, only the Home Windows are displayed by default. Selecting the Home icon will display the Home Windows.
2. If Home Windows are not configured

   And the Home Symbol is configured, by default the home symbol will be displayed.

   If no home symbol is configured then, by default, no symbol or home windows are displayed. Selecting on the Home symbol will not display any graphic or window.

View applications in a web browser

After you configure the InTouch Web Client application you can view the application in the web browser. For a list of tested and supported browsers, see Browser and Mobile Support.

Note: For InTouch Web Client to function correctly, it is recommended to always install the latest version of browser or upgrade your browser to any latest version available.

1. Launch an HTML5 supported browser and enter the URL: `http://<hostname>/InTouch` or `http://<IPAddress>/InTouch`.

   If your credentials are authenticated and a valid Web Server license is acquired, the InTouch Web Client page is displayed. The Home Symbol will be displayed by default.

   You can access the web client using the URL: https://<hostname>/InTouch or https://<IPAddress>/InTouch. The Web Client will automatically use the HTTPS protocol, when the Common Platform System Management Server is configured with a certificate (local or remote server) for encrypted communication between the server and client. If the Web Client page is not displayed, refer to Troubleshooting Viewing Graphics in a Web Browser.

   For information on customizing the Web Client URL, see Customize the InTouch Web Client.
2. Select the Hamburger icon.

   The overlay on the left displays the hierarchical structure of the graphics, under the selected Web Client Root

folder within the running application.

3. Select a folder name.

   The graphics within the folder are displayed.

4. Select a graphic.

   The selected graphic is displayed on the InTouch Web Client.

Only graphics stored within the folder set as the Web Client Root folder are displayed on the web browser. If anonymous access disabled, then the user account accessing the graphics must be Authenticated and Authorized, for more information see Acquire a license for InTouch application graphics.

Note: For improved graphics rendering and better overall performance, we recommend enabling GPU hardware acceleration.

InTouch WindowViewer must be running on the host machine, with a Read/Write InTouch license, for symbols in the Web Client to receive live data from InTouch tags. You can also run WindowViewer as a service. Application Server must be running to receive live data from application objects.

Note: A Read-Only InTouch license does not support remote data connections, and therefore does not provide sufficient functionality to act as a data source for the InTouch Web Client. All WindowViewer applications feeding data to InTouch Web Client must be running under a Read/Write InTouch license."

InTouch Web Client fast switch

During the development of the application you can use the Web Client Fast Switch feature to toggle between WindowMaker and the InTouch Web Client.

- To use the Web Client Fast Switch feature, select the **Web Client** button in InTouch WindowMaker.

Any changes in WindowMaker will reflect automatically on the Web Client, if the same application is being edited in WindowMaker and viewed in the Web Client. For automatic refresh to work, only one application can be edited and viewed at a time. The InTouch Web page refreshes to the symbol previously displayed. If WindowViewer is run for another application, then the Web Client page will automatically switch to that application.

- You can specify the **Graphic Refresh Rate** from the **Configurator**. By default the polling rate is 1000 milliseconds.
- You can also specify the **Alarm Refresh Rate** from the **Configurator**. By default the polling rate is 1000 milliseconds.

Alarm Refresh Rate should be greater than the Graphic Refresh Rate, and the value of the Graphic Refresh Rate should be less than 180 seconds.

Note: To enable the InTouch Web Client to run in Fast Switch mode while editing a Managed InTouch application from a remote IDE, first configure the **InTouch Web** Windows service to run as a user account that is a member of the **aaConfigTools** user group, in both the GR and Remote IDE machines. For more information, refer to Application Server User documentation.

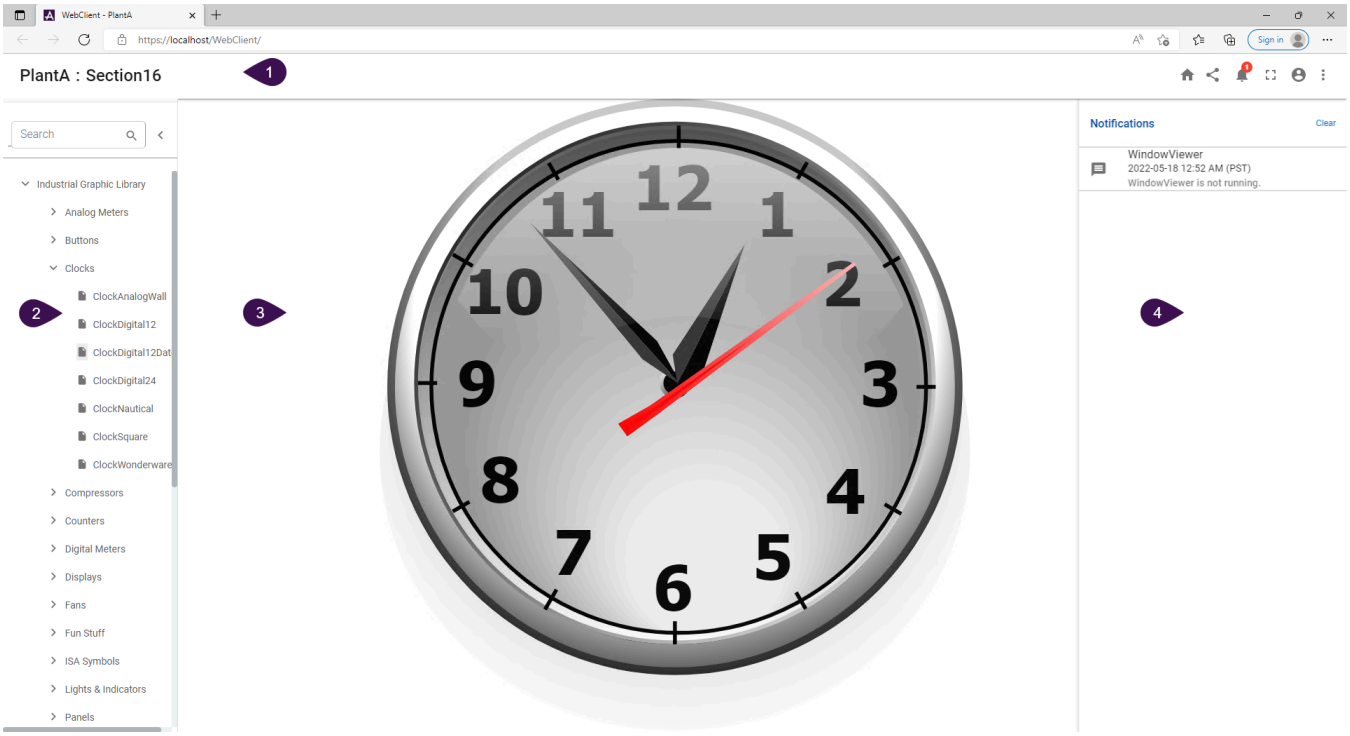
For a given InTouch application, the Web Client fast switch button in WindowMaker is disabled by default. The button is automatically enabled when the application is run in WindowViewer.

Note: The web client fast switch feature is available:

- only during development,
- only on the computer with InTouch WindowMaker and InTouch WindowViewer installed,
- only on a web browser accessing the Web Client pointing to *http://localhost/InTouch* in the URL.

Understand the InTouch Web Client page

The InTouch Web Client page provides various options to inform and organize the user experience using icons on the page. The following table describes these options.



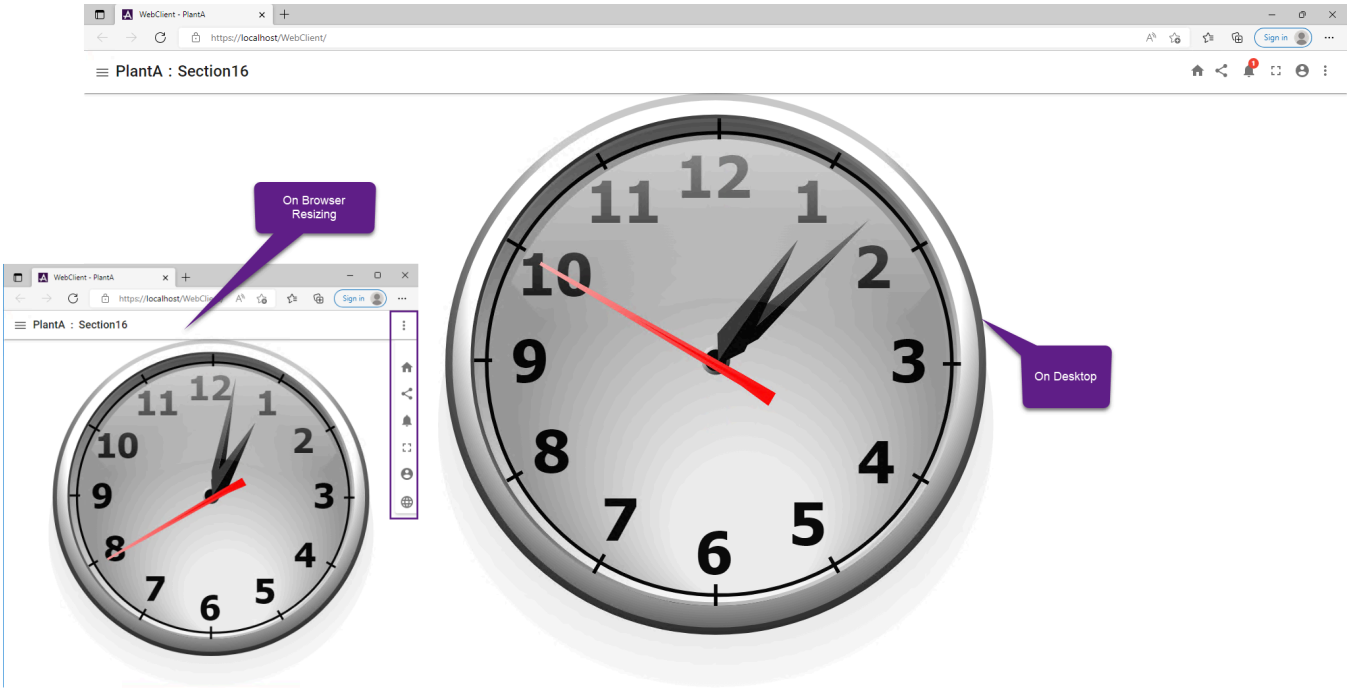
Element	Description
1	The App bar displays the InTouch Application name, symbol name, home icon, notifications icon, full screen icon and profile icon.
Application Name: Symbol Name	The App bar will display the application and symbol name. For example – *PlantA: Section16* - where PlantA is the application name and Section16 is the symbol name.
	Select to view the symbol set as the web client home symbol.
	Select to view the iFrame code snippet for the selected graphic.

	Select the Notifications icon messages related to License status change and WindowViewer status change messages. The number of notifications will be overlayed on the icon.
	Select to maximize the canvas area and display the web page in full screen mode. You can also use the F11 key. The app bar and navigation overlay will not be visible. **To view the title bar:** • Move your cursor to the top of the page, the app bar will slide down. • Select the down arrow to pin the app bar on the page. • Select the up arrow to hide the app bar.
	Select to minimize the canvas area and exit the full screen mode.
	The profile icon displays the user logged in.
	Select the profile icon and then select the **Sign Out** icon. The Sign Out will only be available if System Management Server option is configured. For more information, see Access the InTouch Web Client using single sign on.
	From the App Bar select , and then select the **Language** icon. Select the language. The tooltip, translated static text and custom properties will display in the selected language. Only languages configured in WindowMaker will appear. **Note:** When the user changes the language, a new session will be created causing the data in the local tags to be lost.
2	Select the  icon to view the folder navigation overlay. All folders and symbols under the selected Web Client root folder are displayed. You can use the search box at the top of the overlay to search for symbols. The search box is enabled with the autocomplete feature, where symbol names matching the input search term will be listed as you are typing.

	Select the symbol name to view the symbol in the canvas area.
3	The canvas area displays the symbol selected from the navigation overlay. The symbol will be scaled to fit the browser viewport size, while maintaining the aspect ratio. The navigation and notifications overlays do not consume any space on the canvas or affect the size of the graphic.
4	Select the notifications icon to view the notifications related to connection and licensing issues.
⚠	Select the warning icon to view the list of animations or properties not supported by the InTouch Web Client. This icon is available only during application development and when the browser points to the exact URL of http://localhost/InTouch.

Application graphics and browser resizing

The InTouch Web Client page can be viewed in different screen sizes. As the browser is adjusted the graphics resize to fit the new browser viewport dimensions. The graphic will also resize when the mobile device is rotated between portrait and landscape orientation.

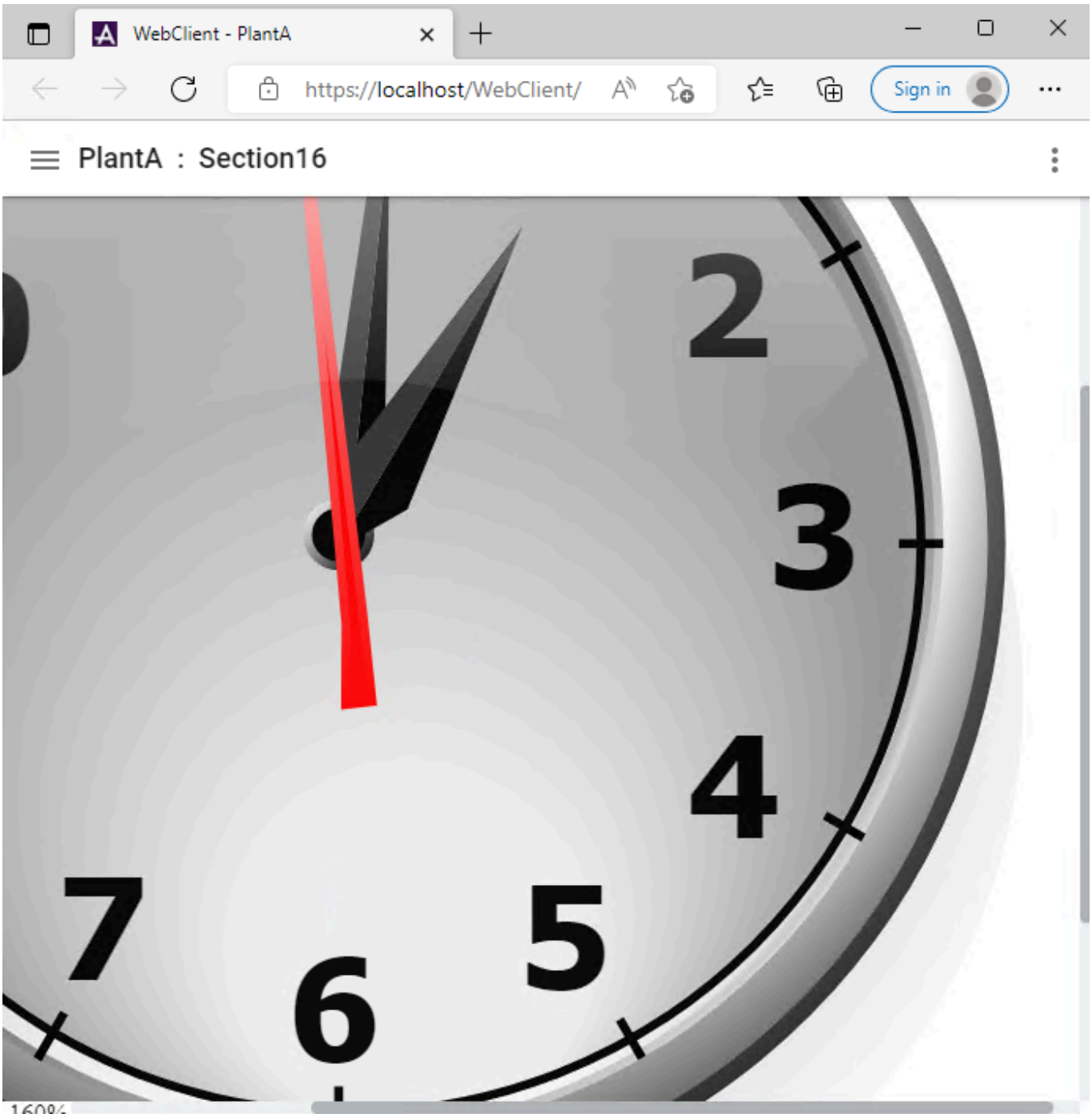


App bar and smaller screens

For very small screens, the icons on the right collapse into a vertical bar. The icons will be hidden to save screen space. You can view Industrial Graphics on any mobile device, and use the pan and zoom gestures to view larger graphics.

Pan and zoom support

InTouch Web Client supports the Pan and Zoom gestures for all supported browsers and devices. When a graphic in the web browser is zoomed in from a non-mobile device, the zoom percentage is displayed in the lower left corner of the horizontal scrollbar. The zoom percentage is limited up to 500%.



Pan and zoom using touch

You can zoom-in and zoom-out on a focused display by using two fingers.

- To zoom-in, place two fingers on the screen and expand. To zoom-out, pinch the display with two fingers.
- Double-tap to restore the display to 100% zoom.

To pan the display, press one finger on the display and move your finger.

Pan and zoom using mouse

You can zoom-in and zoom-out on a focused display by using the mouse's scroll wheel.

- Hold the **Ctrl** key and scroll the wheel up or down to zoom-in or zoom-out. If the mouse pointer is outside of a pane, then zooming will not be allowed.
- Double-click the left mouse button to restore the display to 100% zoom.

To pan a display, keep the center wheel pressed, move the mouse to the area and then release the center wheel.

Zoom using keyboard

You can zoom-in and zoom-out on a focused display by using keyboard.

To zoom-in, use the **Ctrl** and **+** keys together. To zoom-out, use **Ctrl** and **-** keys together.

Zoom by location

You can also zoom in the graphic from the location you select on.

Pan and Zoom toolbars are not supported. Pan and Zoom gestures are not supported for graphics displayed using the ShowSymbol animation or ShowGraphic script, Button, Select and Slider.

Generate the iFrame code block

You can generate the <iframe> block for a symbol directly from the web browser. You can then use this code block in an external website to view Industrial Graphics in run time.

1. Navigate to the graphic.
2. Select the **Share** icon  .
3. In the **Owning Object** field, specify the Owning Object for this graphic.
4. The list of supported custom properties will be displayed. Select a custom property and provide a value.

   All values provided will be verified against the data type. For example, if a custom property requires an integer value, you will not be allowed to enter a character value.
5. Select **Support cross origin** to use the script on any Web Client, either on the local or remote machines.

   The embedded HTML code will change based on the selected options.
6. Select **Copy**.
7. You can now paste the generated code in your website in a .html file.

```
<iFrame src="http://<hostname>/InTouch/?iframe=true&symbol=Sym1?crossOrigin=1"
id="symbol" width="640px" height="480px"
customProperty='[{"n":"CP1","v":"True","t":1},{"n":"CP2","v":"24.5","t":6},{"n":"CP3","v
":"Orange","t":7}]' onload="postMsg()"></iFrame>
<script type="text/javascript">
function postMsg(){
var iframe=document.getElementById('symbol');
var obj={};
```

```
for (var i=0;i<iframe.attributes.length;i++) {
obj[iframe.attributes[i].nodeName]=iframe.attributes[i].nodeValue;
}
var win=iframe.contentWindow;
win.postMessage(obj,obj['src']);
};
</script>
```

If Language = French (France)

```
<iFrame src="http://<hostname>/InTouch/?localeid=1036&iframe=true&symbol=Sym1"
id="symbol" width="640px" height="480px" onload="postMsg()"></iFrame>
<script type="text/javascript">
function postMsg(){
var iframe=document.getElementById('symbol');
var obj={};
for (var i=0;i<iframe.attributes.length;i++) {
obj[iframe.attributes[i].nodeName]=iframe.attributes[i].nodeValue;
}
var win=iframe.contentWindow;
win.postMessage(obj,'*');
};
</script>
```

Supported graphical elements and known limitations

This section describes about the supported graphical elements and known issues.

Known limitations

The InTouch web client will be enhanced with each product release. The following list summarizes the major limitations in the current release.

- InTouch WindowViewer running in an RDS session is not supported for data communication with the Web Server.
- Published applications from System Platform 2017 and earlier versions are not supported by the Web Client. You must migrate the original application to the current version and then republish the application to work with Web Client.
- Frame windows with special (unsupported) characters in the window name are not supported.
- Indirect tags are not supported.
- The PlaySound() script function is not supported.
- For InTouch HMI managed applications, Galaxy based security is not supported.
- System tags will not return user name, and related details per user.
- Setting graphic properties using a script: Only the following properties are supported for reading or writing to a graphic element property using a script:
  - Position (X,Y)
  - Size(Height,Width)
  - Angle

  - StartAngle
  - SweepAngle
- Several symbols in the symbol library incorporate graphic elements or color settings that are not supported by the Web Client in the current release.
- Windows Common Controls (WCC) refer to RadioButton Group, CheckBox, Editbox, ComboBox, Calendar, DateTime Picker, and ListBox.
- The Multi Pens Trend graphic element is not supported.
- Bit State animations are not supported.
- The following Truth Table animations are not supported: Disable, Visibility, Blink, and Value Display.
- The custom properties with History Summary data type are not supported.

The following sections describe these limitations in detail.

Important: When loading a graphic in the Web Client, unsupported features are logged as warnings in the Logger. A generated report will include the warning path.

Properties supported by all supported graphical elements

When viewed in the InTouch web client, the following properties are supported with all supported graphic elements in a symbol:

- Angle
- X
- Y
- Width
- Height
- Enable
- Visible
- AbsoluteAnchor
- RelativeAnchor
- Transparency
- ElementStyle
- OwningObject
- Start
- End
- Radius
- Tension
- Custom Property Overrides

  Note: The ElementStyle property enables users to select a defined element style and apply it to a graphic element. An element style includes options that define color properties like FillColor, LineColor, TextColor and OutlineColor.

Properties supported by all supported graphic elements with some limitations

When viewed in the InTouch web client, the following graphic elements have some rendering limitations:

Graphic Element Property	**Property Limitations**
FillColor	Texture is not supported. Some other limitations on FillColor might apply to individual elements.
UnFilledColor	The same limitations of the FillColor property apply to the UnFilledColor property.
LineColor	Texture is not supported.
Font	Only supports the Font Size, Font Type, and Bold or Regular Font Style options of the Font property.
TextColor	Only the Solid Color option of the TextColor property is supported.
FillOrientation	RelativeToScreen is supported only when Color is set as 'Solid' or 'Fill' or 'UnFill'.
FillBehavior	Will always be set to Both.
Runtime Behavior	TabOrder, TabStop, ZoomPercent are not supported on any graphical element.
Relative references	MyPlatform, MyEngine, MyHost, MyArea, MyContainer and Me are supported. MyViewApp is not supported.
AutoScale	Is not supported.
ShowGraphic	Is supported.
Line	If the Line graphic is configured with the LineWeight property value more than 1px on the rim of the symbol, it will be Truncated.
Group/Embed Graphic	Supports Visibility, Blink, Disable, Push Button, User Input/Value Display, ShowSymbol and ActionScript animation. Changing the 'Treat as icon', and 'Enabled' properties by script is not supported. %FillHorizontal and %FillVertical animations: - Only applies on the group. Not supported for sub elements. - for Button and Image are not supported .

Graphic Element Property	Property Limitations
	- 'Related To Screen' for both group and sub elements are not supported. - when the angle of the sub elements is changed is not supported

Supported graphic elements and additional properties

The graphic elements listed below are the only ones that can be displayed in the InTouch web client. **Any graphic element not listed in this table is not supported in the InTouch web client**.

The table lists the properties supported for each graphic element and any limitations of those properties. When using graphics that incorporate color properties, the colors have the same limitations applicable for FillColor, LineColor, and other related graphic properties.

Note: In addition to the properties listed here, all the properties listed in Properties supported by all supported graphical elements and Properties supported by all supported graphic elements with some limitations can be used with any supported graphic element.

Graphic Elements	Supported Properties	Limitation Notes
Line Polyline Curve 2 Point Arc 3 Point Arc Connector	Line Style LineWeight LinePattern StartCap EndCap	StartCap, EndCap: Supported but there maybe slight differences in rendering.
Button	ButtonStyle Text Word Wrap FillOrientation FillColor Alignment	ButtonStyle: Only Standard button style is supported. Word Wrap: Any caption that exceeds the button width will be truncated. FillOrientation:'RelativeToScreen' is only the Color is set. FillColor: Only Solid color is supported. If you have gradients with multiple colors selected, they will be converted to a single color using the first color. Alignment: Only Centers alignment is supported.
Text	Alignment	Alignment: Only left-top supported.

Graphic Elements	Supported Properties	Limitation Notes
Image	HasTransparentColor TransparentColor ImageStyle	FillColor, FillPercent and unfilledColor are not supported.
Text Box	Text TextFormat WordWrap LineWeight LinePattern Alignment Font	Flip is not supported.
Status	Graphics Expression	Status Style: Only Default configuration is supported.
Web Alarm Client (EAC) (This element is supported only with some HMI/SCADA software.)	Alarm Mode Colors Column Details Queries and Filters Time Settings Run-Time Behavior	**Alarm Mode** • Client Mode: Only supports Current Alarms, Recent Alarms and Events. • Alarm Query: Supports InTouch and application server alarms • Use Default Ack Comment: Only supports comments showing Alarms and Events. **Alarms and Events:** Colors: Shelve, Flash Unack Alarms is not supported. Column Details: Does not support sorting. Queries and Filters: Only supports the 'Default' query. Customized query or filter are not supported. Data Binding: Not supported. Events: Not supported. Time Settings: Not supported. Run-Time Behavior: Only supports Show Heading, Show Grid and allow column resizing.

Graphic Elements	Supported Properties	Limitation Notes
		Only supports user ACK on selected records or ACK on all alarm records.

Graphic Elements	Supported Properties	Limitation Notes
Web Trend Client (This element is supported only with some HMI/SCADA software.)	Pens Appearance Options Historical Sources	**Pens** Only supports Show, Pen Name (As description), Expression or Reference. Pen Details: Min, Max Pen Options: Color Plot type(all will be treated as Line). **Appearance** PlotArea: Single tag mode,Grid (Show vertical grid, show horizontal grid, Color). X time axis:Show cursor(Cursor1:Color), Number of values. Y value axis:Number of values, Value axis label (Multiple Scales,Single Scale). **Options** Only supports Retrieval:Trend Duration **Data Binding**: Not Supported **Event**: Not Supported **Historical Sources**: Connection Timeout in Seconds, Query Timeout in seconds and Use Http are not supported. **Note:** Does not support the Trend Client's methods and properties.
Web SQL Data Grid (This element is supported only with some HMI/SCADA software.)		Data Binding animation Event Animation Custom property • CmdCopy_dg • CmdPaste_dg • CmdWrite_dg • RowsChanged_dg • WriteButtonHide_dg • ColumnAggregateEnable_dg • SupportThemes_dg

Graphic Elements	Supported Properties	Limitation Notes
		- SQLDataGrid control is not supported. - The selectedRowNumber property value cannot be retrieved using custom properties. - As databinding is not supported, the default value of a property cannot be retrieved, if the property is referenced in a custom property and the default value is empty. - The RowCount property cannot display the grouping count due to limitations of the kendo grid. It will always display the total number of records. - When 'Windows' authentication is used in a local database, add the virtual user NT Service\AIGWebServer to allow it to access the database. Accessing a remote database with 'Windows' authentication, and 'SQLServer' mode is not supported. - SQLGrid is not supported when used in a carousel widget and viewed in WindowViewer. - When groups are applied to the SQL Grid, with more than 2000 records and 5 columns, some performance issues may be seen with scrolling and refreshing of the rows.
Trend Pen (This element is supported only with some HMI/SCADA software.)	Data Source	Trend time Period: Does not support Fixed Type.

Graphic Elements	Supported Properties	Limitation Notes
Check Box		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported. Checked: Does not support Checked property.
Radio Button Group	Static Values and Captions States	Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported.
Edit Box		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported.
Combo Box		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported. Type: Does not support MaximumLength
Calendar		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported. Bold Dates: Is not supported ShowToday: Will always display, even if checked out. Calendar Colors: Is not supported
DateTime Picker		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported. Configuration : Is not supported. Calendar DropDown Colors: Not supported.
ListBox		Selected Value Changes: Irrespective of user settings, only values submitted immediately are supported.

Graphic Elements	Supported Properties	Limitation Notes
ConnectorPoint		Moving the ConnectorPoint is not supported.

Supported animations

The following table lists animations supported at runtime when viewing a graphic in the InTouch web client. When using animations that incorporate color properties, the selected animation colors have the same limitations as FillColor, LineColor, and other related graphic element properties. Also, viewing in a browser does not support writing values. Any type of animation that updates attribute value is not supported.

Note: The following table lists all supported animations. **Any animation not listed is not supported when viewing in the InTouch web client**.

Animation	Limitations
Alarm Border	Alarm Border animation shows an alarm border around a graphic element whose visible appearance represents the graphic element's alarm type and acknowledgement status. The appearance of each alarm border is defined by an associated element style. Not all element style properties are supported. For instance, if Gradient is specified as the value of an element style's Line Color Override option in an alarm border element style, it will not be supported.
Element Style	Refer to the graphic elements limitations regarding color, line, and font. The same limitations applicable to graphic elements also apply to Element Style animation. Expression Or Reference: Time and Elapsed Time are not supported. Changes to an element style definition become effective only after redeploying or publishing the application. Does not support Windows Common Controls.
Fill Style Line Style Text Style	Refer to graphic element limitations regarding color, line, and font. The same limitations applicable to graphic elements also apply to Element Style animation. Expression Or Reference: Time and Elapsed Time states are not supported. Does not support Windows Common Controls.

Animation	Limitations
	TextColor: Only Solid Color is supported.
% Fill Horizontal % Fill Vertical	Only RelativeToGraphic is supported.
Slider Horizontal Slider Vertical	A slider cannot write data to tags, but can support data binding with a custom property. Slider only supports On Mouse Release when writing to a custom property. Slider does not support Cursor anchor. Slider does not support Show Tooltip.
Width Height	Does not support Windows Common Controls.
Orientation	The Relative Anchor or RelativeOrigin are not supported when setting Orientation animation. (Only supported anchor at center).
Disable	The RadioButton graphic element does not support Disable animation.
Point	Point animation on Curve and ClosedCurve are not supported.
Tooltip	Tooltip animation is supported except for Image, Radio Button, Button, and Text graphic elements. Tooltip animation is not supported on Group or embedded symbol. Does not support Windows Common Controls.
Action Scripts	Action scripts are supported except for functions that invoke dialogs. The ShowGraphic animation is only supported for a Frame window.
Show Symbol	Title: Only supports the default option. Type: Supports Modal and Modeless where the Close button option is checked. Position: Only supports the center where the Client

Animation	Limitations
	Area x,y value is 0,0. Size: Only supports Relative To symbol option. Shortcut: Is not supported.
ShowGraphic	Title: Only supports the default option. Type: Support Modal and Modeless with the Close Button option checked. Position: Only supports center, Client Area and x,y value is 0,0. Size: Only supports the Relative To Symbol option. Shortcut: Is not supported.
Hyperlink	Does not support on Windows Common Controls, button, text, and textbox.

Web Client mobile application

Operating system requirements

Download the Mobile Application from the Apple App Store or Google Play Store. Search for AVEVA Mobile Operations and select **Install**.

Minimum Android version: Android 6.0 (API Level 23 – Marshmallow)

Target Android version: Android 9.0 (API Level 28 – Pie)

Minimum iOS version: iOS 9.0 or later

For device support, see Browser and Mobile Support.

Log in to mobile application

On installing the app, provide user credentials and login to the app to view the graphics.

1. Launch the mobile app.
2. Enter the **server name** or **IP address** of the node where the application is running.
3. Select or enter the **Website** name. The website name provided should match the *Custom URL* specified in InTouch HMI Application Manager.
4. Enter the **user name** and **password**. Provide the domain name for the user name if required.
5. Select the **Full Screen Mode** checkbox to view the app in a full screen mode.

   This setting is useful for displaying full sized graphics for TV or larger displays, where the navigation bar and header bar are hidden by default.
6. Select **Login**.

On successful login, the Web Client Home page appears.

The user credentials are required for the first time. The application will remember the user credentials on subsequent uses. Use the **Clear** button to clear the details from the login screen. Only users that are part of the Web Client user groups can access the Mobile application. Use the log off button to exit the app.

Note: The Web Client mobile app supports anonymous login as well as login through AVEVA Identity Manager (AIM).

Use the InTouch Web Client mobile application

If your credentials are authenticated and a valid Web Server license is acquired, the application is displayed. The Home Symbol, if configured will be displayed by default.

1. Select the  icon.
   The overlay on the left displays the hierarchical structure of the graphics within the running application.
2. Select a folder name.
   The graphics within the folder are displayed.
3. Select a graphic.
   The selected graphic is displayed.
   You can also Pan and Zoom to view the graphics at a different scale.

The InTouch web client mobile app supports runtime language switching.

Switch a language at runtime

You can develop applications that can be switched to another language at runtime.

To enable run-time language switching, you must complete the following tasks:

- Configure multiple languages for the application.
- Export your application text for offline translation.
- Translate one or more exported dictionary files.
- Import one or more translated dictionary files.

About switching a language at runtime

You can develop applications that can be switched to another language at runtime.

To enable run-time language switching, you must complete the following tasks:

- Configure multiple languages for the application.
- Export your application text for offline translation.
- Translate one or more exported dictionary files.

- Import one or more translated dictionary files.

Configuring languages for runtime language switching

Every InTouch application is associated with a base language used to develop the application. You must configure any additional languages that you want to support.

Note: If you are using language switching in combination with Network Application Development (NAD), we recommend that you set the change mode to "Restart WindowViewer" or "Prompt user to restart WindowViewer" instead of "Load changes into WindowViewer" or "Prompt user to load changes into WindowViewer" for the NAD client node.

To configure languages for run-time language switching

1. In WindowMaker, open the application for which you want to configure languages.
2. On the **File** menu, in the **Configure** group, select **Languages**. The **Languages** window appears.

   The **Configure Languages** dialog box shows the base language of the application.
3. Select **Add**. The **Add Language** dialog box appears.

4. Specify the language and font settings. Configuring the font settings defines the default font properties of your translated text.
   - In the **By Name** or the **Locale ID** list, select the language to add. If you select the language by name, the corresponding locale ID appears in the **Locale ID** list, and vice versa.
   - Select **Font**. The **Font** dialog box opens. Configure the font and then select **OK**.
5. Select **OK** to close the **Add Language** dialog box. The language you configured is listed in the **Configure Languages** dialog box.
6. To add more languages, repeat steps 3 through 5.
7. When you are done, select **Close**.

To remove a language:

1. Select the language you want to remove from the **Configure Languages** dialog box.
2. Select **Remove**.

   The **Confirm Delete** dialog box appears to verify that you want to remove the language from the application.
3. Select **Yes**.

   The **Configure Languages** dialog box refreshes and shows the language has been deleted.

To set a default language:

1. Select the language you want to set as the default for the application on the **Configure Languages** dialog box.
2. Select **Set Default**.

   The **Configure Languages** dialog box refreshes and shows the default language of the application in the bottom left corner.

Change the font settings for a configured language

The default font for all languages is Tahoma. The font style and size depends on the corresponding settings for the individual phrases in the base language. You can change the font setting for a language that you have already configured. Because of the differences between the textual display of different languages, you can specify an appropriate font to ensure that your translated text fits correctly on buttons and other objects.

Change the font settings for a configured language

1. In WindowMaker, open the application for which you want to change font settings for a configured language.
2. On the **File** menu, point to **Configure**, and then select **Languages**.

   The **Languages** screen appears.

3. Under the **General** tab, In the list of languages, select the target language.
4. Double-click the **Font Name** and **Font Size** entries to edit the values.
5. Select or clear the **Bold** and **Italic** check-boxes as required.
6. Make your changes and then select **OK**.

Add runtime language switching functionality

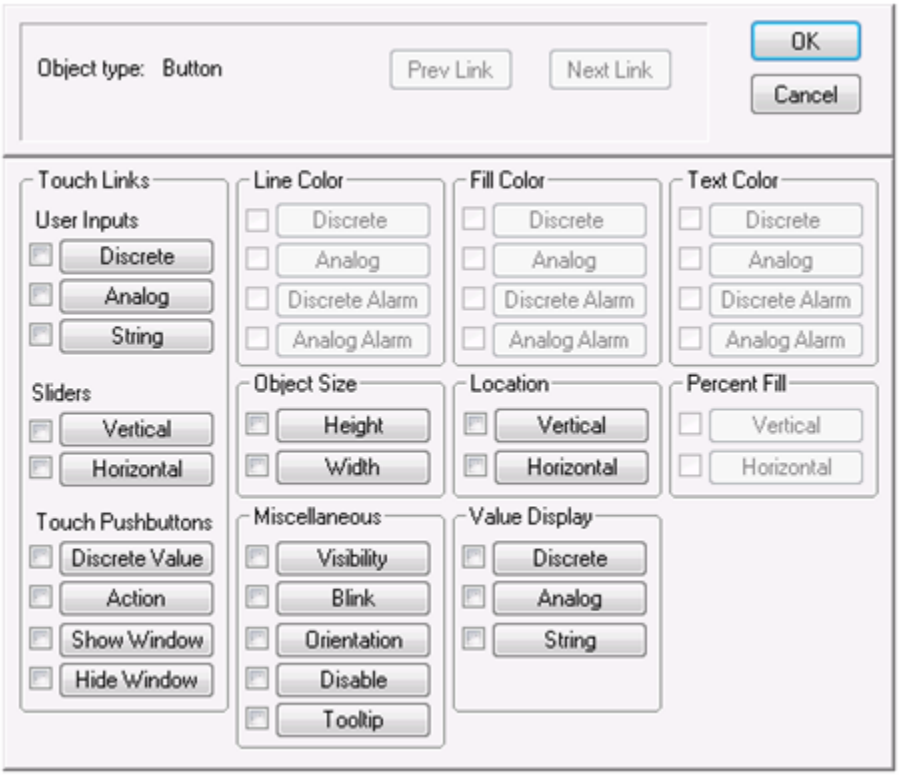
Runtime users can switch the language of an application interface by using the WindowViewer **Language** command on the **Special** menu.

You can also add a button to your application to allow run-time users to switch the language. Before you start, make sure that you have configured the additional language for the application and that you know the locale ID for the language. For more information on configuring languages for an application, see Configuring languages for runtime language switching.

Add a button for switching languages at run time

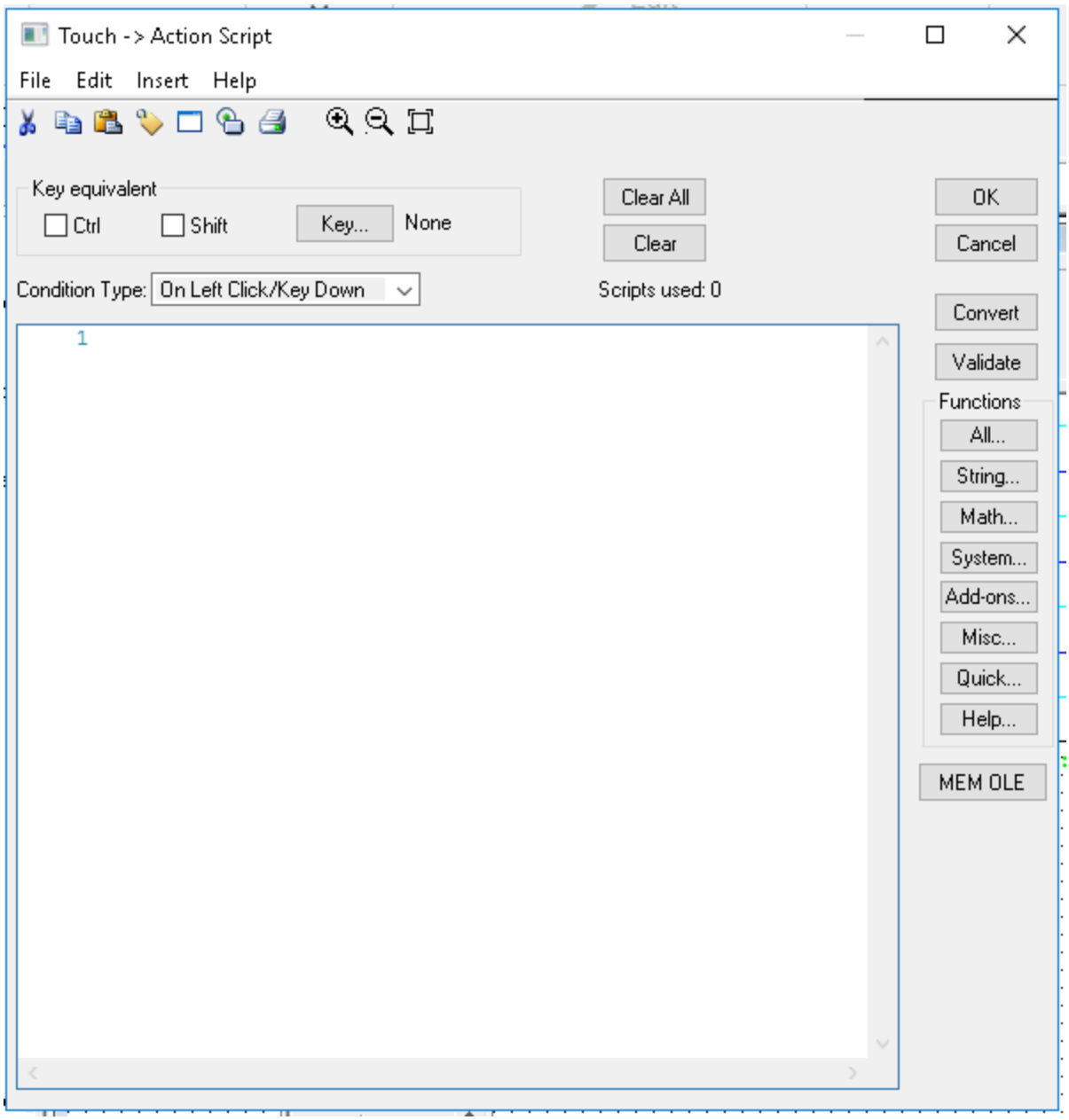
1. In WindowMaker, open the application window that you want to add the language switching button.
2. Draw a button, and assign a text label to the button that indicates the language to be switched to when selected.
3. Double-click the button.

   The animation selection dialog box appears.

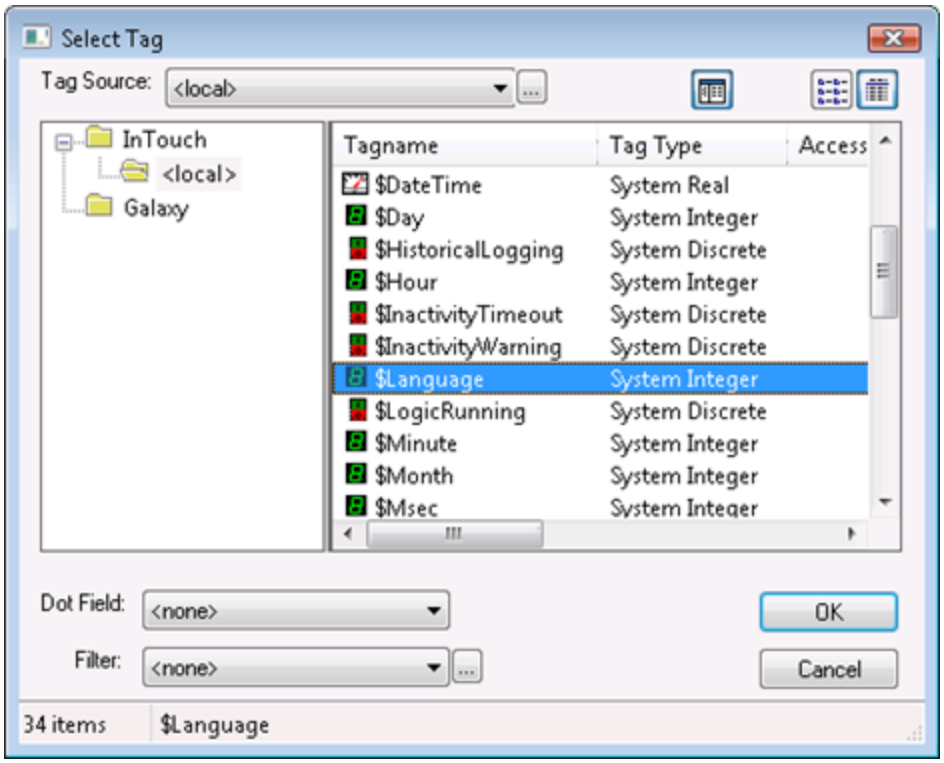


4. In the **Touch Pushbuttons** area, select **Action**.

   The **Touch -> Action Script** dialog box appears.

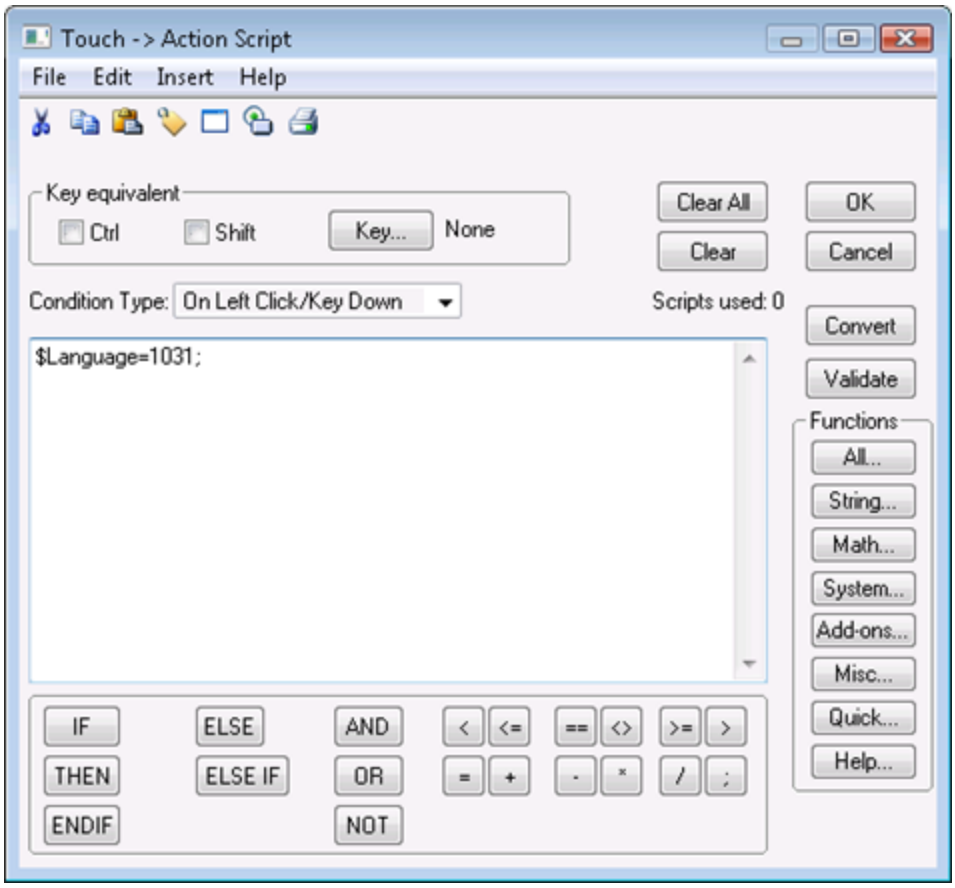


5. Double-click anywhere in the script area of the **Touch -> Action Script** dialog box.
   The **Select Tag** dialog box appears.



6. Select the **$Language** system tag and then select **OK**.

   Set the **$Language** system tag equal to the locale ID of the language you are assigning to the button and select **OK**.



Note: You can also use the script function SwitchDisplayLanguage(LocaleID) instead of the $Language tag.

7. Select **OK** to close the dialog box.

SwitchDisplayLanguage() function

Switches the display of visible, static texts and alarm fields in a desired language for which translated strings are provided.

Category

misc

Syntax

```
SwitchDisplayLanguage(LocaleID);
```

Parameter

LocaleID

The language in which static text strings and alarm fields are to be shown at run time.

Example(s)

In this example, German is the language to be shown at run time.

```
SwitchDisplayLanguage(1031);
```

See also

$Language system tag

$Language system tag

If multiple languages are defined for an InTouch application, the $Language system tag reflects the value of the Language ID for the currently shown language. By default this is the language ID (locale ID) of the base InTouch system (E/F/G/J/SC). Setting this to another ID switches strings and alarm fields with defined values in the new language.

Note: The $Language tag is configured as a local tag to allow independent language switching in the Web Client. Multiple user sessions can view the web client in different languages. The change in language in the web client will not affect the language selected in WindowViewer and vice versa.

Category

system

Data Type

Integer (read / write)

Export application text for offline translation

If your InTouch application has many strings, you typically send the text strings out for bulk translation. You can export your application's strings for translation and keep them organized using a text editor, an XML editor, or a spreadsheet program like Microsoft Excel.

You can export static text from the following:

- Text objects
- Button text
- Text inside SmartSymbols
- Tooltip static text
- User messages
- On/off messages inside input links
- On/off messages in output links
- Text on wizards

You cannot export the dictionary until you close all windows in WindowMaker. If you make changes to your application after you export your dictionary files, you must export the dictionary file again. For more information, see Export text to an existing dictionary file.

You can only export the text strings for one language at a time. By default, the InTouch HMI opens the My InTouch Applications folder. If you choose any other folder, the InTouch HMI then defaults to that path. Creating a new folder to export phrases for each language makes it easy to manage dictionary files. For example, ...\My InTouch Applications\My German Files\.

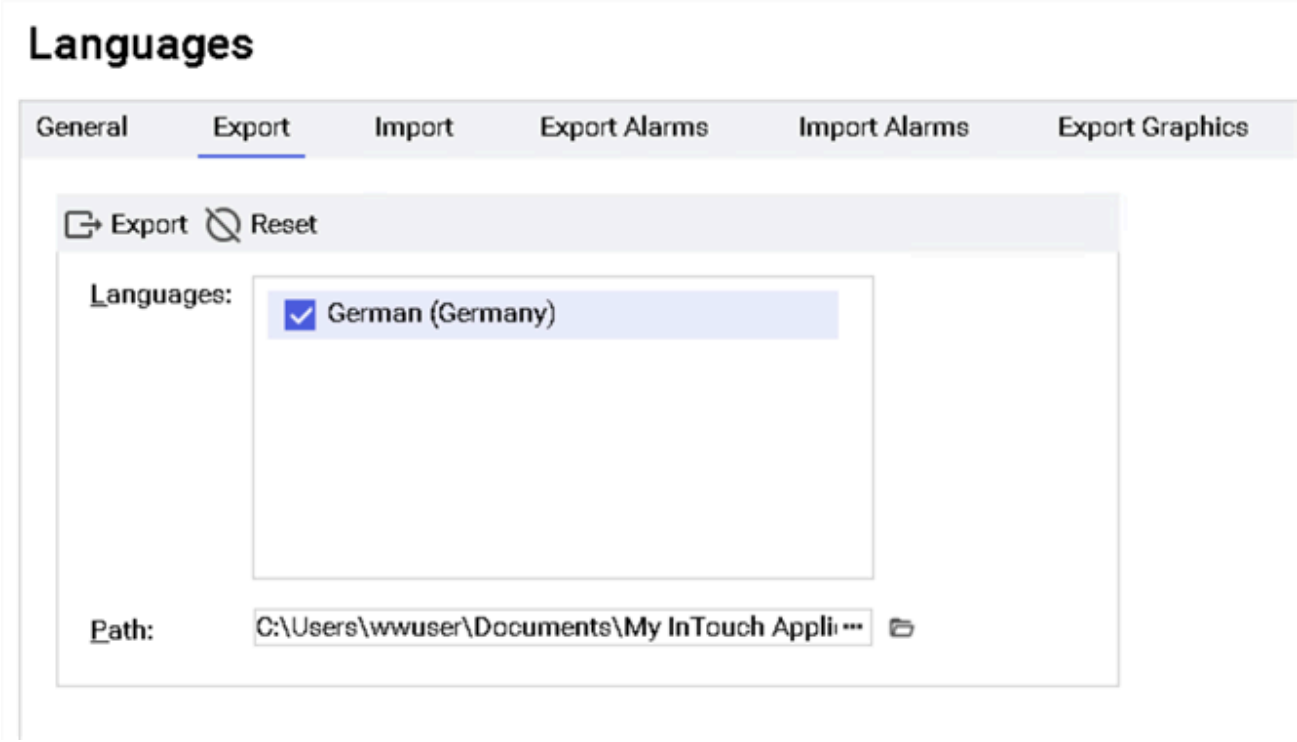
The InTouch HMI creates a dictionary file for your application and a separate dictionary file for each SmartSymbol within the application. The application dictionary name has a format of application name_localeID whereas SmartSymbol dictionary files have a format of SSD_Name of the Symbol_localeID_GUID.

When you export the dictionary for an application, the file is an .xml file that you can edit using Microsoft Excel 2003 or later.

Export application text for offline translation

1. Start WindowMaker and open the application for which you want export text strings for offline translation.
2. On the **File** menu, point to **Configure**, and then select **Languages**.

   The **Languages** screen appears.
3. Select the **Export** tab. The **Export Dictionary** dialog box appears.



4. Configure the export settings.
   - In the **Defined Languages** list, select the language dictionary to export.
   - In the **Path** box, type the folder to which you want to export the dictionary. Select **Browse** to select an existing folder or create a new folder.
5. Select **Export**. The export progress is shown. If the export is successful, the **Export Successful** dialog box appears.
6. Select **Close** to return to the WindowMaker window or select **Close and Launch Explorer** to open the folder

containing the dictionary files.

Export text to an existing dictionary file

After you export your application text for offline translation, you might need to make changes to your application. If you change the application, you need to export the text again. For more information, see Export application text for offline translation.

If you export more than one time to the same directory, the **Confirm File Replace** message appears.

If you select **Yes**, the existing .xml files are updated with any new strings and language information added since you exported last. If the existing dictionary file contains translations for any phrases and you imported it to the InTouch HMI previously, those translations are preserved. If you deleted any phrases from the application since the last export, they are removed from the dictionary file.

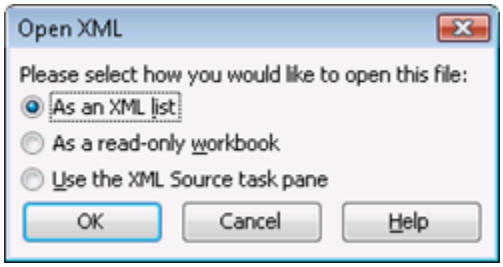
Translate an exported dictionary file

After you export the dictionary file containing your application text, use Microsoft Excel 2003 or later to edit the text.

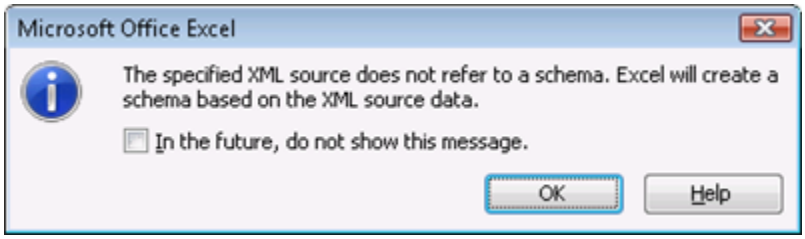
The InTouch HMI creates a separate dictionary file for each language that you export. The InTouch HMI also creates separate dictionary files for each SmartSymbol in your application. Be sure to translate all dictionary files for all languages and SmartSymbols.

To translate an exported dictionary file

1. Open the XML file in Excel. The **Open XML** dialog box appears.



2. Select **As an XML list**, then select **OK**. A message may appear.

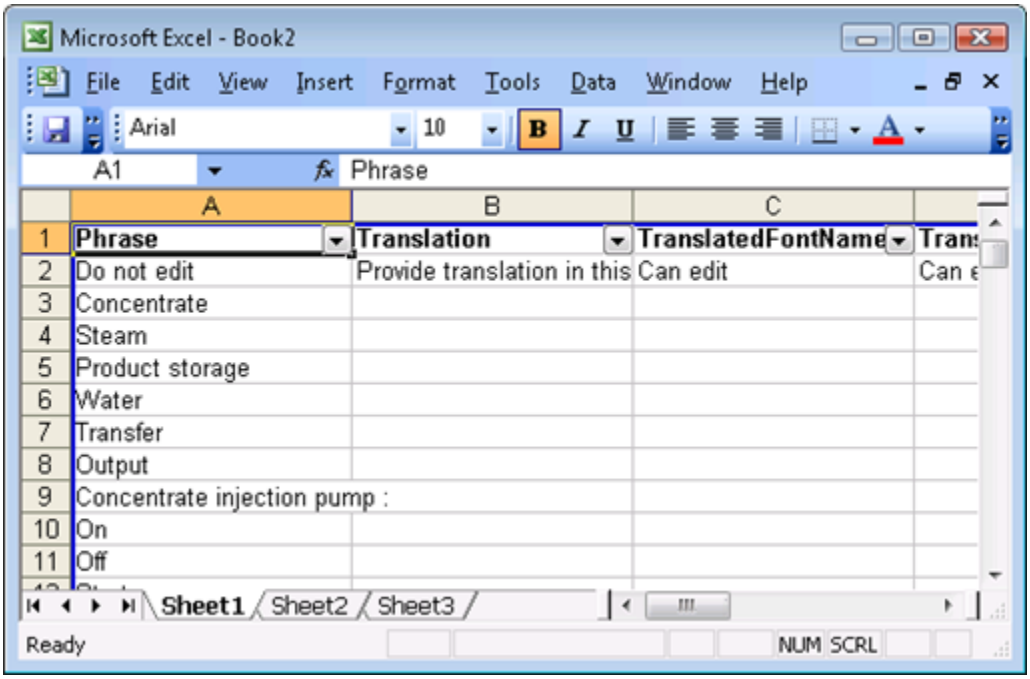


3. Select **OK**.

   The XML file opens in Excel with columns for the:

   - Phrases in your application
   - Translated phrases from the translator
   - Translated font name

- Translated font properties
- Translated font size
- Base font properties
- Base font size
- Context, phrase ID, language ID and foreign language ID



Important: Only modify data in the **Translation**, **TranslatedFontSize**, **TranslatedFontName**, and **TranslatedFontProperty** columns. Do not change any column header. Do not insert or delete rows.

4. Type the language-specific text in the **Translation** column in the row that corresponds with the base language string in the **Phrase** column.
5. If necessary, change the font parameters for the translated strings to fit the text in the space allowed in WindowViewer.
   - In **TranslatedFontName** column, type the font name.
   - In the **TranslatedFontProperty** column, type the notation for the font properties:

     B = **bold**

     I = *italic*

     U = underline

     For example, if you want to the text to be bold, type **B** in the **TranslatedFontProperty** column. If you want the text to be bold and underlined, type **BU** in the **TranslatedFontProperty** column.
6. Save the file using XML Data as the file type.

Important: If you save as another file type, such as XML Spreadsheet, Excel changes the schema and the InTouch HMI cannot load the file. If you change the name of the XML file, run-time language switching does not work.

Import translated dictionary files

The InTouch HMI creates a dictionary file for each language that you export. The InTouch HMI also creates separate dictionary files for each SmartSymbol in your application. After translation, you must import the dictionary files for each language to enable run-time language switching for those languages. All dictionary files for a given language should be placed in the same folder.

Import a translated dictionary file

1. Start WindowMaker and open the application to import translated dictionary files into.
2. On the **File** menu, point to **Configure**, and then select **Languages**.
3. In the **Languages** screen, select the **Import** tab.

4. Configure the import settings.
   - In the **Defined Languages** list, select the language dictionary to import.
   - In the **Path** box, type the path to the dictionary file to import. Select **Browse** to browse and select the file.
5. Select **Import**.
6. If you are re-importing a SmartSymbol dictionary file, you are prompted to replace the existing file.

   If the import is successful, the **Import Successful** dialog box appears.

Language switching at runtime for alarms

As part of the setup for run-time language switching, you can also localize the alarm group names, alarm comments, alarm fields, internal status messages. In addition to switching the run-time language of text strings, you can also configure run-time language switching of alarm comments, alarm states, alarm types, and alarm classes in the Alarm Viewer and Alarm DB View controls.

You can also script, query, or filter using the translated alarm group names.

Export alarm comments for translation

You can export alarm comments for translation.

You export the Alarm State, Alarm Type, and Alarm Class fields for:

- All tags with an alarm comment.
- All tags with a tag comment.
- System tags so you can localize comments shown in clients when events are raised by system tags.

Understand two-character application IDs

When you export alarm and tag comments for localization, you must specify a two-character application ID. The ID is used internally by the system to distinguish between alarms generated by applications having the same name.

Because a tag can contain both a tag comment and an alarm comment, 1 and 2 are added after the two-character application ID to differentiate between these two fields. Tag comments have a 1 between the ID and the tag name. Alarm comments have a 2 between the ID and the tag name. For example, AA1TankLevel is a tag comment, and AA2TankLevel is an alarm comment.

If you export an application, the application ID information is removed.

If the alarm database contains old data without a two-character application ID and new records are prefixed with an ID, then alarm comment queries in the Alarm DB View control do not work with the following operators: <, <=, >, and >=.

Export alarm comments

You can export alarm comments for translation.

Caution: Before exporting alarm and tag comments, back up any files in your target directory in case of possible data corruption or errors.

Export alarm comments for offline translation

1. Start WindowMaker and open the application for which you want export alarm comments for offline translation.
2. On the **File** menu, point to **Configure**, and then select **Language**.

   The **Language** screen appears.
3. In the **Language** screen, select the **Export Alarms** tab.

4. In the **Path** box, type the folder to which you want to export the dictionary. Select **Browse** to select an existing folder or create a new folder.
5. In the **Two characters representing Unique application** box, type the two characters. The ID can only contain alphanumeric characters and it is case-sensitive.

   Caution: If you previously exported alarm or tag comments from this application, you must use the same two-character application ID when you export them the next time. If you enter a new two-character application ID, the InTouch HMI regenerates the IDs for all the alarms and tags, which causes all existing translations to be lost.
6. Select **Export** to export the information to an XML dictionary file.

   The InTouch HMI creates an individual export file for each configured language. All the dictionary files for different languages are exported to the single directory you specify.

   If a duplicate file exists for any language being exported, you are prompted with the name of the file. You can cancel the export or continue the export operation.

   If the export is successful, the **Export Successful** dialog box appears.

   Note: If the size of the alarm comment configured in the tag dictionary is greater than 127 characters or the tag comment is greater than 46 characters, the alarm or tag comment is not exported. You are notified that the comment was not exported to the dictionary file at the end of the export process and an AlarmComment.log or TagComment.log file is created in the export directory.
7. Select **Close** to return to the WindowMaker window or select **Close and Launch Explorer** to open the folder containing the dictionary file.

Export to an existing alarm comment file

After you export your alarm and tag comments for offline translation, you may need to make changes to your application that require you to export alarm and tag comments again. For more information, see Export alarm comments for translation.

If you export more than one time to the same directory, the **Confirm File Replace** dialog box appears.

Select **Yes** to update the existing dictionary files with any new strings and language information added since you exported last. If the existing dictionary file contains translations for any phrases and you imported it to InTouch previously, those translations are preserved. If you deleted any phrases from the application since the last export, they are removed from the dictionary file.

Select **Yes to All** to update existing dictionary files for all languages configured in the InTouch HMI.

Select **No** or **No to All** to prevent overwriting the existing file or the existing files for all languages, respectively.

The existing translations for any alarm comments, alarm fields and tag comments are preserved if they are exported again.

Edit the dictionary file

After creating the dictionary file, you need to edit the strings.

The name of the dictionary file is created from the two-character application ID and the language being exported. For example, if the configured language is Chinese(PRC)-2052 and two-character application ID is **AA**, the resulting file name is **AA_2052_ AlarmComment.xml**. The file is written using the same XML schema used by the run-time language switching files.

The general structure of the dictionary file is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfAlarmCommentPhraseItem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <AlarmCommentPhraseItem Phrase="Do not edit">
      <Translation>Provide translation in this column</Translation>
      <Context>Do not edit</Context>
      <PhraseID>0</PhraseID>
      <LanguageID>0</LanguageID>
      <ForeignLanguageID>0</ForeignLanguageID>
    </AlarmCommentPhraseItem>
  - <AlarmCommentPhraseItem Phrase="System">
      <Translation />
      <Context>$System : System Tag Comment</Context>
      <PhraseID>AA1$System</PhraseID>
      <LanguageID>1033</LanguageID>
      <ForeignLanguageID>2052</ForeignLanguageID>
    </AlarmCommentPhraseItem>
</ArrayOfAlarmCommentPhraseItem>
```

Enter the translation for the translation strings. Do not change any of the other information.

You can override some of the Alarm State, Alarm Type, and Alarm Class values. The maximum allowed length for these values is 50 characters.

The following Alarm State values can be overridden for InTouch generated alarms:

Value to override	Default string to be shown
UNACK_RTN	UNACK_RTN
ACK_RTN	ACK_RTN
UNACK_ALM	UNACK_ALM
ACK_ALM	ACK_ALM

The following Alarm Type values can be overridden for InTouch generated alarms:

Value to override	Default string to be shown
SPC	SPC
HIHI	HIHI
HI	HI
LO	LO
LOLO	LOLO
MINDEV	MINDEV
MAJDEV	MAJDEV
ROC	ROC
DSC	DSC
OPR	OPR
LGC	LGC
DDE	DDE
SYST	SYST
USER	USER
PRO	PRO
LOGON_FAILED	LOGON_FAILED

The following Alarm Class values can be overridden for InTouch generated alarms:

Value to override	Default string to be shown
DEV	DEV
ROC	ROC

Value to override	Default string to be shown
DSC	DSC
EVENT	EVENT
VALUE	VALUE

Import translated alarm comments

After translating the strings, you must import the dictionary files for each language to enable run-time language switching for those languages.

After you import the translated alarm comment dictionary files, they are copied into the respective language folders inside the application directory.

Any translated alarm comments for an existing application are overwritten with the contents of the imported files and the application version ($AppVersion) increments by 1.

To import multiple dictionary files from other nodes to support localization of alarm fields from multiple nodes, copy the translated dictionary files from the other nodes into a single directory. Select this directory as the import path. Multiple dictionary files are imported on a single import operation. The InTouch HMI automatically creates the file path based on the language being imported.

Import a translated alarm comment file

1. Start WindowMaker and open the application to import translated dictionary files into.
2. On the **File** menu, select **Configure**, and then select **Languages**.

   The **Languages** dialog box appears.
3. In the Languages screen, select the **Import Alarm** tab.

   Import-Localization-AlarmMessages
4. In the **Path** box, type the path to the dictionary file to import or select **Browse** to browse and select the file.
5. Select **Import**. If there is no translation done in the dictionary file, a dialog box appears saying that there are no translated dictionary files to import.
6. Select **OK**. If the import is successful, the **Import Successful** dialog box appears.

Export alarm group names for translation

You can export alarm group name for translation.

Caution: Before exporting alarm and tag comments, back up any files in your target directory in case of possible data corruption or errors.

Export alarm comments for offline translation

1. Start WindowMaker and open the application for which you want export alarm comments for offline translation.
2. On the **File** menu, point to **Export**.
3. Select **Localization** and then, select **Alarm messages.**

4. The **Export alarm fields** screen appears
5. In the Select language to export list, select the language to export for translation.
6. In the Select directory box, select a location to export the file.
7. In the **Two characters representing Unique application** box, type the two characters. The ID can only contain alphanumeric characters and it is case-sensitive.

   Caution: If you previously exported alarm or tag comments from this application, you must use the same two-character application ID when you export them the next time. If you enter a new two-character application ID, the InTouch HMI regenerates the IDs for all the alarms and tags, which causes all existing translations to be lost.

8. Select **Export** to export the information to an XML dictionary file.

   The InTouch HMI creates an individual export file for each configured language. All the dictionary files for different languages are exported to the single directory you specify.

   If a duplicate file exists for any language being exported, you are prompted with the name of the file. You can cancel the export or continue the export operation.

   If the export is successful, the **Export Successful** dialog box appears.

   Note: If the size of the alarm comment configured in the tag dictionary is greater than 127 characters or the tag comment is greater than 46 characters, the alarm or tag comment is not exported. You are notified that the comment was not exported to the dictionary file at the end of the export process and an AlarmComment.log or TagComment.log file is created in the export directory.

9. Select **Close** to return to the WindowMaker window or select **Close and Launch Explorer** to open the folder containing the dictionary file.

Import translated alarm group names

After translating the strings, you must import the dictionary files for each language to enable run-time language switching for those languages.

After you import the translated alarm comment dictionary files, they are copied into the respective language folders inside the application directory.

Any translated alarm comments for an existing application are overwritten with the contents of the imported files and the application version ($AppVersion) increments by 1.

To import multiple dictionary files from other nodes to support localization of alarm fields from multiple nodes, copy the translated dictionary files from the other nodes into a single directory. Select this directory as the import path. Multiple dictionary files are imported on a single import operation. The InTouch HMI automatically creates the file path based on the language being imported.

Import a translated alarm group file

1. Start WindowMaker and open the application to import translated dictionary files into.
2. On the **File** menu, select **Import.**
3. Select Localization, and then select **Alarm messages**.

   The **Import alarm fields** dialog box appears.
4. In the Select the language to import list, select the required language.
5. In the **Path** box, type the path to the dictionary file to import or select **Browse** to browse and select the file.
6. Select **Import**. If there is no translation done in the dictionary file, a dialog box appears saying that there are

no translated dictionary files to import.

7. Select **OK**. If the import is successful, the **Import Successful** dialog box appears.

Test the language switching functionality at runtime

After you enable run-time language switching in your application, test the language switching functionality. Language switching of alarm and tag comments and alarm fields can be viewed only in the Alarm Viewer and Alarm DB View controls.

As you work with localized alarm and tag comments, be aware of the following:

- If the alarm or tag comment hasn't been translated to the language specified by $Language, the default comment appears in the alarm client.
- If an Alarm Viewer control is querying from multiple providers, the alarm comment, tag comment, and alarm fields from remote nodes also appear translated if the application has the translated dictionary files of the remote node applications.
- If you acknowledge an alarm and provide a comment, this comment appears in the alarm client instead of the localized alarm comment.
- When an Alarm Viewer control is in freeze mode, then the language does not appear switched even though you switched the language. The moment you unfreeze the control, the control is updated with the translated strings.
- The localization of the Alarm Viewer control is only for the display of the control. All script functions still return the default strings even though the language is switched.
- The Alarm DB Logger only stores the data default language strings in the database. The localized strings are not stored in the database.
- The unique IDs for the alarm fields such as EVENT and ACK, are predefined and have the same ID across multiple dictionary files in different nodes. Alarm clients pick the translation from the first loaded dictionary file and the translations from other dictionary files are ignored. Ideally, the alarm fields in all dictionary files should have the same translation in a language. Multiple alarm clients (Alarm DB View and Alarm Viewer controls) use the same translation for the same alarm state for a given language.
- Translated text is truncated to 131 characters for alarm comments and to 160 characters for tag comments.

Test the language switching functionality

1. Open the application in WindowViewer.
2. On the **File** menu, point to **Configure**, and then select **Language** and then select the name of the language to switch to.

   The information from the corresponding translated dictionary file (if one exists) loads and appears.
3. If you added a button to switch the language, select the button to test the script.

Distribute localized files to network application development clients

The files containing the localized alarm comments, tag comments, and alarm fields are distributed to Network Application Development (NAD) clients as part of the InTouch application. When you receive updated files containing alarm comments, you must restart WindowViewer before the translated alarm comments can be

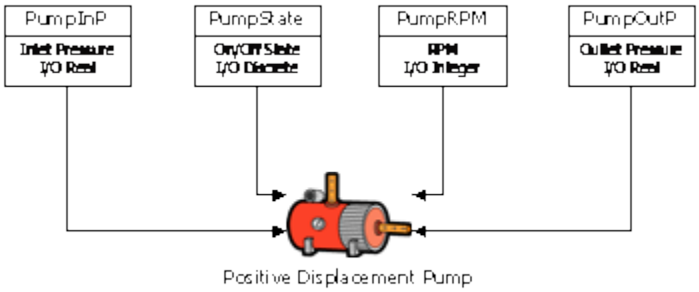
seen in the supported alarm clients.

If you are using language switching in combination with Network Application Development (NAD), set the change mode to "Restart WindowViewer" or "Prompt user to restart WindowViewer" for the NAD client node.

Tags

An InTouch Human Machine Interface (HMI) application is a graphical representation of the components in a manufacturing environment. Plant operators work with this graphical interface to monitor and administer their manufacturing processes.

The figure below shows an example of a pump that is a component of a manufacturing process. The pump has properties with associated values. Pressure, RPM, and status are pump properties whose values are monitored through an HMI.



A *tag* represents a data item in an InTouch HMI application. You use tags to make specific component properties accessible as data items from a manufacturing environment. In the figure above, the PumpState tag indicates whether the pump is on or off. You create tags for components in your manufacturing environment whose properties you want to monitor or control in your InTouch application.

You can use different types of tags for the different types of data collected from a manufacturing component. For example, the PumpState tag returns a Boolean On/Off value to indicate if the pump is running or stopped. You assign the appropriate type of InTouch tag for the type of data that you want to be part of your application.

Get started with tag viewer

Tag Viewer is an external application that allows you to watch and monitor tags and modify tag values at run time. It provides you with a list of all available tags in the application arranged hierarchically, based on their alarm groups. You can run Tag Viewer only when WindowViewer is running. Tag Viewer displays only the tags available in the local InTouch application and does not support remote references.

To use Tag Viewer, you must first enable it in WindowMaker. You can then launch the application at run time. You can open only one instance of Tag Viewer at a time.

Enable Tag Viewer

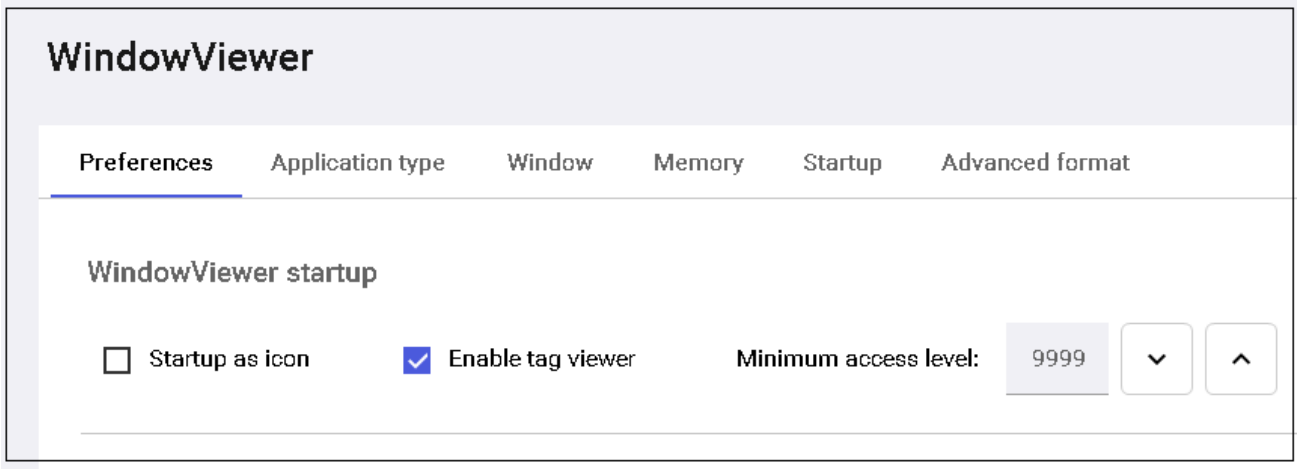
You can configure the WindowViewer properties at design time to enable Tag Viewer to run. You can also configure the WindowViewer menu so that the Tag Viewer option is displayed in the **Special** menu.

After you modify these properties, you must restart WindowViewer for your changes to become effective, if WindowViewer is already open.

Enable Tag Viewer

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **WindowViewer**.

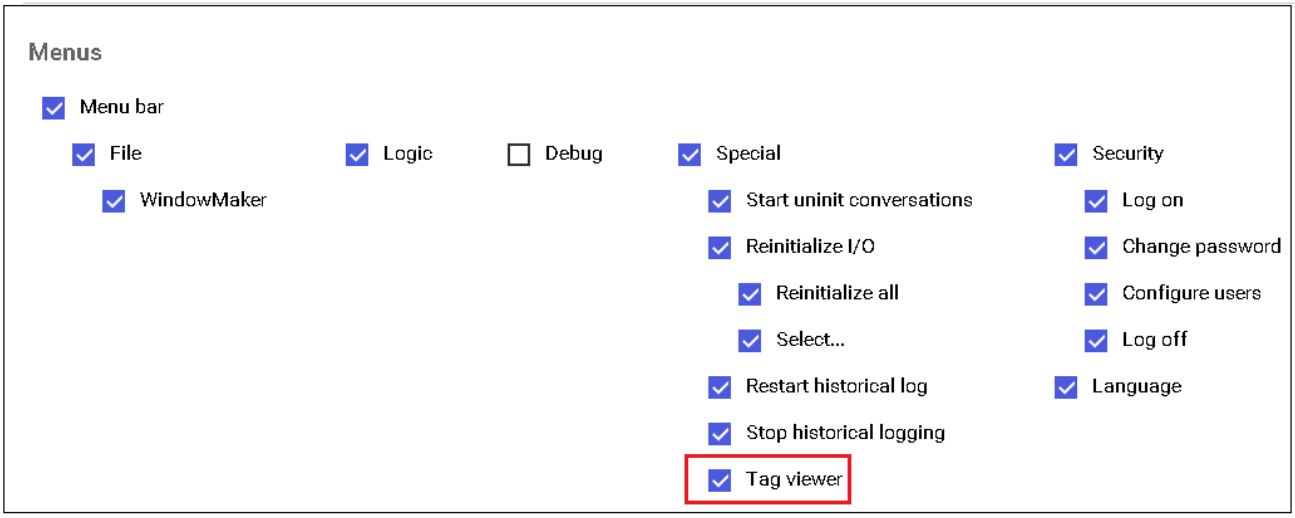
   The **WindowViewer** configuration screen appears.
3. Select the **Preferences** tab.



4. In the **WindowViewer Startup** area, select the **Enable Tag Viewer** check box so that you can start the Tag Viewer. This check box is not selected by default.
5. In the **Minimum Access Level** box, configure the security access level required to run the Tag Viewer application. The default value is 9999, and you can type any value between 0 and 9999.

Note: If your access level is less than the minimum access level, you cannot start Tag Viewer at run time.

6. Select the **Window** tab.



7. In the **Menus** area, select the **Tag Viewer** check box to enable the **Tag Viewer** option in the **Special** menu of WindowViewer.
8. Select **OK**.
9. After you modify any of these parameters, you must restart WindowViewer to apply your changes.

Start Tag Viewer

You can start Tag Viewer only if WindowViewer is running, and Tag Viewer has been enabled at design time. You can launch Tag Viewer from the WindowViewer menu or from a script calling the LaunchTagViewer() function. This function can be executed from any script type, except application scripts: OnStartup and OnShutdown. If Tag Viewer is not enabled, calling the function will not start Tag Viewer and a warning message will be logged in the logger.

Security

You must have adequate security privileges to run Tag Viewer. The following tables explains the InTouch user security levels that are available.

Security Level	Description
ArchestrA	If you select this option, you must do the following: 1. In the **WindowViewer Properties** screen, set **Access Level >= Minimum Access Level**. 2. Log on to WindowViewer and start Tag Viewer. This is applicable if you are running a managed InTouch application.
InTouch	If you select this option, you must do the following: 1. In the **WindowViewer Properties** screen, set **Access Level >= Minimum Access Level**. 2. Log on to WindowViewer and start Tag Viewer. This is applicable if you are running a stand-alone InTouch application.
None	If you select this option, then any user can start Tag Viewer at run time without logging on to WindowViewer.
OS	If you select this option, you must do the following: 1. In the **WindowViewer Properties** screen, set **Access Level >= Minimum Access Leve**l. 2. Log on to WindowViewer and start Tag Viewer. This is applicable if you are running a stand-alone InTouch application.
CONNECT	This is applicable if you are running a stand-alone InTouch application and if you have selected the

Security Level	Description
	connected experience option under the **License mode** tab of the Configurator.

If you are opening an InTouch application, you must log on again to re-authenticate yourself to run Tag Viewer. For more details on configuring users, see Manage users and setting their authorization levels in AVEVA™ InTouch HMI Management Guide.

Note: You do not have to log on again if the security level of InTouch is None.

Log on

1. Open WindowViewer.
2. On the **Special** menu, select **Security**, and then select **Log On**.
3. Log on with a valid user ID, password and the required minimum access level.

Note: You can log on as administrator or create a user ID that has adequate privileges.

Start Tag Viewer

1. Open WindowViewer.
2. On the **Special** menu, select **Tag Viewer**. The Tag Viewer appears.

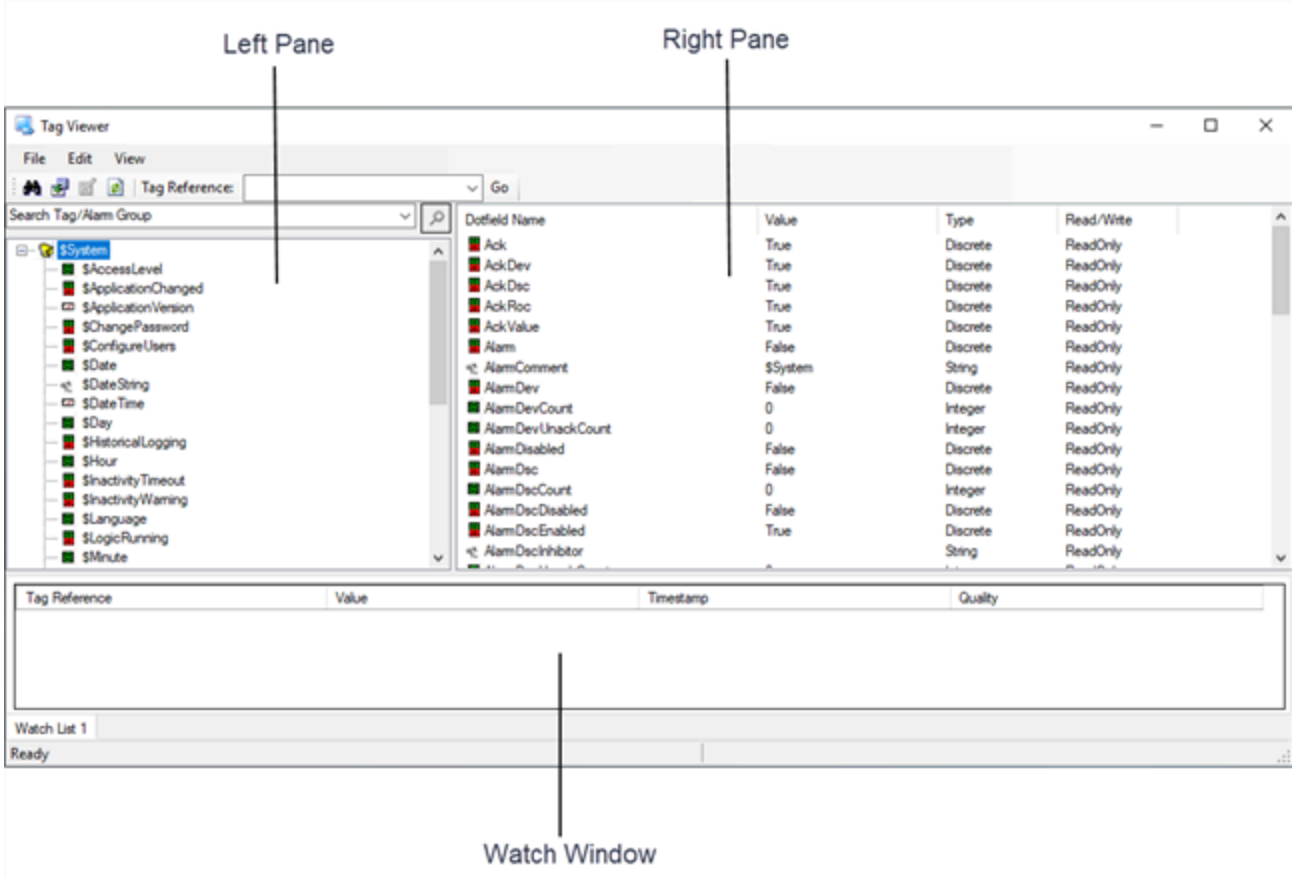
Start Tag Viewer from a script

1. Open WindowMaker.
2. Configure any script except application scripts "On Startup" and "On Shutdown" with the LaunchTagViewer() function.
3. Open WindowViewer.
4. Execute the script to start Tag Viewer.

Navigate in Tag Viewer

The Tag Viewer window contains three parts:

- The left pane, which displays the tags in a hierarchical structure, based on their alarm groups
- The right pane, which displays the list of all available dotfields for the tag or alarm group selected on the left pane
- The watch window at the bottom, which displays the run-time values of tags that you want to monitor.



Close Tag Viewer

Tag Viewer can be closed in the following ways:

- By closing Tag Viewer
- By closing WindowViewer
- By changing the logged-on user

Use Tag Viewer

You can use Tag Viewer at run time to:

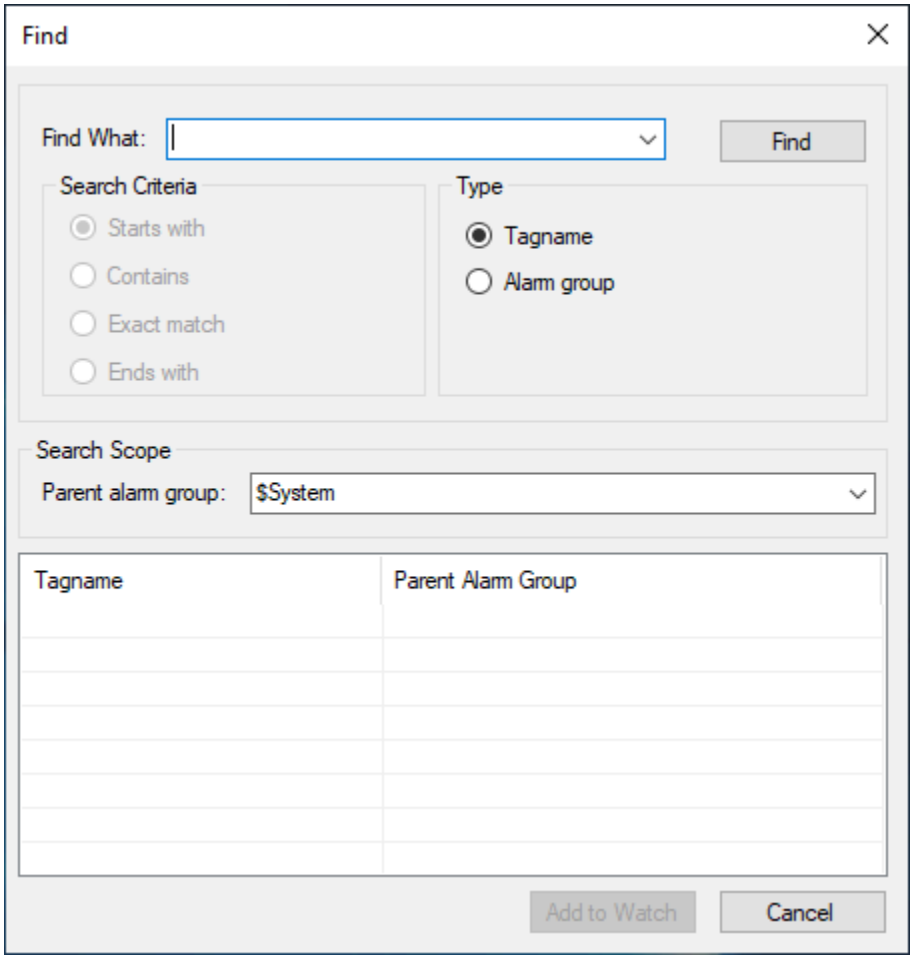
- Search for tags
- View tags
- Modify tag properties
- Add and populate watch windows
- Manage watch windows

Search for a tag

You can use the **Find** dialog box to search for specific tags in WindowViewer.

Search for a tag

1. On the **Edit** menu, select **Find Tag/Alarm** group. The **Find** dialog box appears.



2. In the **Find What** box, enter the name of the tag or alarm group that you want to find. By default, this displays the last tag or alarm group for which you have searched. The **Find What** box also maintains search history. If you are searching for a tag or an alarm group for the first time, the **Find What** box will be empty.

Note: The search history is removed when you close Tag Viewer.

3. In the **Search Criteria** area, select one of the following:
   - Starts with: Select this option to find a tag or alarm group that begins with the text you entered in the **Find What** box. By default, this option is selected.
   - Contains: Select this option to find a tag or alarm group that contains the text you entered in the **Find What** box.
   - Exact match: Select this option to find a tag or alarm group that matches the text you entered in the **Find What** box.
   - Ends with: Select this option to find a tag or alarm group that ends with the text you entered in the **Find What** box.

4. In the **Type** area, select one of the following:
   - Tagname: Select this option to find a tag. By default, this option is selected.
   - Alarm group: Select this option to find an alarm group.
5. In the **Parent Alarm Group** box, enter the name of the alarm group to which the tag belongs. If you retain **$System,** which is the default value, the system does not filter the search result by any alarm group.
6. Select **Find**. The list of tags or alarm groups appears.
7. In the search result, double-select the tag name to select it in the Tag Viewer.

Note: To add the tag to the watch window, select **Add to Watch**.

Perform a quick search

Tag Viewer allows you to search for a tag quickly, if required.

Perform a quick search

1. In the search box above the left pane, enter all or part of the tag name.

Note: The search box also maintains search history until you close Tag Viewer.

2. Press **Enter** or select the search icon. The system selects a tag that begins with the text you entered.

Note: If the text you entered begins with a wildcard (*), the system will find any tag name that contains the text.

3. Select the search icon again to resume your search.

Note: You can also enter the text and press **F3** to perform a quick search.

Manage tags

Tag Viewer allows you to watch tags and monitor their values at run time. You can view all tags available in the application. The tags are arranged in a tree structure, based on their alarm groups.

View tags

You can view the details of tags in Tag Viewer.

View tags

1. Open Tag Viewer.
2. On the left pane, view the list of tags available in the application.

Note: Tag Viewer displays the tags available in the InTouch application only and does not support remote references.

3. On the left pane, select a tag to view its details on the right pane. The following information appears:
   - Dotfield Name: Displays the dotfields of the selected tag or alarm group.
   - Value: Displays the value of the dotfield. If you select the dotfield, the value is updated.

- Type: Displays the dotfield type like integer, discrete.
- Read/Write: Indicates whether the tag is a read-only tag or can be modified.

4. Add a tag to the **Watch** window to view its dotfield values at run time. For more information on adding tags to the **Watch** window, see Adding a Tag or a Dotfield to a Watch Window in this addendum.

Modify tag properties

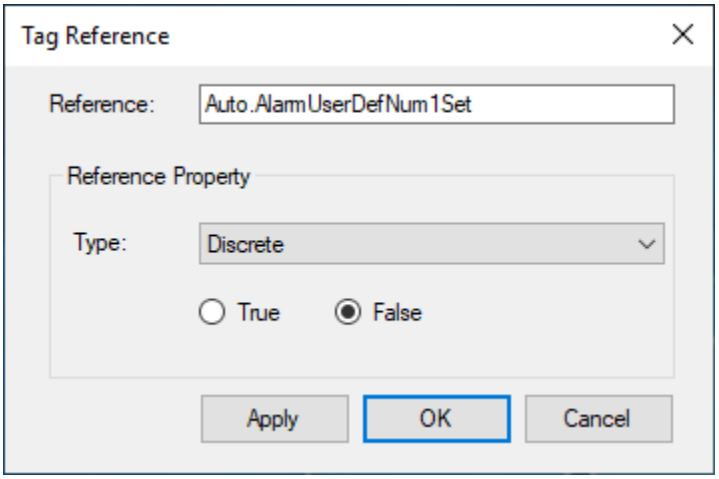
You can modify the properties of a tag in Tag Viewer. You can modify the properties of a tag only if the dotfield is marked as Read/Write or Write Only. You cannot modify any tag if you are running WindowViewer on a read-only license.

You will be asked to re-authenticate yourself if you try to modify the following:

- An indirect tag that is pointing to a Galaxy attribute, which is configured with Secured Write.
- An indirect tag that is pointing to a Galaxy attribute, which is configured with Verified Write. In this case, you must provide the log on details of the additional user too.

Modify tag properties

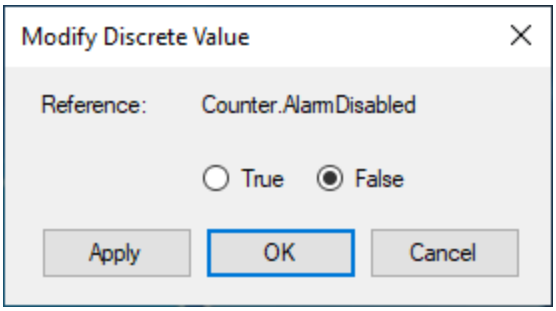
1. Open Tag Viewer.
2. On the left pane, select the tag to modify its properties. The tag details are displayed on the right pane.
3. Do any of the following:
   - On the right pane, select the property, and then select **Go** on the toolbar. The **Tag Reference** dialog box appears.



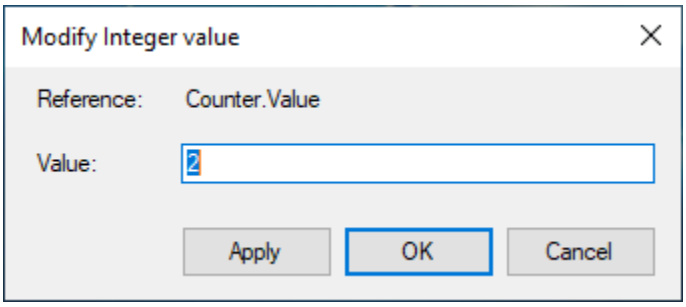
   - In the **Type** list, select the tag type, and change the value.

     Double-click any value on the right pane if you want to change the data type of the dotfield. The **Modify** dialog box appears. The dialog box varies depending on the data type of the dotfield. You can modify the value of the following data types:

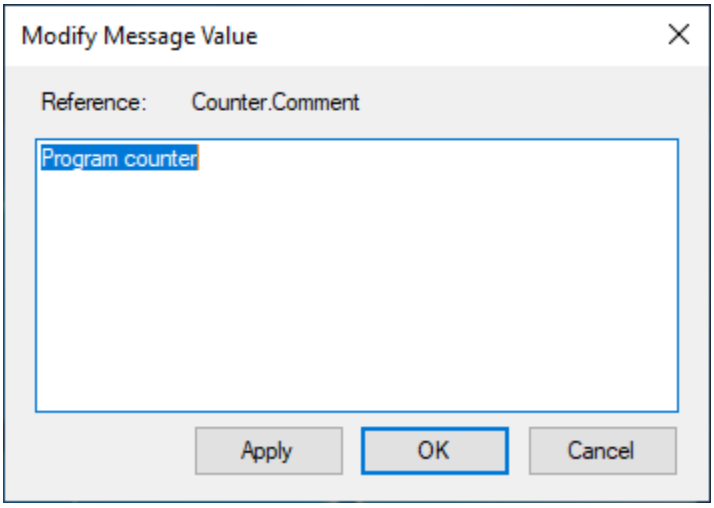
     Discrete Value: You can select **True** or **False**.



Number: You can enter any number between -2147483648 and 2147483647.



Message: You can enter any text message upto 131 characters.



- Right-click the item, and then select **Modify**. The **Modify** dialog box appears. Modify the value.

4. Select **Apply**. The value is changed.

Manage watch windows

The watch window is displayed at the bottom of the Tag Viewer by default. You can use this window to view the value of tags at run time. You can create additional watch windows to group relevant tags together. You can also remove a watch window, if required.

Add a watch window

You can create additional watch windows if you want to group tags and view them. The new watch windows are

added as tabs next to the default watch window. You can create up to 50 watch windows.

Add a watch window

1. Right-click the watch window, and then select **Add Watch Window**. The new watch window is added. By default, it is called Watch List <n>.

   Note: To rename the watch window, right-click the watch window tab and select **Rename Tab**.
2. Add tags to the watch window, as required.
3. Move from one watch window to another by selecting the tabs.

Add a tag or a dotfield to a watch window

You can add tags or dotfields to a watch window to monitor their values at run time. You can add up to 2000 tags or dotfields to a watch window.

Note: If you add an alarm group to a watch window, all tags under this alarm group are added to the watch list.

Add a tag to a watch window

1. Do any of the following:
   - On the left pane, right-click the tag and select **Add to Watch**. The tag is added to the watch window.
   - On the left pane, select the tag and drag and drop it to the watch window.
   - Right-click the watch window and select **Add Tag Reference**.
   - On the right pane, right-click any dotfield and select **Add to Watch**. The dotfield is added to the watch window.

     Note: You can select multiple tags or dotfields and add them to the watch window.
2. Select any column header to sort the table, based on the column in ascending or descending order.

Add a separator to group tags

You can group similar tags by adding a separator after a group of tags. If you save a watch window, the separators in the watch window are automatically saved.

Add a separator to group tags

1. On the watch window, select the row after which you want to add the separator.
2. Right-click the watch window, and select **Add Separator**. The separator is added after the selected row.
3. Select the separator along with the tag references and drag and drop them to another location to change the order of tags in the watch window.

   Note: If you sort the information on the watch window the separator is removed.

Back up and restore watch windows

You can save a set of watch windows as a back up and load them again when required.

Save a set of watch windows

You can save all the watch windows that you have opened, if required. You can later load this set of watch windows and monitor the tags.

Save a set of watch windows

- On the **File** menu, select **Save Watch List** and save the watch windows. By default, InTouch saves the watch windows in the **Watch Lists** folder in the current application directory.

Note: To save the watch windows in a different location, select **Save Watch List As** on the **File** menu.

Load a set of watch windows

You can open any watch window that you have saved earlier.

Load a watch window

1. On the **File** menu, select **Load Watch List**. The **Select a File** dialog box appears.

   Note: You can also right-click the watch window and select **Load Watch List**.
2. Browse to the location where the file is saved and open the file.

   Note: On opening a watch window, the existing windows are replaced with the new one. Only the references that are valid in the current application are added to the watch window.

I/O references

I/O references or indirect tags that are displayed in Tag Viewer behave like the windows in WindowViewer.

- If you remove an I/O or indirect tag from the watch window, the I/O reference is unsubscribed.
- If you have not added an I/O or indirect tag to the watch window, but its dotfield has been refreshed or you have selected the tag on the left pane, Tag Viewer subscribes to the I/O reference. The subscription remains as long as the tag remains selected.
- If you close Tag Viewer, all I/O or indirect tags subscribed by Tag Viewer are unsubscribed.
- Closing Tag Viewer or removing tags from the watch window has no impact on tags in WindowViewer.

Persistence of Tag Viewer

Tag Viewer should be persistent with respect to its InTouch application in terms of the following:

- Location and size of the main window.
- Size of the tag tree structure on the left pane, the dotfield list on the left pane, and the watch window.

Tag Viewer should be persistent with respect to its WindowViewer session in terms of the following:

- Selected tag on the left pane
- Watch window
- Status bar and tool bar

Note: The persistence with its WindowViewer session is removed when you close WindowViewer.

Watch window file format

The watch window file that you have saved uses XML file format. The XML schema is:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="InTouchTagViewer">
<xs:complexType>
<xs:sequence>
<xs:element name="Watch" minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="ReferenceString" type="xs:string" minOccurs="0"
maxOccurs="unbounded"></xs:element>
<xs:element name="Separator" minOccurs="0" maxOccurs="unbounded"></xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Reduce tag usage

The maximum number of tags you can use in an InTouch application is based upon your license. Your license prevents a running application's concurrent tag count from exceeding the maximum limit.

You can track tag usage in an InTouch application. The figure below shows how the Cross Reference Utility produces a tag usage report by analyzing the contents of the Tagname Dictionary.

You should maintain the minimum number of tags required to run your InTouch application. One way to minimize your tag count is to delete unused tags. You must update the tag count before deleting unused tags.

InTouch and the Historian

The Historian is a real-time and historical database that stores plant process data. It combines the storage capacity of a relational database with the speed of a real-time system. As an enhancement to Microsoft SQL Server, the Historian acquires plant data at dramatically increased speeds. It also integrates plant data with event, summary, production, and historical data from the InTouch HMI and other related products.

The InTouch HMI can store historical tag data to remote historical repositories. In this case, you must share the InTouch HMI application directory and configure Historian to access the InTouch application. If you use the Historian to store InTouch HMI historical data, you must use the Distributed Name Manager from WindowMaker to specify a connection to the database.The InTouch HMI can show historical data stored in a Historian database in a Trend Wizard. You need to install SQL Server client components and browse the Historian computer to access historical tag data.

Determine tag usage

The InTouch HMI maintains a use count for all tags defined in the local Tagname Dictionary. The tag count does not include internal system tags.

Remote tags are not defined in the Tagname Dictionary. InTouch increments the tag count for each reference within an application to a remote tag. For more information about how InTouch counts a remote tag reference, see Determine maximum number of remote tags based on licensing.

Before you delete unused tags, make sure that you complete the following tasks:

- Close WindowViewer.
- Update local tag counts and references to remote tags.
- Produce a Cross Reference tag report.
- Locate tag usage within an application with the Cross Reference Utility.
- Save your application in WindowMaker.

Determine tag counts

You can update the count of local tags that are currently defined in an application's Tagname Dictionary. You can also update the count of an application's references to remote tags located on another node.

Show the local tag count

1. Open an InTouch application in WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowMaker**.

   The WindowMaker configuration screen appears.
3. Select **Show Tag Count**.

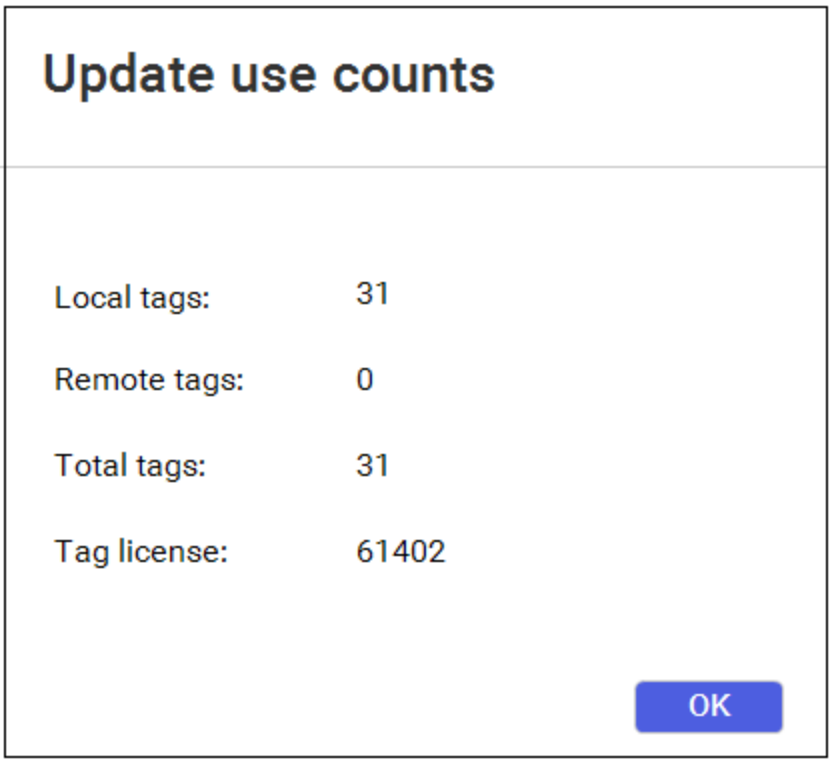
   When you select this option, the entire Tagname Dictionary must be read to update the displayed tag count. Updates to the Tagname Dictionary may take longer.
4. Select **Save**. A message appears that WindowMaker must be restarted before the configuration changes become effective.
5. Close WindowMaker.
6. Restart the application in WindowMaker.

   The number of local tags defined in the application's Tagname Dictionary are shown at the right of the WindowMaker menu bar.

Update the total tag count including remote references in an application

1. Open an InTouch application in WindowMaker.
2. Close all open windows.
3. On the **Home** menu, in the **Tags** group, select **Update use counts.**

   After the calculations are complete, a dialog box shows the counts for the application's local tags and references to remote tags.



The dialog box also includes a total tag count and maximum tag usage based upon the InTouch license.

Determine maximum number of remote tags based on licensing

Your product license enforces the number of tags you can use with your InTouch applications. A license enforces not only the local tag count, but also the number of tags that reference tag sources located on remote nodes.

For more information about how an InTouch license enforces the number of remote reference tags that can be used in your applications, see InTouch Licensing.

Locate where tags are used

Using the Cross Reference Utility, you can produce an online report that shows tag usage within an InTouch application.

The Cross Reference Utility report shows local tags, remote tags, and SuperTags in:

- Animation links
- Wizards
- All types of InTouch scripts
- QuickFunctions
- Active controls
- Optional InTouch components such as SQL Access Manager and Recipe Manager.
- Industrial Graphics references

Create a report with the Cross Reference Utility

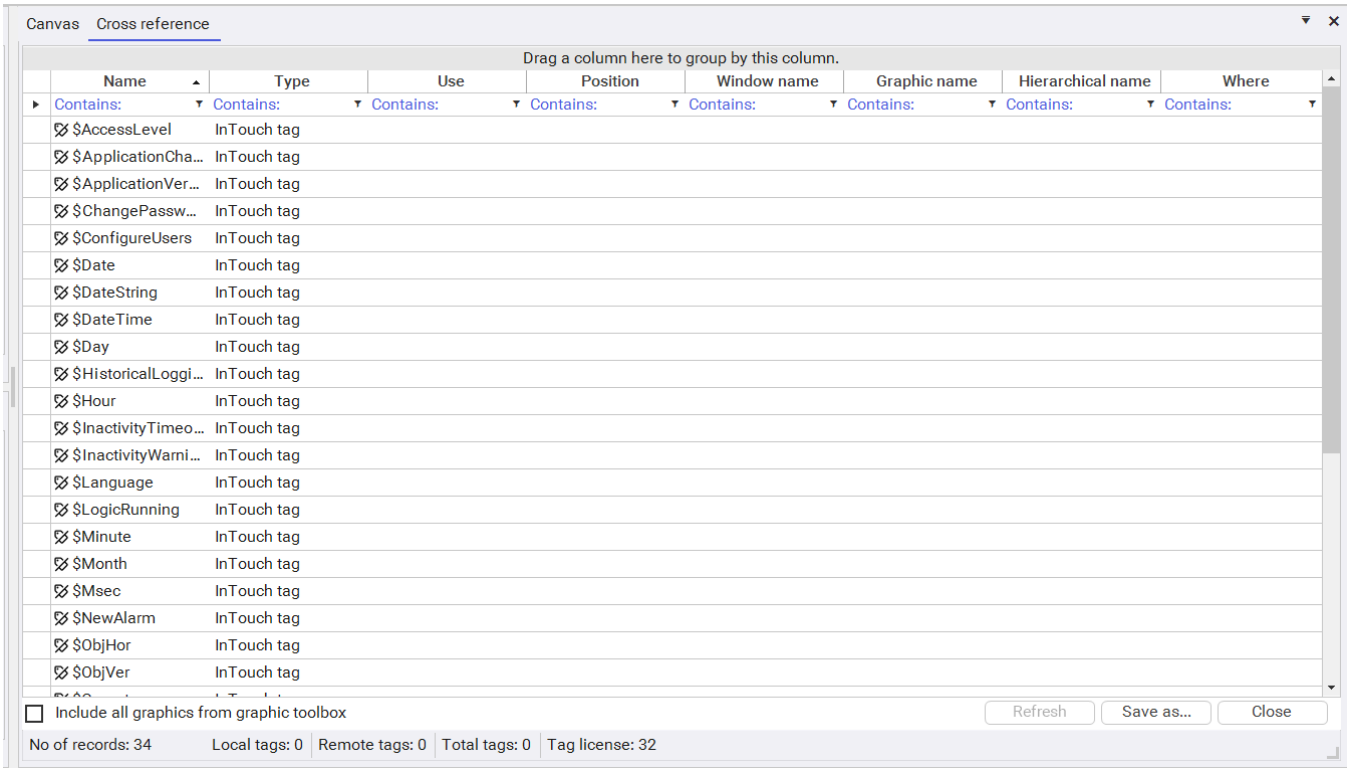
1. Open an InTouch application in WindowMaker.
2. On the **Home** menu, in the **Tags** group, select **Cross Reference**.

   The **Cross Reference** grid appears.

   The Cross Reference grid lists all records related to local and remote tags defined in the current application's Tagname Dictionary.

Understand the Cross Reference utility report

In a grid format, the report lists the name of the reference type, type of tag reference, use, position on the window, window name, graphic name, full hierarchical path of the Industrial graphic having the tag reference, and the condition type if the tag is used in a script.



For example:

Name	Type	Use	Position	Window Name	Graphic Name	Hierarchical Name	Where
NSelect	InTouch Tag	Key Script					F6 On Key Down
PLCSim.SP3	Object Attribute	Industrial Graphic	At (0,0) by (826,455)	Section5	SA_Meters2	SA_Meters.SA_Tank_Vessel.SA_OperatingRange	

The status bar at the bottom of the report displays the:

- **No. of records**: Total number of records related to tags in the application. This value will change dynamically based on the applied filter.
- **Local Tags**: Total number of tags created on this node.
- **Remote Tags**: Total number of tags that reference remote tags.
- **Total Tags**: Total number of tags in the InTouch application.
- **Tag License**: Total number of tags allowed as per the license.

Icons appear in the InTouch Cross Reference utility report to show status and usage.

Icon	Description
	The tag or SuperTag is defined in the application's Tagname Dictionary, but is not assigned to an object.
	The tag or SuperTag is used in an animation link, InTouch QuickScript or Industrial graphic.
	The Industrial graphic is referencing a tag or SuperTag.
	The tag or SuperTag is assigned to an animation link.
	The tag or SuperTag is used in an Application script.
	Shown for all selected scripts. Double-click on the script name to view the script in read only mode.
	The tag or SuperTag is used in a Window script.
	The tag or SuperTag is used in a Data Change script.
	The tag or SuperTag is used in a Condition script.
	The tag or SuperTag is used in a Key script.
	The tag or SuperTag is used in a QuickFunction.
	The tag or SuperTag is used in an ActiveX Event script.
	When cross-referencing by **Window**, this icon precedes the window name in which the displayed tag or SuperTag is used. Double-click on the icon to view all tags used in the window.
	Displayed tag or SuperTag is used in a SQL application.
	Displayed tag or SuperTag is used in a Recipe Manager application.
	The tag is used as an alarm inhibitor.

Include graphics from the Graphic Toolbox

- Select **Refresh** to view the latest tag information. If you edit the tag information or edit a symbol embedded on a window, while the Cross Reference window is open, then the Refresh button is enabled.
- Select the **Include all graphics from Graphic Toolbox** checkbox to view the tags that are not used in any Windows but present in the Graphic toolbox. By default, the checkbox is not selected.

  For example: 10 tags are configured in the tag dictionary, 4 are referenced in graphics in the graphic toolbox (but not embedded on any windows), 4 tags are referenced on graphics placed on a window. By default, 4 tags (tags placed on a window) will be displayed. If the checkbox is selected, then 8 tags (4 from the window + 4 from the graphics) are displayed.

The graphics displayed are subject to the following conditions:

- Remote tag references within symbols, which are not used in InTouch windows will not be displayed.
- Object symbols, which are not used in InTouch windows, will not be displayed.
- If the checkbox is selected in the Cross Reference Utility, the condition will also be set for the other operations like Update Use Count and Delete Unused Tags.

Cross reference tag usage for InTouch tags in ArchestrA or situational awareness library symbols

The InTouch Cross Reference utility has been enhanced to include Industrial Graphics in an application's standard InTouch tag cross reference list. A cross reference search for Industrial Graphics can be conducted by the criteria shown below, which will be included as part of the Cross Reference utility's InTouch Tag Use.

- Custom property with reference to an InTouch tag
- Animation with direct reference to an InTouch tag
- Client script with direct reference to an InTouch tag
- Embedded symbols' reference to an InTouch tag

The Cross Reference Utility will also search Industrial graphics for attribute references, which are part of the Object Attribute. The exceptions to the tag usage report are:

1. Object Attribute references:
   - Object Attribute using relative reference (such as me.I1) will not be supported and will not be displayed in the cross reference utility tag usage report.
   - Reference using <InTouchViewApp instance name>.<Tagname> syntax will not be considered as InTouch Tag reference. This reference will be considered as Object Attribute.
     Example: InTouchViewApp1.Tag1
   - Any reference that is not in "InTouch:<Tag>" syntax will be considered as Object Attribute reference.
2. Client script references that are used as the script function parameters will not be considered as InTouch tag or object attribute reference.

   Example: reference as string parameter
   SetCustomPropertyValue("CP1", "InTouch:Tag1", false);

   Example: reference as Attribute parameter

SetBad(InTouch:Tag1);

3. Industrial Graphics referencing to an InTouch tag that is not defined in the current InTouch application will be displayed with Type as "Undefined".
   Example: Object symbol
   ArchestrA object "UD_001" has instance symbol "S1", "S2", and "S3".
   The instance symbol "UD_001.S1" has reference to InTouch tag "Tag51".
   Embed UD_001.S1 into UD_001.S2;
   Embed UD_001.S2 into UD_001.S3;
   Embed UD_001.S3 into InTouch window "Win1".
   Also embed UD_001.S2 into window "Win1".

The instance symbol can be defined within the instance or in its derived template. The instance can be embedded using absolute or relative reference.

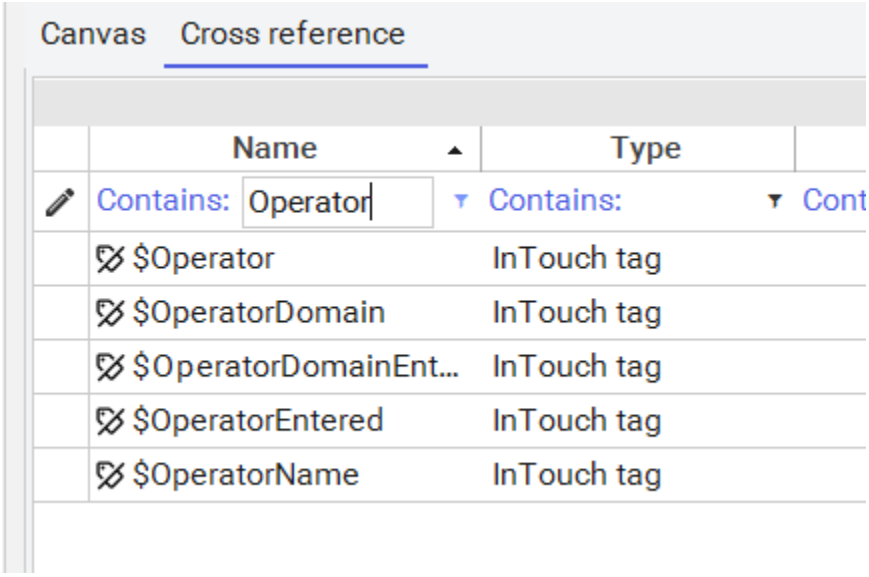
Display tag usage in the Cross Reference utility

You can modify the display of the cross reference tag usage report using the filtering, sorting and grouping options. Applying filters will update the number of records displayed. Sorting and grouping will only modifying the way the records are displayed on the grid.

Refine the tag usage report using the filter option

1. Use the Filter icon ( ) to select from a list of options like Contains, Does not contain, Starts with, Ends with, Equals, Not equal to, Is null, Is not null and Custom.
2. Enter the search term against the selected filter option.

   The grid will display all records that match the filter option and the search term.



Use the custom filter option

The custom filter option allows you to add multiple tags and create a complex search expression.

Sort the tag usage report

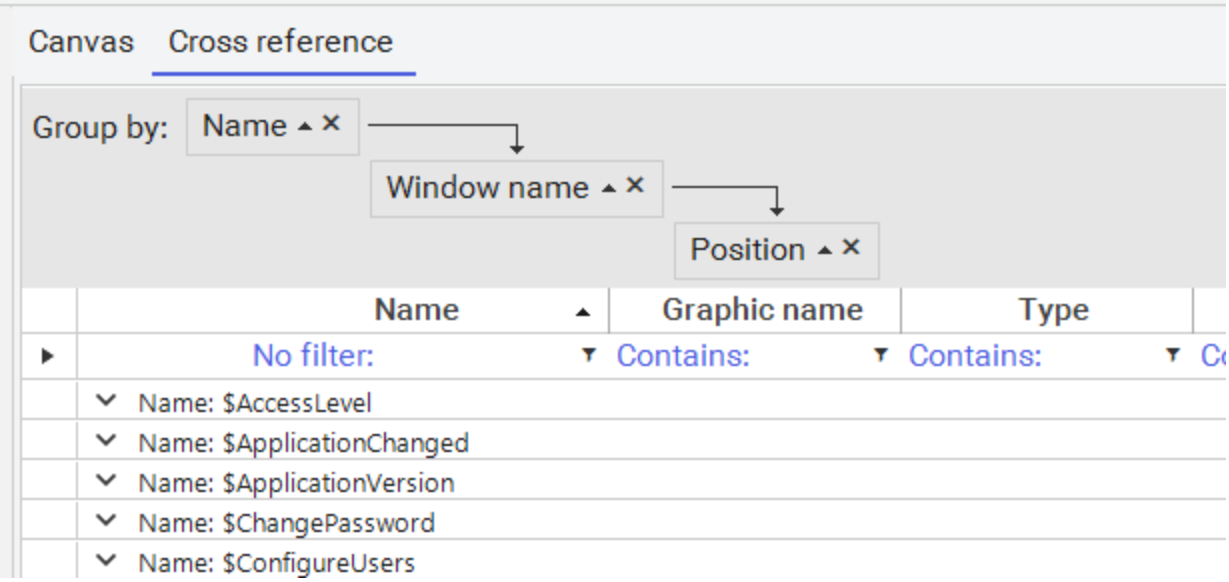
Each column in the tag usage report can be sorted in ascending or descending order.

1. Select the column heading.
   All records in the grid will be sorted based on the column. The direction of the arrow determines the order.
2. Select the column heading again to change the order.

Group the tag usage report

Organize the tag usage report in a pattern using a combination of columns.

1. Select a column name and drop it to the empty space above the column names.
   A box with name of the column appears.
2. Select another column name and drop it above or below the first box.
   The utility will automatically create a hierarchical relationship between the two columns.
3. To create two columns at the same level drop the second column name on top of the first box.
4. Sort individual columns in ascending or descending by selecting the column name box within the pattern.



Saving and Printing a Tag Cross-Reference List

You can save your cross reference tag listing to a file and view it with any application that supports the .csv file format.

The cross reference tag listing file lists all tags by name, how they are used in the application, and the name of the window where they are located. You can open the cross reference tag listing file with Excel or any other program that supports .csv files.

You can also print the contents of the Tagname Dictionary. Printing the contents of the Tagname Dictionary shows you database entries used in the application. You can specify the level of detail you want to see.

You can send this report to the printer or save it to a file.

To save a cross-reference file

1. In the **Cross Reference Utility** dialog box, select **Save As**. The **Save As** dialog box appears. Specify a name and location for the file.
2. Select **Save**. The tag usage file is saved in the specified folder.

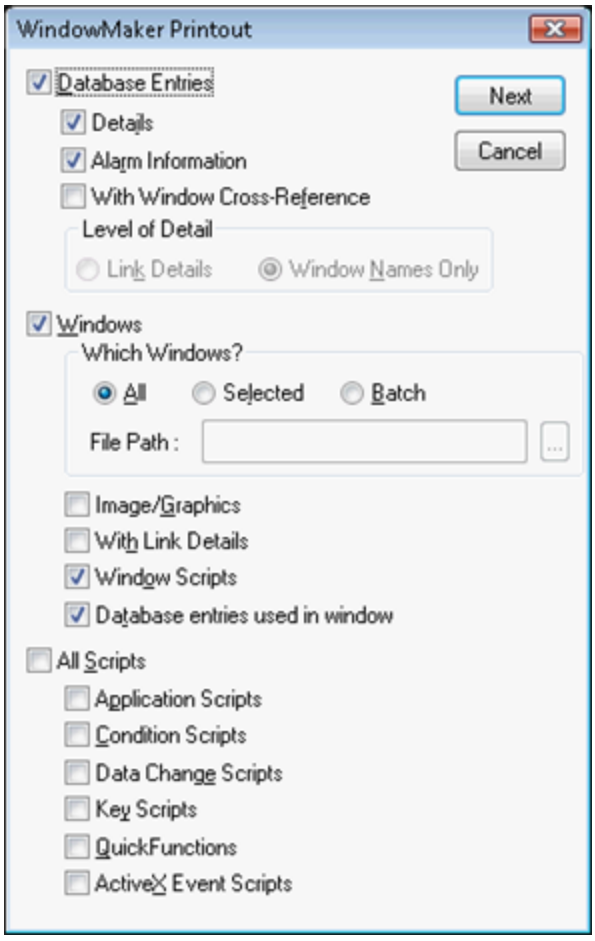
The tag usage report saved to a .csv file will depend on the following conditions:

- If no search conditions are selected then all records are saved to the .csv files. For example: If the total number of records in the cross reference grid is 1000 and on refining the search, the No. of records reduces to 320. If you use Save As... only the selected 320 records are saved to the .csv file.
- If you have applied a grouping pattern to your records, the .csv will be saved with that grouping pattern.

To print the contents of the Tagname Dictionary

1. Open an InTouch application in WindowMaker.
2. On the quick access toolbar, select **Print**.

   The **WindowMaker Printout** dialog box appears.



3. Select **Database Entries** if you want to print tag information from the Tagname Dictionary.

   If you select **Database Entries**, the following options become active:

   - Select **Details** to include the details of the tags in your report.
   - Select **Alarm Information** to include the tag alarm information in your report.
   - Select **With Window Cross-Reference** to print all Tagname Dictionary entries with window cross-references. If you select this option, specify the level of detail to print.

     Link Details prints the location and animation link details where the tag is used.

     Window Names Only prints only the name of the cross-referenced windows.
4. Select **Next**. The **Select Output Destination** dialog box appears.
5. Select the option to print the contents of the Tagname Dictionary or send the output to a text or .html file.
6. Select **Print**.

You can save your cross reference tag listing to a file and view it with any application that supports the .csv file format.

Delete unused tags

You can delete unused tags from an InTouch application after updating the use count. You can delete tags one at a time from the Tagname Dictionary, or you can delete several tags at one time.

The tag count is not automatically updated when a window containing tags is deleted from an application or tags are changed in link scripts. InTouch regards the tags as being in use and prevents them from being deleted.

Note: Deleting a single tag from the Tagname Dictionary: If the **Delete** button is disabled for a unused tag in an Industrial Graphic reference, run **Update Use Counts** to enable the Delete button.

The tag count must be updated to remove unused tags from the total before you can delete unused tags. For more information about updating tag counts, see Determine tag counts.

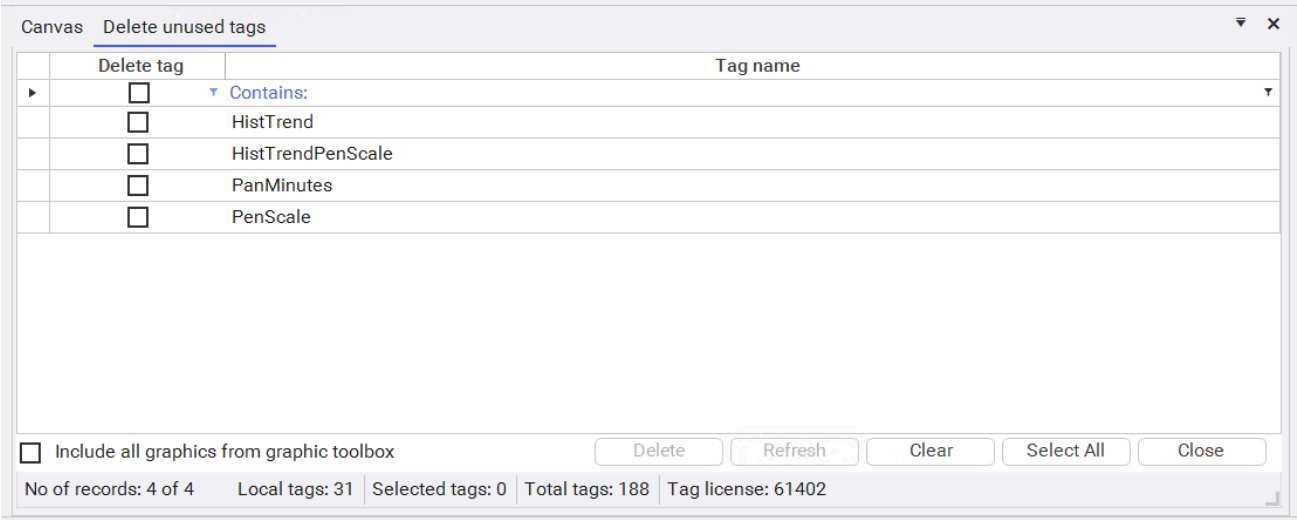
Caution: Tags that are only alarmed have no use count and can be accidentally deleted. To ensure that alarmed only tags are included in the use count, you need to use them in a window or QuickScript.

Delete multiple unused tags

1. Close WindowViewer if it is running.
2. Open an InTouch application in WindowMaker.
3. On the **Home** menu, in the **Tags** group, select **Delete Unused Tags.**

   The **Delete Unused Tags** dialog box appears with a list of unused tags.

   The status bar at the bottom of the dialog box lists the Number of records, Number of Local tags, Number of selected tags, Total number of tags and the Tag license count.



4. Use the filter ( ) option to refine the list of unused tags. Select the filter option and enter a search term.

   The tags matching the criteria are displayed.
5. Select the column headings to sort the columns.
6. Select the tags you want to delete. To select all tags for the current filter criteria, select **Select All**.

   Note: The status bar displays the number of tags that have been selected (for example: No. of records: 1 of 14). The tag selection is maintained even if the column criteria is changed. The “Selected Tags” value will reflect the total number of tags that user has selected, independent of filter criteria. To view the entire list of selected tags, clear all the filter criteria.
7. Select **Clear**, to clear the tag selection for the current filter.
8. Select the **Include all graphics from Graphic Toolbox** checkbox to only see tags that are not used in any graphics or windows. These tags are safe to delete as they are not referenced in graphics or windows that contains graphics.

   For example: 10 tags are configured in the tag dictionary, 4 are referenced in graphics in the graphic toolbox

(but not embedded on any windows), 4 tags are referenced on graphics placed on windows. By default, 6 tags (2 unused tags + 4 tags from the graphics in the graphic toolbox not embedded on any window) will be displayed. If the checkbox is selected, then only the 2 unused tags are displayed.

9. Select **Delete**, to delete all tag selections independent of the filter criteria.

   A confirmation message appears. The message will specify the number of tags that will be deleted.

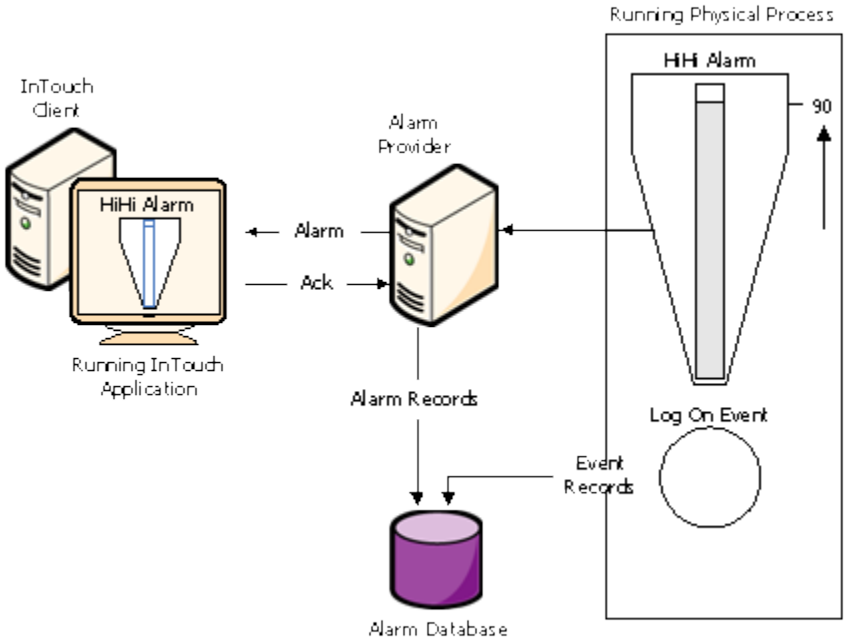
10. Select **OK** to confirm.

Alarms

You can create InTouch applications that generate alarms and events to notify operators about the status of process activity.

- Alarms warn run-time operators about process conditions that could potentially cause problems. Typically, you set up an alarm to trigger when a process value exceeds a defined limit. An operator must usually acknowledge the alarm.
- Events represent normal system status messages. A typical event is when a system condition occurs, such as an operator logging on to an InTouch application. Operators do not have to acknowledge events.

The following figure shows how the InTouch HMI handles alarms and events while an application is running. Alarm and event data is saved to the alarm database.

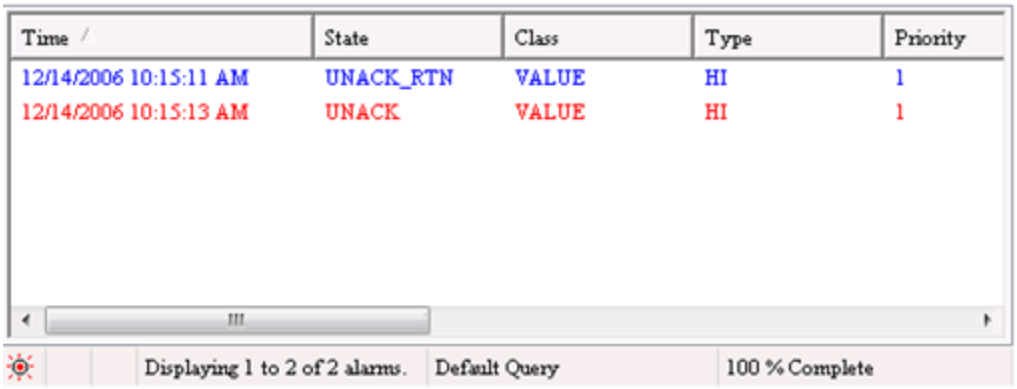


You can configure any tag for event monitoring. An event message is logged to the alarm system each time the tag value changes. The event message includes how the value changed and whether the operator, I/O, scripts, or the system initiated the change.

View current alarms

Use the InTouch Alarm Viewer ActiveX control to view alarms. The Alarm Viewer control has scroll bars, sizable

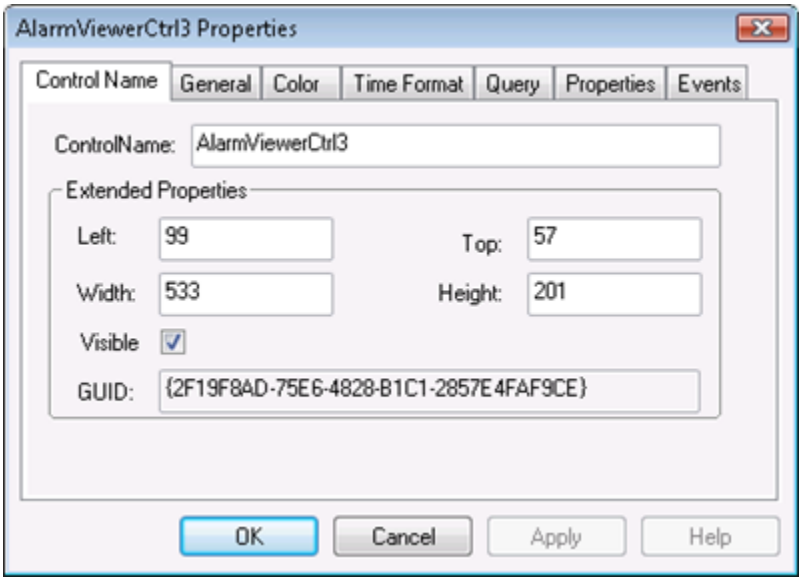
columns, multiple alarm selections, an update status bar, dynamic display types, and show colors based on the type of alarm.



We recommend that you use the Alarm Viewer control to view InTouch alarms. However, you can continue to use the Distributed Alarm object to view alarms from applications created with versions of InTouch earlier than 7.1.

Configure an alarm viewer control

You can set Alarm Viewer control options from WindowMaker and options that users can modify while the Alarm Viewer control is running. You set these option in the **AlarmViewerCtrl Properties** dialog box.



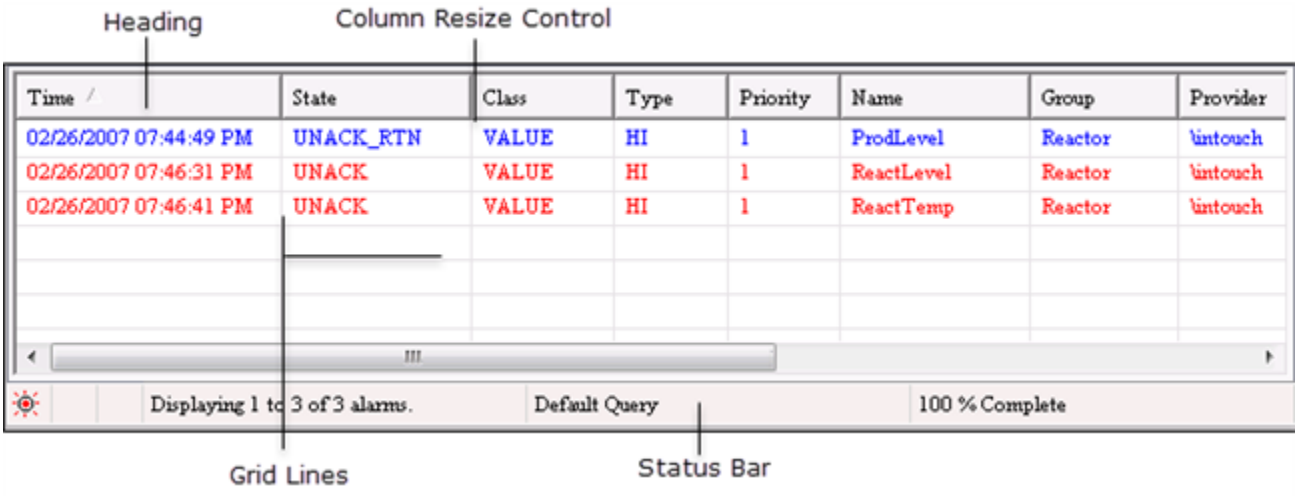
Configure the appearance of the grid

When you configure the visual appearance of the Alarm Viewer control, you can:

- Include a status bar.
- Include a column header.
- Include horizontal and vertical grid lines that show the rows and columns.
- Include a run-time option that enables the user to adjust the width of columns.

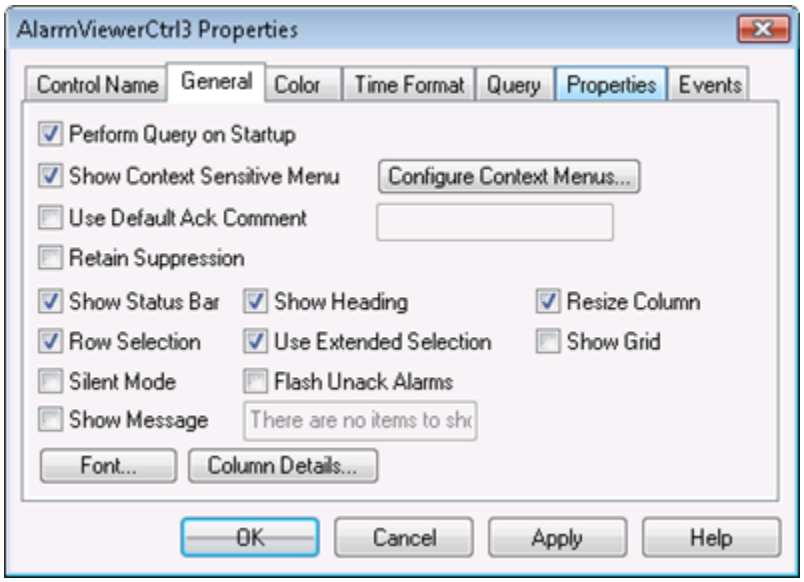
- Set the colors of visual elements.

The following figure shows how the Alarm Viewer control appears when all visual properties are active.



Configure the visual appearance

1. Right-click the Alarm Viewer control, and then select **Properties**. The **AlarmViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.



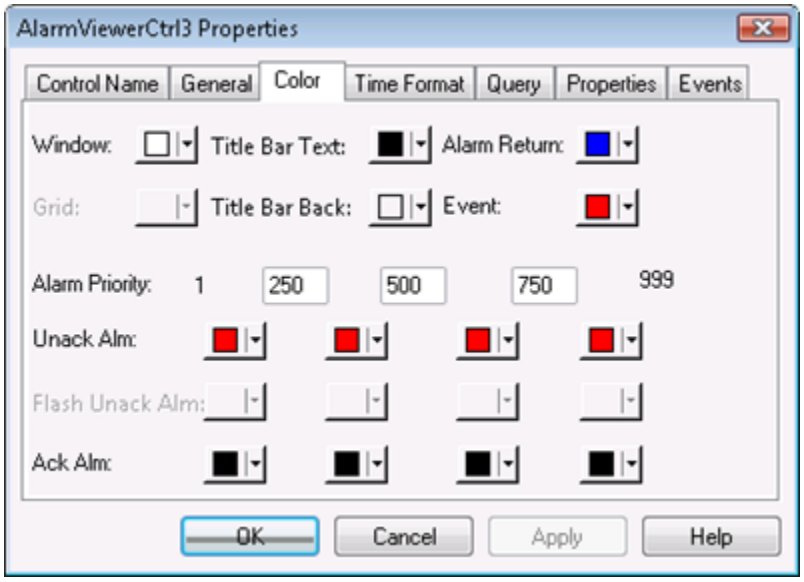
3. Set the visual appearance. Configure any of the following.

Option	Description
Perform Query on Startup	Automatically begins updating the control using default query properties. If a query is not run when the application starts, you need to run a script using the Requery() function to update the grid. The Requery option is

	also available on the grid's shortcut menu during run time.
Show Context Sensitive Menu	Enables a right-click shortcut menu during run time.
Use Default Ack Comment	Controls whether a default comment appears when an operator acknowledges an alarm. If this box is checked and a string is entered, the string is used during run time as the default comment. If this box is not selected, when the operator acknowledges an alarm, a dialog box appears to enter an optional comment. The dialog box can be filled in or left blank.
Retain Suppression	Retain alarm suppression between alarm queries when the alarm query is changed.
Show Status Bar	Toggles whether the status bar appears at the bottom of the Alarm Viewer control.
Row Selection	Enables users to select individual rows during run time. Each row represents an alarm record. Users can select multiple alarms.
Silent Mode	If Silent Mode is selected, the Alarm Viewer control does not show error messages during run time. If it is not selected, the alarm display shows pop up error messages. In either case, error messages are always sent to the Log Viewer.
Show Message	Shows the message typed in the text box. This is the message shown when there are no alarms.
Show Heading	Toggles whether the heading bar appears at the top of the Alarm Viewer control.
Use Extended Selection	Enables the user to select multiple alarms simultaneously by holding down **CTRL** or **SHIFT** keys in conjunction with the mouse button. Available only if Row Selection check box is selected.
Flash Unack Alarms	Enables unacknowledged alarms to flash once per second until they are acknowledged. Freezing the alarm display in WindowViewer does not stop unacknowledged alarms from flashing.

Resize Columns	If Resize Columns is selected, the user can resize the width of the columns during run time. Otherwise, column width is static and can be set only from WindowMaker.
Show Grid	If Show Grid is selected, the Alarm Viewer control shows horizontal and vertical lines that separate the rows and columns of the alarm display. If it is not selected, no grid is visible.

4. Select **Apply**.
5. Select the **Color** tab.



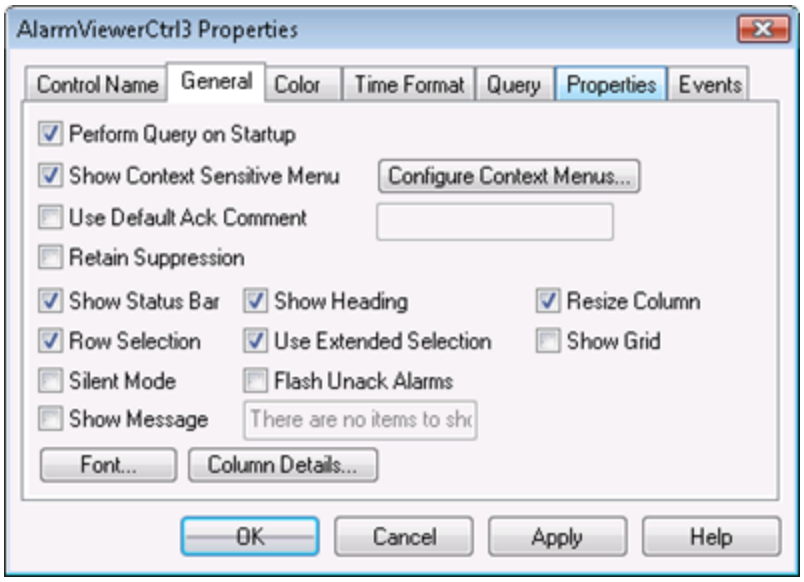
6. Select the palette button to assign colors to the visual elements of the Alarm Viewer control.
7. Select **Apply** to save your color selections.
8. Select **OK**.

Configure the font display

You can configure how the text appears for the Alarm Viewer control.

Configure the font properties

1. Right-click the Alarm Viewer control, and then select **Properties**. The **AlarmViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.



3. Select **Font**. The standard Windows **Font** dialog box appears. Configure the font and then select **OK**.
4. Select **OK**.

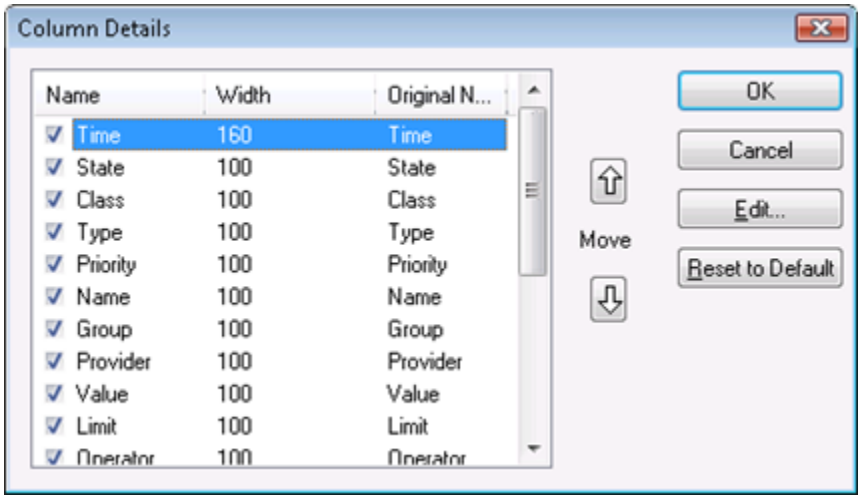
Configuring Display Column Details

For the Alarm Viewer control, you can:

- Select and order the columns.
- Set the width of a column in pixels.
- Rename a column.

To configure the display column details

1. Right-click the Alarm Viewer control and then select Properties. The **AlarmViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.
3. Select **Column Details**. The Column Details dialog box appears.

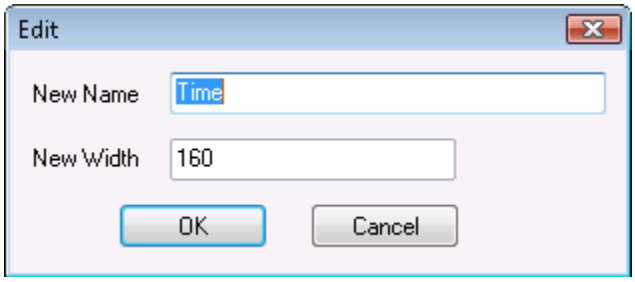


4. In the **Name** column, select the check boxes next to the names of the columns that you want to appear. You must select at least one column from the list.

Column	Shows
Time	The time as specified in the Time Format tab.
State	The state of the alarm.
Class	The alarm category.
Type	The alarm type.
Priority	The alarm priority.
Name	The tagname.
Group	The alarm group name.
Provider	The name of the alarm provider.
Value	The value of the tag when the alarm occurred. The width of the column should be large enough to provide the desired level of precision.
Limit	The alarm limit value of the tag. The width of the column should be large enough to provide the desired level of precision.
Operator	The logged-on operator’s ID associated with the alarm condition.
Operator Full Name	The logged-on operator’s full name.
Operator Node	The logged on operator’s node associated with the alarm condition. In a Terminal Services environment, this is the name of the client computer that the operator established the Terminal Services session from. If the node name can't be retrieved, the node's IP address is used instead.
Operator Domain	The logged on operator’s domain associated with the alarm condition.
Tag Comment	The comment for the tag.
Alarm Comment	The comment associated with the tag's alarm. This comment was typed in the Alarm Comment box when the tag’s alarm was defined. When an acknowledgement comment is introduced for alarms, the new comment is updated in this

	comment column.
User 1	The numerical value of the AlarmUserDefNum1 property of the alarm.
User 2	The numerical value of the AlarmUserDefNum2 property of the alarm.
User 3	The string value of the AlarmUserDefStr property of the alarm.

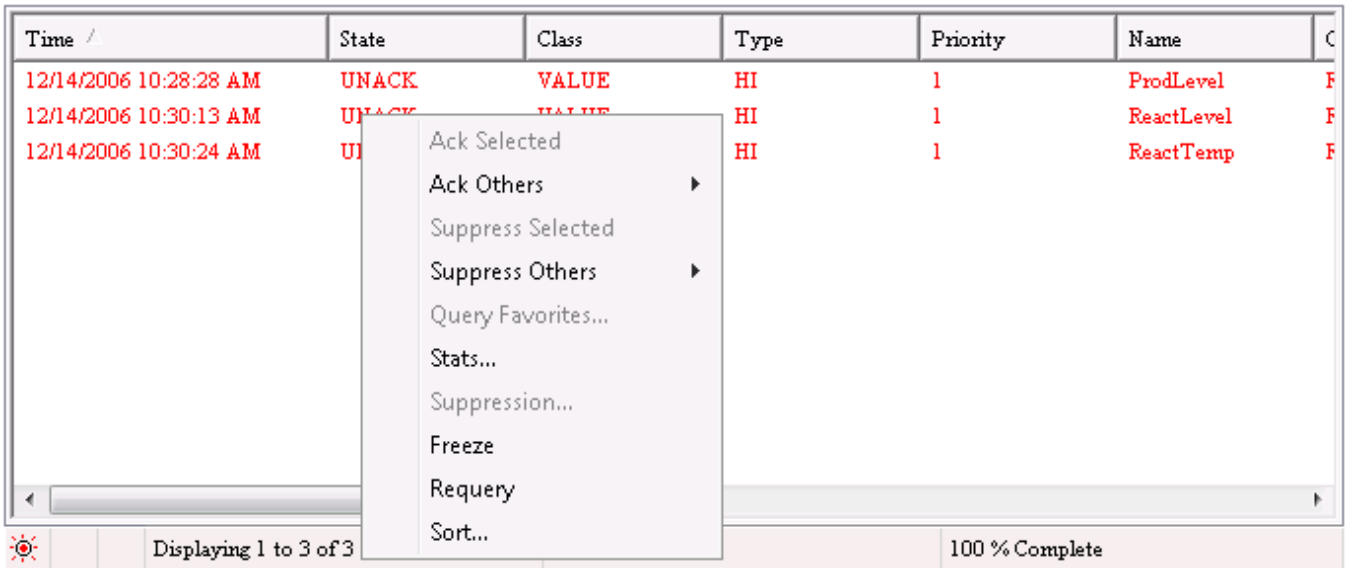
5. Rearrange columns by selecting the column name and using the up and down arrows. The column name appearing at the top of the **Column Details** dialog box is the left-most column of the alarm control.
6. To change the name of a column or its width, select the column and select **Edit**. The Edit dialog box appears.



   a. In the **New Name** box, type the new column name.
   b. In the **New Width** box, type the column width. The column width can range from 1 to 999 pixels.
   c. Select **OK**.
7. Select **OK** in the Column Details dialog box.
8. Select **Apply**.

Control access to features at runtime

If you right-click on the Alarm Viewer control during run time, a context-sensitive (shortcut) menu appears.



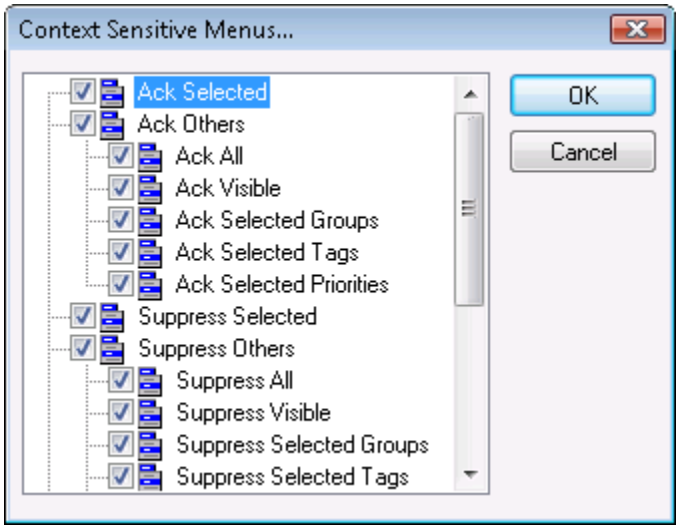
You can control which menu commands are shown in the shortcut menu.

Configure the shortcut menu

1. Right-click the **Alarm Viewer** control, and then select **Properties**.

   The **AlarmViewerCtrl Properties** dialog box appears.
2. In the **General** tab, select the **Show Context Sensitive Menu** check box.
3. Select **Configure Context Menus**.

   The **Context Sensitive Menus** dialog box appears.



   This dialog box shows a hierarchical list of commands that can appear in an Alarm Viewer control shortcut menu.
4. Configure the shortcut menu options. You must select at least one shortcut menu command.

This command	Allows the run-time user to
Ack Selected	Acknowledge selected alarms. If **Ack Selected** and **Ack Others** menu items are both unchecked, the **Use Default Ack Comment** check box and the text box are disabled.
Ack Others	Acknowledge alarms by other methods. The user can select which alarms to acknowledge. If **Ack Others** is selected, you must select at least one of the submenu items.
Ack All	Acknowledge all active alarms.
Ack Visible	Acknowledge visible alarms.
Ack Selected Groups	Acknowledge all alarms with the same group name as the selected group(s) and with the same provider name.
Ack Selected Tags	Acknowledge all alarms with the same group name

	as the selected tag(s) and with the same provider, group, and priority.
Ack Selected Priorities	Acknowledge all alarms with the same priority as the selected priority or priorities and with the same provider and group.
Suppress Selected	Suppress selected alarms.
Suppress Others	Suppress alarms by other methods shown in the shortcut menu.
Suppress All	Suppress all alarms.
Suppress Visible	Suppress all visible alarms.
Suppress Selected Groups	Suppress all alarms with the same group name as the selected group(s).
Suppress Selected Tags	Suppress all alarms with the same tagname as the selected tag(s).
Suppress Selected Priorities	Suppress all alarms with the same priority as the selected priority or priorities.
Unsuppress All	Unsuppress all suppressed alarms.
Query Favorites	Open the Alarm Query dialog box.
Stats	Open the Alarm Statistics dialog box.
Suppression	Open the Alarm Suppression dialog box.
Freeze	Toggle the freeze/unfreeze mode of the Alarm Viewer control.
Requery	Re-run the alarm query.
Sort	Open the Sort dialog box.

5. Select **OK**.
6. Select **Row Selection** to enable users to select a row from the Alarm Viewer control during run time.
7. Select **Use Extended Selection** to enable users to select multiple alarm records simultaneously from the Alarm Viewer control using the **SHIFT** or **CTRL** keys.
8. Select **Apply**.

Select the alarms to display

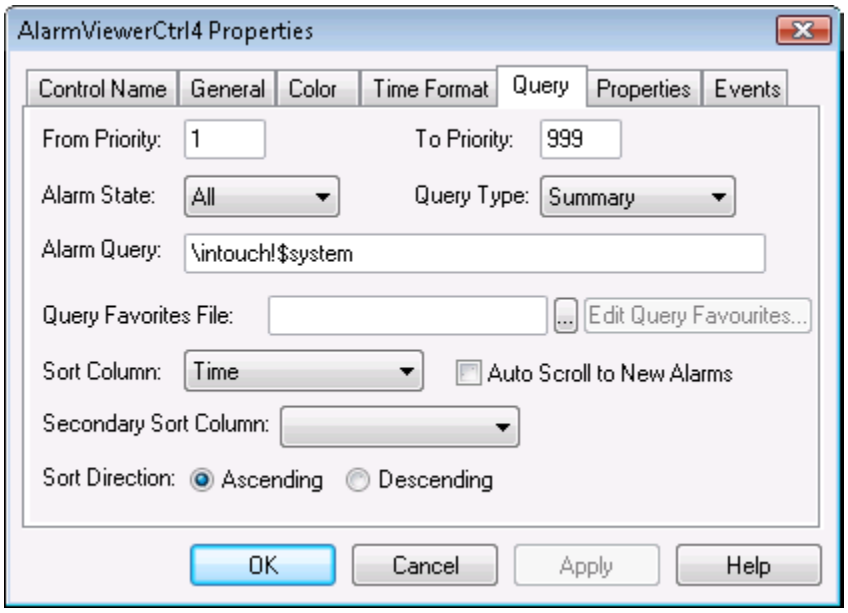
The Alarm Viewer control can show summaries of active alarms or listings of historical alarms.

Set general alarm query properties

1. Right-click the Alarm Viewer control and then select **Properties**. The **AlarmViewerCtrl Properties** dialog box appears.
2. Select the **General** tab.
3. Select the **Perform Query on Startup** check box to automatically update the Alarm Viewer control using default query properties when the application starts.
4. Select the **Show Message** check box to show a default message when there are no alarms. In the text box, type the message to show.
5. Select **Apply**.

Configure the query default

1. Right-click the Alarm Viewer control and then select **Properties**. The **AlarmViewerCtrl Properties** dialog box appears.
2. Select the **Query** tab.



3. In the **From Priority** box, type the minimum alarm priority value (1 to 999).
4. In the **To Priority** box, type the maximum alarm priority value (1 to 999).
5. Select the **Query Type** arrow and select either **Historical or Summary** as the default run-time alarm display.

   The default type of display can be changed during run time by running a QuickScript containing a query function. For example, if the script includes the ApplyQuery() method with its Type parameter set to “Summary,” then the grid shows a summary of current alarms. Conversely, if the same grid has an ApplyQuery() method run against it with the Type parameter set to "Historical", it shows historical alarms. The QueryType property reflects the current state of the alarm display.
6. In the **Alarm Query** box, type a valid alarm query. For example, type \InTouch!$System to query for all alarms that belong to the default $System alarm group.
7. Select **OK**.

Use query favorites to create custom saved queries

You can configure a list of query favorites for operators to select from a shortcut menu.

Make sure that you put the query favorites file in a folder that is accessible to Windows Vista or newer standard users. If WindowViewer always runs with the same user account, you can use:

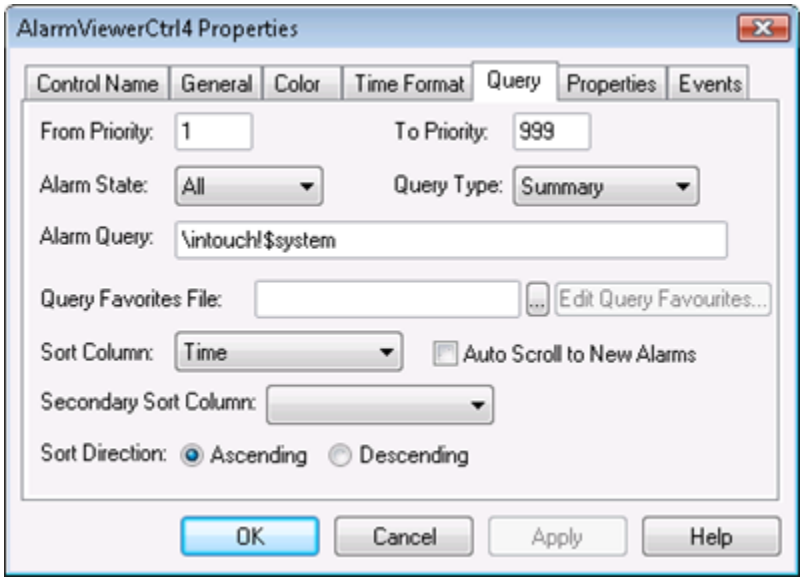
C:\Users\<username>\AppData\Local\

If different users will run WindowViewer, you can use:

C:\Users\Public\Documents\

Configure the query favorites file

1. Right-click the Alarm Viewer control and then select **Properties**. The **AlarmViewerCtrl Properties** dialog box appears.



2. Select the **Query** tab.
3. Configure the query favorites file.
   a. In the **Query Favorites File** box, type the network path and file name or select the ellipse button to browse for the file.
   b. To edit the Filter Favorites file, select the **Edit Query Favorites** button. The **Alarm Query** window opens, allowing you to add, modify, or delete filters from your favorites file. When you are done, select **OK** to save your changes and close the window.
4. Select **OK**.

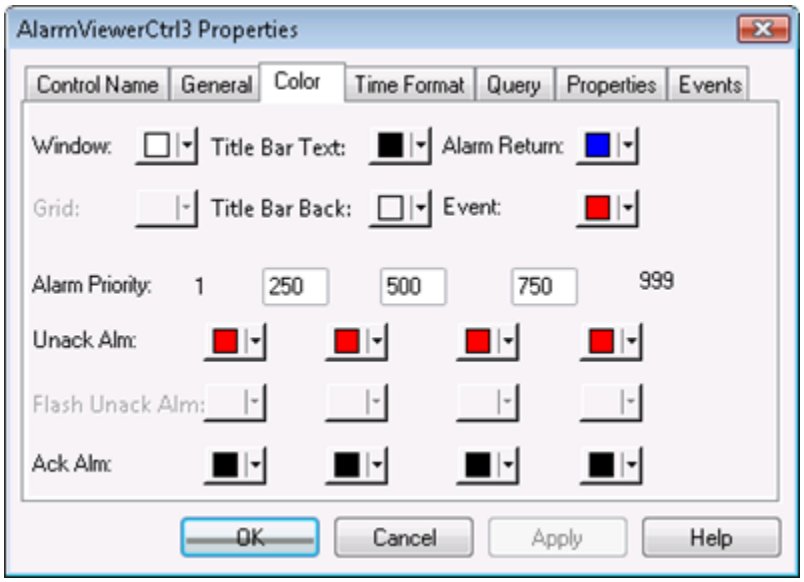
Use colors for various types of alarm records

You can set color options for different alarm states that appear in the Alarm Viewer control.

Configure the alarm display colors

1. Right-click the Alarm Viewer control and then select **Properties**. The **AlarmViewerCtrl Properties** dialog box appears.

2. Select the **Color** tab.



3. Select each color box to open the color palette. Select the color that you want to use in the palette for each of the following:

Property	Description
Window	Sets display background color.
Title Bar Text	Sets title bar text color (available only if Show Heading option is selected).
Alarm Return	Sets color of returned alarms (alarms that have returned to normal without being acknowledged).
Grid	Sets color of the grid. By default the grid is not shown. The default grid color is light gray. The color of the grid in the alarm object is automatically set to a contrasting color of the selected Window color.
Title Bar Background	Sets title bar background color (available only if the **Show Heading** option is selected).
Event	Sets color of events.

4. In the Alarm Priority boxes, type alarm priority numbers that serve as breakpoints for the different colors used to identify unacknowledged alarms, acknowledged alarms, and flashing unacknowledged alarms.
5. Select the **UnAck Alarm** and **Ack Alarm** color boxes to open the color palette. Select the color in the palette that you want to use.
6. To configure the alarm query to flash unacknowledged alarms, select the **General** tab, select the **Flash Unack Alarms** check box, then select the **Color** tab and select the **Flash Unack Alarms** color boxes. Select the color that you want to use for each alarm priority range.

Note: The Alarm Viewer control cannot show changes that occur in less than a second. If an alarm changes state twice within a second, the Alarm Viewer control does not recognize the change.

7. Select **Apply**.

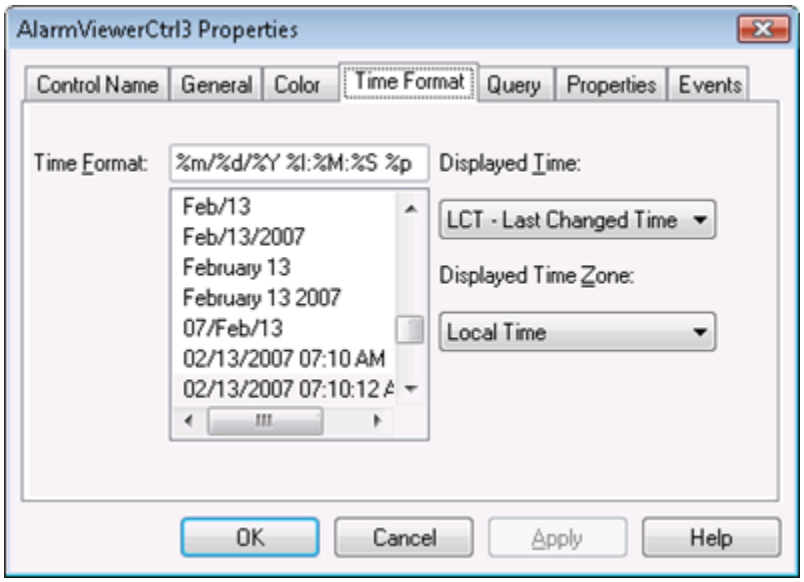
Configure the shown time format of alarm records

You can configure the time format for shown alarm records.

For the Alarm Viewer control, the original alarm time is the date/time stamp of the onset of the alarm. If tag is an I/O tag, then it is the time stamp from the I/O Server if that server is capable of passing time stamps.

Configure the time format

1. Right-click the **Alarm Viewer** control and then select **Properties**. The **AlarmViewerCtrl Properties** dialog box appears.
2. Select the **Time Format** tab.



3. In the **Time Format** list, select the desired time format. The **Time Format** box shows a set of strings consisting of characters separated by the % symbol for the format you selected.

String character	Description
d	Two-digit day of the month.
b	Three-letter month abbreviation.
Y	Four-digit year.
m	Two-digit month.
y	Two-digit year.
#x	Full day and date. For example: Friday, August 10, 2007
B	Complete month name.

String character	Description
H	Hours in 24 hour time format.
M	Minute.
p	AM or PM (for 12 hour time format).
S	Seconds.
s	Fractions of a second.
I	Hours in 12 hour time format.

4. In the **Displayed Time** list, select the shown time:

OAT	The original alarm time, which is the date/time stamp of the onset of the alarm.
LCT	The last changed time, which is the date/time stamp of the most recent change of state for the instance of the alarm: onset of the alarm, change of sub-state, return to normal, or acknowledgment.
LCT But OAT on ACK	The last changed time, but the original alarm time on acknowledge. The last changed time is used while the alarm is unacknowledged, then the original alarm time is used after the alarm has been acknowledged.

5. In the **Displayed Time Zone** list, select the time zone:

GMT	Greenwich Mean Time, also known as Coordinated Universal Time, UTC, or Zulu.
Local Time	Alarm time adjusted for the local time zone.
Origin Time	Alarm time adjusted for the time zone of the alarm source.

6. Select **Apply**.

Configure the sort order of alarm records

You can sort the alarm records in the list. By default, the Alarm Viewer control lists alarm records by time in ascending order.

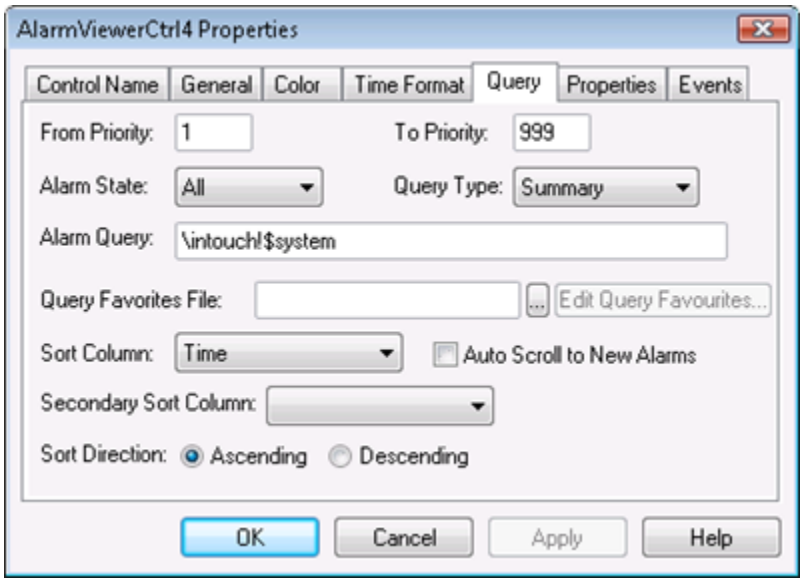
You can sort alarm records in ascending or descending order based on a primary column and an optional secondary sort column.

Configure the sort order of alarm records

1. Right-click the Alarm Viewer control, and then select **Properties**. The **AlarmViewerCtrl Properties** dialog box

appears.

2. Select the **Query** tab.



3. Select sorting options by completing the following:
   a. Select the primary sort column from the **Sort Column** list. Only visible columns appear in the **Sort Column** list. If you do not see the column you want, go to the General tab and select the column from Column Details.
   b. Select the secondary sort column from the **Secondary Sort Column** list.
   c. If you selected Time as the primary sort column, the **Auto Scroll to New Alarms** check box becomes available. Select this option if you want to automatically scroll and show new alarms as they occur.
   d. Select **Ascending** or **Descending** as the sort direction.
4. Select **Apply**.

Use an alarm viewer control at runtime

The Alarm Viewer control includes a shortcut menu that provides operators quick access to commands that can be applied to the display object of one or more selected alarms, alarm groups, tags, and priorities.

The following lists shows the commands available from the Alarm Viewer control shortcut menu:

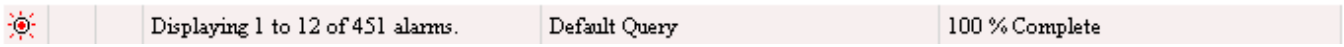
- **Ack Selected** - Acknowledges the selected alarm(s).
- **Ack Others** - A sub-menu appears that lists other acknowledgement commands.
  - Ack All - Acknowledges all the alarms in the current alarm query. Because the alarm grid has only a limited display area, the Ack All command may acknowledge alarms that are not visible in the grid.
  - Ack Visible - Acknowledges only those alarms that are currently visible in the alarm grid.
  - Ack Selected Groups - Acknowledges all alarms that have the same group name from the same provider as one or more of the selected alarms.
  - Ack Selected Tags - Acknowledges all alarms that have the same tag from the same provider and group name and having the same priority as one or more of the selected alarms.
  - Ack Selected Priorities - Acknowledges all alarms that have the same priority from the same provider and

group name as one or more of the selected alarms.

- **Suppress Selected** - The selected alarm(s) is/are suppressed.
- **Suppress Others** - A sub-menu opens that contains suppression commands.
  - Suppress All - Suppress showing current and future occurrences of all alarms.
  - Suppress Visible - Suppress showing current and future occurrences of any visible alarm.
  - Suppress Selected Groups - Suppress showing current and future occurrences of any alarm that belongs to the same groups of one or more selected alarms having the same Provider name.
  - Suppress Selected Tags - Suppress showing current and future occurrences of any alarm that belongs to the same tag name of one or more selected alarms having the same Provider name, Group name and Priority range.
  - Suppress Selected Priorities - Suppress showing current and future occurrences of any alarm that belongs to the same priorities of one or more selected alarms having the same Provider name and Group name.
  - Unsuppress All - Clears the suppression settings.
- **Query Favorites** - Shows the Alarm Query dialog box to select a previously saved alarm query. You can also add, modify and delete alarm queries.
- **Stats** - Shows the **Alarm Statistics** dialog box.
- **Suppression** - Shows the **Alarm Suppression** dialog box.
- **Freeze** - Freezes the current display.
- **Requery** - Queries the alarm provider again.
- **Sort** - Shows the **Secondary Sort** dialog box.

View status bar information

If you select the Show Status Bar option from the General properties page, a status bar appears at the bottom of an Alarm Viewer control during run time.



The status bar contains three indicators: A status message, current alarm query, and a progress bar. These indicators provide an overview of the current state of the display query and provide details about the suppression available in the Alarm Viewer control. The right pane of the status bar is red when the control is frozen and the left pane of the status bar is red when one or more alarms are suppressed. The word "suppression" is shown in the left pane when suppression is in effect.

Use query favorites at runtime

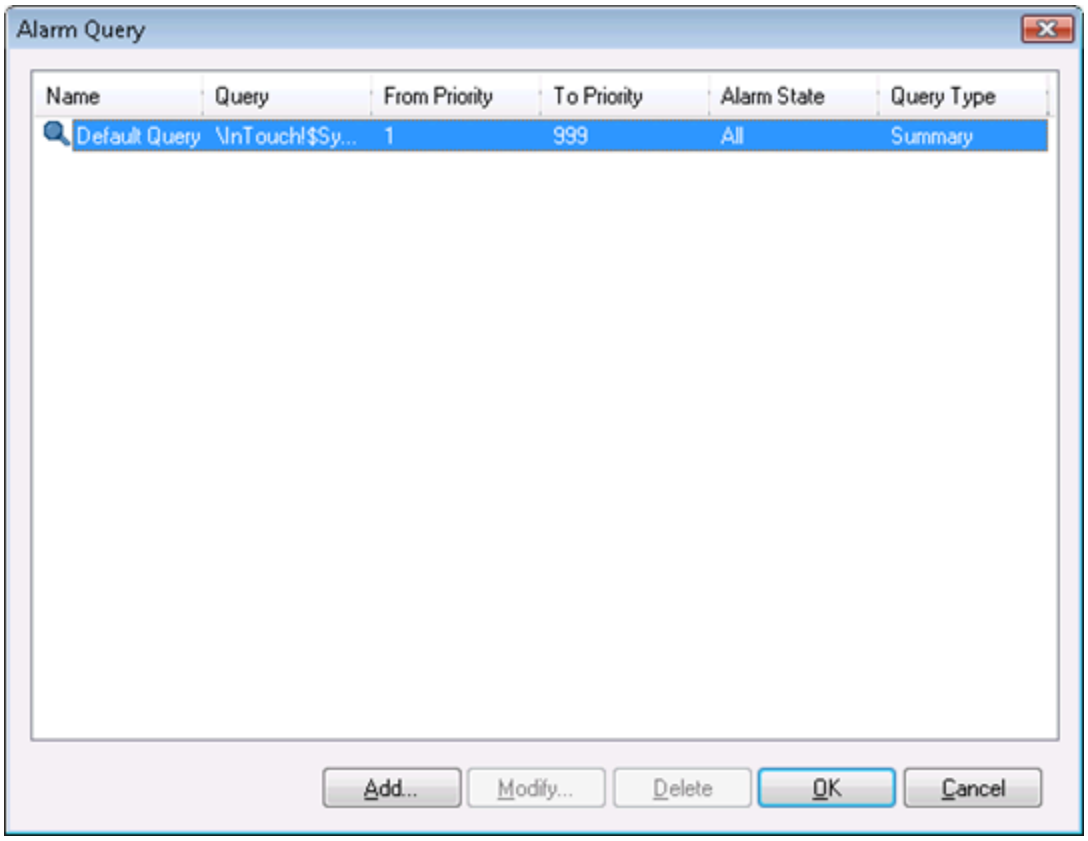
Use the Query Favorites command on the Alarm Viewer control's shortcut menu to quickly select an alarm query from a list of previously defined alarm queries. You can also create new named queries, edit an existing query, or delete an existing query.

Changes to an alarm query are not automatically applied to other Alarm Viewer controls using the same query. Deleting an alarm query does not automatically remove the query from other Alarm Viewer controls using the same query.

Note: For multi-line alarm queries appearing in the Alarm Viewer control, line separations appear as "garbage" characters. This does not affect the function.

Select an alarm query at run time

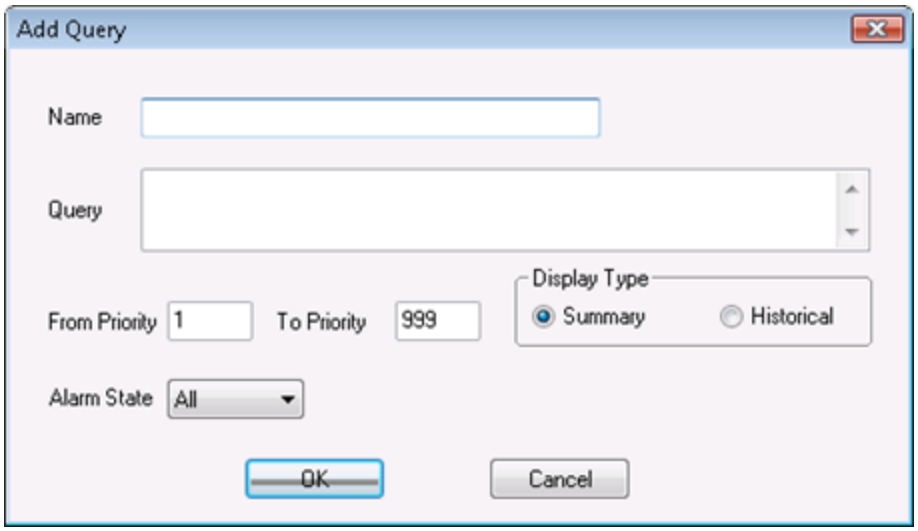
1. Right-click the Alarm Viewer control and then select **Query Favorites**. The **Alarm Query** dialog box appears.



2. Select the named query that you want to show in the list of currently defined queries.
3. Select **OK**. The Alarm Viewer control now shows alarm information retrieved by the query.

Add a new named query at run time

1. Right-click the **Alarm Viewer** control and then select **Query Favorites**.
   The **Alarm Query** dialog box appears.
2. Select **Add**.
   The **Add Query** dialog box appears.



3. Configure the query. Do the following:
   a. In the **Name** box, type the name that you want to use to identify the query.
   b. In the **Query** box, type the sets of InTouch alarm queries that you want to perform. You can specify one or more Alarm Providers and groups.
   c. In the **From Priority** box, type the minimum alarm priority value (1 to 999).
   d. In the **To Priority** box, type the maximum alarm priority value (1 to 999).
   e. Select the **Alarm State** arrow and select the alarm state that (**All, Ack**, **Unack**) you want to use in the alarm query.
   f. In the **Display Type** area, select **Summary** or **Historical** for the type of records you want to query.
4. Select **OK** to close the **Add Query** dialog box.
5. Select **OK** in the **Alarm Query** dialog box to add the query to your favorites.

Modify an existing named query at run time

1. Right-click the **Alarm Viewer** control and then select **Query Favorites**.
   The **Alarm Query** dialog box appears.
2. Select the named query that you want to modify in the list of currently defined queries.
3. Select **Modify**.
   The **Modify Query** dialog box appears.
4. Make the necessary modifications and then select **OK**.
5. Select **OK** in the **Alarm Query** dialog box.

Delete an existing named query at run time

1. Right-click the **Alarm Viewer** control and then select **Query Favorites**.
   The **Alarm Query** dialog box appears.
2. Select the named query that you want to delete in the list of currently defined queries.
3. Select **Delete**. When a message appears, select **Yes**.
4. Select **OK** in the Alarm Query dialog box.

Use Alarm Viewer control ActiveX properties

You can set the value an Alarm Viewer control property directly using a script or you can assign it to an InTouch tag or I/O reference. For more information about setting properties, see Scripting ActiveX Controls in *AVEVA™ InTouch HMI Application Development Guide.*

The following table lists the Alarm Viewer control properties.

Property	Type	Purpose
AckAllMenu	Discrete	Enables/disables **Ack All** menu item.
AckAlmColorRange1	Integer	Sets color to be used to show acknowledged alarms with priorities in the range 1 to ColorPriorityRange1. The default priority range is 1 to 250.
AckAlmColorRange2	Integer	Sets color to be used to acknowledged alarms with priorities in the range ColorPriorityRange1 to ColorPriorityRange2. The default priority range is 250 to 500.
AckAlmColorRange3	Integer	Sets color to be used to acknowledged alarms with priorities in the range ColorPriorityRange2 to ColorPriorityRange3. The default priority range is 500 to 750.
AckAlmColorRange4	Integer	Sets color to be used to acknowledged alarms with priorities in ColorPriorityRange3 to 999. The default priority range is 750 to 999.
AckOthersMenu	Discrete	Enables/disables **Ack Others** menu item.
AckSelectedGroupsMenu	Discrete	Enables/disables **Ack Selected Groups** menu item.
AckSelectedMenu	Discrete	Enables/disables **Ack Selected** menu item.
AckSelectedPrioritiesMenu	Discrete	Enables/disables **Ack Selected Priorities** menu item.

Property	Type	Purpose
AckSelectedTagsMenu	Discrete	Enables/disables **Ack Selected Tags** menu item.
AckVisibleMenu	Discrete	Enables/disables **Ack Visible** menu item.
AlarmQuery	Message	Sets the initial alarm query. This field accepts text only; it does not accept tags. The following example uses the full path to the alarm group: \\Node\InTouch!Group This example uses the full path to the local alarm group: \InTouch!Group This example uses another Group List: GroupList
AlarmState	Message	Default alarm state to query (All, UnAck, Ack).
AlmRtnColor	Integer	Sets the color for alarms that have returned to normal and were unacknowledged. This color is also used for alarms that returned to normal from the acknowledged state but the acknowledgement state transition was not observed.
AutoScroll	Discrete	If the user scrolls the list from the beginning, this automatically jumps to the new alarm. New alarms are defined as those that are not currently shown within the display object.
ColorPriorityRange1	Integer	Sets the boundary of the priority range in which alarms are to be shown. The value of this property must be greater than one and less than the value for ColorPriorityRange2.

Property	Type	Purpose
ColorPriorityRange2	Integer	Sets the boundary of the priority range in which alarms are to be shown. The value of this property must be greater than the value for ColorPriorityRange1 and less than the value for ColorPriorityRange3.
ColorPriorityRange3	Integer	Sets the boundary of the priority range in which alarms are to be shown. The value of this property must be greater than the value of ColorPriorityRange2 and less than 999.
ColumnResize	Discrete	Returns or sets a value that determines whether the columns can be resized at run time.
CustomMessage	Message	The default message to show when there are no alarms.
DefaultAckComment	Message	Used as a comment when the alarm is acknowledged and when the "UseDefaultAckComment" is TRUE. Otherwise, the user is prompted to enter a comment.
DisplayedTime	Message	Shows the alarm message time. The values can only be "OAT" or "LCT" or "LCT But OAT on ACK."
DisplayedTimeZone	Message	Gets or sets the current time zone string. The values can only be "GMT" or "Origin Time" or "Local Time."
EventColor	Integer	Sets color of events.
ExtendedSelection	Discrete	Allows you to select multiple alarms by holding down the Ctrl or Shift key in conjunction with the mouse button. The default is to toggle selection of alarms by simply selecting on them (available only if the Row Selection check box is selected).

Property	Type	Purpose
FlashUnAckAlarms	Discrete	Enables or disables the flashing of unacknowledged alarms. It takes a discrete input value of 1 or 0. If this property is set to 1, unacknowledged alarms flash once per second. If this property is set to 0, unacknowledged alarms do not flash. This property corresponds to the **Flash Unack Alarms** check box on the Alarm Viewer control **General** tab.
FlashUnackAlmColorRange1	Integer	Sets the flashing color for unacknowledged alarms belonging to Alarm Priority Range 1.
FlashUnackAlmColorRange2	Integer	Sets the flashing color for unacknowledged alarms belonging to Alarm Priority Range 2.
FlashUnackAlmColorRange3	Integer	Sets the flashing color for unacknowledged alarms belonging to Alarm Priority Range 3.
FlashUnackAlmColorRange4	Integer	Sets the flashing color for unacknowledged alarms belonging to Alarm Priority Range 4.
Font	None	Sets the font for the records and the header in the control.
FreezeMenu	Discrete	Enables/disables the **Freeze** menu item.
FromPriority	Integer	Sets the low priority value of the default query.
GridColor	Integer	Sets the color of the background grid.

Property	Type	Purpose
NewAlarmEventMode	Integer	Controls the triggering of the NewAlarm event. 0 = The NewAlarm event can not be triggered. (Default). 1 = The NewAlarm event is active. 2 = The NewAlarm event is active and continually triggers when at least one new unacknowledged alarm arrives.
QueryFavoritesFile	Message	Returns or sets the query favorites file name.
QueryFavoritesMenu	Discrete	Enables/disables the Query Favorites menu item.
QueryName	String	Returns the current query name.
QueryStartup	Discrete	Automatically begins updating the grid using default query properties, if set. Otherwise, you must run the ApplyDefaultQuery or ApplyQuery method in a script to update the grid.
QueryType	Message	Sets the display type as either Summary or Historical.
RequeryMenu	Discrete	Enables/disables Requery shortcut menu item.
RetainSuppression	Discrete	Retains alarm suppression between alarm queries when the alarm query is changed.
RowSelection	Discrete	Allows user to select alarms during run time.
SecondarySortColumn	Message	Returns or sets the current secondary sort column.
SelectedCount	Integer	Returns the total number of selected alarms.
ShowContextMenu	Discrete	Enables the activation of the shortcut menu.

Property	Type	Purpose
ShowGrid	Discrete	Returns or sets a value that determines whether the grid lines are shown in the control.
ShowHeading	Discrete	Shows the title bar of the control.
ShowMessage	Discrete	Returns or sets a value that determines whether error messages are shown for the control.
ShowStatusBar	Discrete	Gets or sets a value that determines whether the status bar is shown.
SilentMode	Discrete	Gets or sets a value that determines whether the control is in silent mode.
SortColumn	Message	Gets or sets the current sort column.
SortMenu	Discrete	Enables/disables the **Sort** menu item.
SortOrder	Discrete	Gets or sets the sort direction. Possible values are "Ascending" and "Descending," represented as 0 and 1 respectively.
StatsMenu	Discrete	Enables/disables the **Stats** menu item.
SuppressAllMenu	Discrete	Enables/disables the **Suppress All** menu item.
SuppressedAlarms	Integer	Gets the total number of suppressed alarms.
SuppressionMenu	Discrete	Enables/disables the **Suppression** menu item.
SuppressOthersMenu	Discrete	Enables/disables the **Suppress Others** menu item.
SuppressSelectedGroupsMenu	Discrete	Enables/disables the **Suppress Selected Groups** menu item.

Property	Type	Purpose
SuppressSelectedMenu	Discrete	Enables/disables the **Suppress Selected** menu item.
SuppressSelectedPrioritiesMenu	Discrete	Enables/disables the **Suppress Selected Priorities** menu item.
SuppressSelectedTagsMenu	Discrete	Enables/disables **Suppress Selected Tags** menu item.
SuppressVisibleMenu	Discrete	Enables/disables the **Suppress Visible** menu item.
TimeFormat	Message	Sets the format of the alarm time stamps.
TitleBackColor	Integer	Sets title bar background color.
TitleForeColor	Integer	Sets title bar foreground color.
ToPriority	Integer	Sets the maximum priority for the alarm query.
TotalAlarms	Integer	Gets the number of alarms.
UnackAlarms	Integer	Gets the total number of unacknowledged alarms.
UnAckAlmColorRange1	Integer	Sets the color to be used to show unacknowledged alarms with priorities in the range of 1 to ColorPriorityRange1.
UnAckAlmColorRange2	Integer	Sets the color to be used to show unacknowledged alarms with priorities in the range of ColorPriorityRange1 to ColorPriorityRange2.
UnAckAlmColorRange3	Integer	Sets the color to be used to show unacknowledged alarms with priorities in the range of ColorPriorityRange2 to ColorPriorityRange3.
UnAckAlmColorRange4	Integer	Sets the color to be used to show unacknowledged alarms with priorities in the range of ColorPriorityRange3 to 999.

Property	Type	Purpose
UnsuppressAllMenu	Discrete	Enables/disables Unsuppress All menu item.
UseDefaultAckComment	Discrete	If set to True, the default acknowledgement comment is used when the alarm is acknowledged. Otherwise, the operator is prompted to enter a comment.
WindowColor	Integer	Sets the grid background color.

Configure colors for ActiveX Controls

You specify a color as an integer value. The alarm ActiveX controls use the ABGR model for specifying color as a 32-bit integer, where:

A = Transparency

B = Blue

G = Green

R = Red

Transparency is not supported by the alarm ActiveX controls. Any values for the upper 8 bits are ignored.

For example, to set the color to Blue, use the following ABGR values:

A = 0

B = 255

G = 0

R = 0

The hexadecimal value for this color is 0x00FF0000. The decimal value is 16711680.

The following table shows examples of colors you can use:

Color	Hexidecimal Value	Decimal Value
White	0x00FFFFFF	16777215
Black	0x00000000	0
Blue	0x00FF0000	16711680
Red	0x000000E1	225
Green	0x0000FF00	65280

Use Alarm Viewer control ActiveX methods

You use the Alarm Viewer control ActiveX methods in scripts to:

- Acknowledge alarms.
- Suppress alarms.
- Get information about an alarm.
- Run alarm queries.
- Move and freeze the display.
- Sort alarm records.
- Select specific alarms.
- Show the shortcut menu, **About** dialog box, and **Alarm Statistics** dialog box.

For more information about calling methods, see Scripting ActiveX Controls in *AVEVA™ InTouch HMI Application Development Guide.*

Acknowledge alarms during runtime

Use the following methods to acknowledge alarms during run time.

- AckSelected() method
- AckAll() method
- AckVisible() method
- AckSelectedGroup() method
- AckSelectedTag() method
- AckSelectedPriority() method
- AckGroup() method
- AckPriority() method
- AckTag() method

AckSelected() method

Acknowledges alarms that are selected in the Alarm Viewer control at run time.

Syntax

```
Object.AckSelected (Comment)
```

Parameter

Comment

Alarm acknowledgment comment.

Example

Tag1 is defined as a message tag and the name of the control is AlarmViewerCtrl1.

```
Tag1 = "Alarm Comment";
#AlarmViewerCtrl1.AckSelected (Tag1);
```

AckAll() method

Acknowledges all the alarms in the current alarm query. Because the Alarm Viewer control has a limited display area, the AckAll() method can also acknowledge alarms not shown in the display.

Syntax

```
Object.AckAll (Comment)
```

Parameter

Comment

Alarm acknowledgment comment.

Example

Tag1 is defined as a message tag and the name of the control is AlarmViewerCtrl1.

```
Tag1 = "Alarm Comment";
#AlarmViewerCtrl1.AckAll (Tag1);
```

AckVisible() method

Acknowledges only those alarms that are currently visible in the Alarm Viewer control.

Syntax

```
Object.AckVisible (Comment)
```

Parameter

Comment

Alarm acknowledgment comment.

Example

Tag1 is defined as a message tag and the name of the control is AlarmViewerCtrl1.

```
Tag1 = "Alarm Comment";
#AlarmViewerCtrl1.AckVisible (Tag1);
```

AckSelectedGroup() method

Acknowledges all alarms that have the same group name as one or more selected alarms.

Syntax

```
Object.AckSelectedGroup (Comment)
```

Parameter

Comment

Alarm acknowledgment comment.

Example

Tag1 is defined as a message tag and the name of the control is AlarmViewerCtrl1.

```
Tag1 = "Alarm Comment";
#AlarmViewerCtrl1.AckSelectedGroup (Tag1);
```

AckSelectedTag() method

Acknowledges all alarms that have the same tag, group name and priority as one or more of the selected alarms.

Syntax

```
Object.AckSelectedTag (Comment)
```

Parameter

Comment

Alarm acknowledgment comment.

Example

Tag1 is defined as a message tag and the name of the control is AlarmViewerCtrl1.

```
Tag1 = "Alarm Comment";
#AlarmViewerCtrl1.AckSelectedTag (Tag1);
```

AckSelectedPriority() method

Acknowledges all alarms that have the same priority range as one or more of the selected alarms.

Syntax

```
Object.AckSelectedPriority (Comment)
```

Parameter

Comment

Alarm acknowledgment comment.

Example

Tag1 is defined as a message tag and the name of the control is AlarmViewerCtrl1.

```
Tag1 = "Alarm Comment";
#AlarmViewerCtrl1.AckSelectedPriority (Tag1);
```

AckGroup() method

Acknowledges all alarms for a given group name and provider.

Syntax

```
Object.AckGroup(ApplicationName, GroupName, Comment)
```

Parameter

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the group. For example, Turbine.

Comment

Alarm acknowledgment comment.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.AckGroup ("\Intouch", "Turbine", "Turbine acknowledgement Comment");
```

AckPriority() method

Acknowledges all of the alarms specified priority range having same provider name and group name.

Syntax

```
Object.AckPriority(ApplicationName, GroupName, FromPriority, ToPriority, Comment)
```

Parameter

ApplicationName

The name of the application. For example, \\node1\Intouch

GroupName

The name of the group. For example, Turbine.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

Ending priority of alarms. For example, 900.

Comment

Alarm acknowledgment comment.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.AckPriority ("\Intouch", "Turbine", 100, 900, "Turbine acknowledgement
Comment");
```

AckTag() method

Acknowledges the alarms of the given tag name having the same provider name and group name within the given priority range.

Syntax

```
Object.AckTag(ApplicationName, GroupName, tag, FromPriority, ToPriority, Comment)
```

Parameter

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the group. For example, Turbine.

tag

The name of the alarm tag. For example, Valve1.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

Ending priority of alarms. For example, 900.

Comment

Alarm acknowledgment comment.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.AckTag ("\Intouch", "Turbine", "Valve1", 100, 900, "Turbine
acknowledgement Comment");
```

Suppress alarms during runtime

Use the following methods to suppress alarms during run time:

- ShowSuppression() Method
- SuppressSelected() method
- SuppressAll() method
- SuppressVisible() method
- SuppressSelectedGroup() method
- SuppressSelectedTag() method
- SuppressSelectedPriority() method
- UnSuppressAll() method
- SuppressGroup() method
- SuppressPriority() method
- SuppressTag() method

ShowSuppression() Method

Shows the suppression dialog box, which contains all suppressed alarms.

Syntax

```
Object.ShowSuppression()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.ShowSuppression();
```

SuppressSelected() method

Suppresses showing current and future occurrences of the selected alarm(s).

Syntax

```
Object.SuppressSelected()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressSelected();
```

SuppressAll() method

Suppresses showing current and future occurrences of all active alarms.

Syntax

```
Object.SuppressAll()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressAll();
```

SuppressVisible() method

Suppresses showing current and future occurrences of any visible alarm.

Syntax

```
Object.SuppressVisible()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressVisible();
```

SuppressSelectedGroup() method

Suppresses showing current and future occurrences of any alarm that belongs to the same Group and Provider of one or more selected alarms.

Syntax

```
Object.SuppressSelectedGroup()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressSelectedGroup();
```

SuppressSelectedTag() method

Suppresses showing current and future occurrences of any alarm that belongs to the same tag name of one or more selected alarms having the same Group name, Provider name, and Priority range.

Syntax

```
Object.SuppressSelectedTag()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressSelectedTag();
```

SuppressSelectedPriority() method

Suppresses showing current and future occurrences of any alarm that belongs to the same priority range of one or more selected alarms.

Syntax

```
Object.SuppressSelectedPriority()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressSelectedPriority();
```

UnSuppressAll() method

Clears alarm suppression.

Syntax

```
Object.UnSuppressAll()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.UnSuppressAll();
```

SuppressGroup() method

Suppresses showing current and future occurrences of any alarm that belongs to a given Group name.

Syntax

```
Object.SuppressGroup(ApplicationName, GroupName)
```

Parameter

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the group. For example, Turbine.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressGroup ("\Intouch", "Turbine");
```

SuppressPriority() method

Suppresses showing current and future occurrences of any alarm of the specified priority range, having the same Provider name and Group name.

Syntax

```
Object.SuppressPriority(ApplicationName, GroupName, FromPriority, ToPriority)
```

Parameter

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the group. For example, Turbine.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

Ending priority of alarms. For example, 900.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressPriority ("\Intouch", "Turbine", 100, 900);
```

SuppressTag() method

Suppresses showing current and future occurrences of any alarm emitted by a given tag name or group name and in the specified priority range.

Syntax

```
Object.SuppressTag(ApplicationName, GroupName, tag, FromPriority, ToPriority)
```

Parameter

ApplicationName

The name of the Application for example, \\node1\Intouch.

GroupName

The name of the group. For example, Turbine.

tag

The name of the alarm tag. For example, valve 1.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

Ending priority of alarms. For example, 900.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SuppressTag ("\Intouch", "Turbine", "Valve1", 100, 900);
```

Retrieve information about a particular alarm

Use the GetItem() function to retrieve information about an alarm.

GetItem() method

Returns the data at a specified row and column as a string.

Syntax

```
Object.GetItem(Integer, Message)
```

Parameters

Integer

An integer expression that evaluates to a specific row in the control.

Message

A string expression that evaluates to the column name in the control.

Example

The name of the control is AlmDbView1 and tag is defined as a Message tag.

```
tag = #AlmDbView1.GetItem(1, "Group");
```

Run alarm queries

Use the following methods to run queries.

- ShowQueryFavorites() method
- Requery() method
- ApplyQuery() method
- ApplyDefaultQuery() method
- SetQueryByName() method

ShowQueryFavorites() method

Shows the **Query Favorites** dialog box if the QueryFavoritesFile property contains a valid query favorite file name in .xml format.

Syntax

```
Object.ShowQueryFavorites()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.ShowQueryFavorites();
```

Requery() method

Queries the alarm provider again.

Syntax

```
Object.Requery()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.Requery();
```

ApplyQuery() method

Performs the query as specified by its parameters Alarm Query, From and To Priorities, State of alarms to query for. and the type of alarms to retrieve.

Syntax

```
Object.ApplyQuery(AlarmQuery, FromPriority, ToPriority, State, Type)
```

Parameter

AlarmQuery

The alarm query. For example: \InTouch!$System

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

Ending priority of alarms. For example, 900.

State

Specifies type of alarms to show. For example, "UnAck" or message tag. Valid states are All, UnAck, or Ack.

Type

Specifies type of query for example, Historical (Historical alarms) or Summary (Summary alarms).

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.ApplyQuery ("\InTouch!$System",100,900,"All", "Historical");
```

ApplyDefaultQuery() method

Performs a query using the FromPriority, ToPriority, AlarmState, QueryType, and AlarmQuery properties as specified at design time. The default properties can only be changed at development time and are not overwritten by other alarm queries.

Syntax

```
Object.ApplyDefaultQuery()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.ApplyDefaultQuery();
```

SetQueryByName() method

Sets the current query as specified by the query name passed. The query must be in the query favorites file.

Syntax

```
Object.SetQueryByName(QueryName)
```

Parameter

QueryName

The name of the query as created by using query favorites. For example, Turbine Queries.

Example

The name of the control is AlarmTreeViewerCtrl1.

```
#AlarmTreeViewerCtrl1.SetQueryByName("Turbine Queries");
```

Move and Freeze the display

Use the following functions to move or freeze the display:

- MoveWindow() method
- FreezeDisplay() method

MoveWindow() method

Scrolls the alarms in the control in a specified way.

Syntax

```
Object.MoveWindow(Option, Repeat)
```

Parameters

Option

The type of action to perform.

Type	Description
LineDn	Line down. The Repeat parameter controls the number of lines to be scrolled.
LineUp	Line up. The Repeat parameter controls the number of lines to be scrolled.

Type	Description
PageDn	Page down. The Repeat parameter controls the number of pages to be scrolled.
PageUp	Page up. The Repeat parameter controls the number of pages to be scrolled.
Top	To the top of the control.
Bottom	To the bottom of the control.
PageRt	Page to the right. The Repeat parameter controls the number of pages to be scrolled.
PageLf	Page to the left. The Repeat parameter controls the number of pages to be scrolled.
Right	Scrolls right. The Repeat parameter controls the number of columns to be scrolled.
Left	Scrolls left. The Repeat parameter controls the number of columns to be scrolled.
Home	Scrolls to the top row and left most column of the control.

Repeat

The number of times this operation should be repeated.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.MoveWindow ("Bottom", 0);
#AlarmViewerCtrl1.MoveWindow ("LineUp", 3);
#AlarmViewerCtrl1.MoveWindow ("PageLf", 7);
```

FreezeDisplay() method

Freezes the display.

Syntax

```
Object.FreezeDisplay(Freeze)
```

Parameter

Freeze

True = Freezes the display.

False = Unfreezes the display.

Example

Tag1 is defined as Memory discrete tag and the name of the control is AlarmViewerCtrl1.

```
Tag1 = 1;
#AlarmViewerCtrl1.FreezeDisplay(Tag1);
```

Sort alarm records

Use the following functions to sort alarm records:

- ShowSort() Method
- SetSort() method

ShowSort() Method

Shows the **Secondary Sort** dialog box if the SortMenu property is enabled.

Syntax

```
Object.ShowSort()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.ShowSort();
```

SetSort() method

Sets the sort criteria as specified by the SortColumn and SortOrder properties.

Syntax

```
Object.SetSort()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SetSort();
```

Show additional information

Use the following functions to show the About dialog box and the Alarm Statistics dialog box:

- AboutBox() Method
- ShowStatistics() method

AboutBox() Method

Shows the **About** dialog box.

Syntax

```
Object.AboutBox()
```

Example

The name of the control is AlarmPareto1.

```
#AlarmPareto1.AboutBox();
```

ShowStatistics() method

Shows the **Alarm Statistics** dialog box.

Syntax

```
Object.ShowStatistics()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.ShowStatistics();
```

Select specific alarms

Use the following functions to select specific alarms:

- SelectGroup() method
- SelectPriority() method
- SelectTag() method
- SelectAll() method
- SelectItem() method
- UnSelectAll() method

SelectGroup() method

Selects all of the alarms that contain the same alarm group name and provider name.

Syntax

```
Object.SelectGroup(ApplicationName, GroupName)
```

Parameter

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the group. For example, Turbine.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SelectGroup ("\Intouch", "Turbine");
```

SelectPriority() method

Selects all of the alarms that are of the specified priority range, having the same provider name and group name.

Syntax

```
Object.SelectPriority(ApplicationName, GroupName, FromPriority, ToPriority)
```

Parameter

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the group. For example, Turbine.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

Ending priority of alarms. For example, 900.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SelectPriority ("\Intouch", "Turbine", 100, 900);
```

SelectTag() method

Selects all of the alarms from a specific Provider/Group/Tag. You can also specify a Priority range, or use 1-999.

Syntax

```
Object.SelectTag(ApplicationName, GroupName, tag, FromPriority, ToPriority)
```

Parameter

ApplicationName

The name of the Application for example, \\node1\Intouch

GroupName

The name of the group. For example, Turbine.

tag

The name of the alarm tag. For example, valve 1.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

Ending priority of alarms. For example, 900.

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.SelectTag ("\Intouch", "Turbine", "Valve1",100, 900);
```

SelectAll() method

Toggles the selection of all the alarms in a display. Because the alarm display has only a limited display area, the SelectAll() function may select alarms that are not visible in the display.

Syntax

```
Object.SelectAll()
```

Example

```
#AlarmViewerCtrl1.SelectAll();
```

SelectItem() method

Toggles the selection of an alarm record at a given row.

Syntax

```
Object.SelectItem(RowNumber)
```

Parameter

RowNumber

An integer value that is the row number for the alarm record to select. The first row in the control is 0.

Example

The name of the control is AlarmViewerCtrl1 and Tag1 is defined as a memory integer. This toggles the selection of the tenth alarm record in the Alarm Viewer control.

```
Tag1 = 9;
#AlarmViewerCtrl1.SelectItem (Tag1);
```

UnSelectAll() method

Unselects all the selected records.

Syntax

```
Object.UnSelectAll()
```

Example

The name of the control is AlarmViewerCtrl1.

```
#AlarmViewerCtrl1.UnSelectAll();
```

Show the context menu at runtime

Use the ShowContext() method to show the shortcut menu at run time.

ShowContext() method

Shows the shortcut menu if any one of RefreshMenu or ResetMenu or SortMenu property is enabled.

Syntax

```
Object.ShowContext()
```

Example

The name of the control is AlmDbView1.

```
#AlmDbView1.ShowContext();
```

Handle errors when using methods and properties

You can use the SilentMode property to hide errors during run time. If the SilentMode property is set to 1, the Alarm Viewer control does not show error messages in during run time. If it is set to 0, the Alarm Viewer control shows error messages. Error messages are always sent to the Operations Control Log viewer.

Use ActiveX events to trigger scripts

You can assign QuickScripts to Alarm Viewer control events, such as a mouse click or double-click. When the event occurs, the QuickScript runs.

The Alarm Viewer control supports the following events:

- Select
- Double-click
- NewAlarm
- ShutDown
- StartUp

The Select event has one parameter called ClicknRow, which identifies the row that is selected at run time.

The Double-click event has one parameter called DoubleclicknRow, which identifies the row that is double-clicked at run time.

Select and Double-click events are zero-based. When select and/or DoubleClick events are published for the user, the row count in the display starts with 0.

Note: The Alarm Viewer control ignores the user interface methods when they are called from StartUp event because the control is not visible yet. These include: ShowSort(), ShowContext(), GetSelectedItem(), GetNext(), GetPrevious() and AboutBox().

For more information about scripting ActiveX events, see Scripting ActiveX Controls in *AVEVA™ InTouch HMI Application Development Guide.*

Run a script when a new alarm is detected

You can configure the Alarm Viewer control to run an ActiveX event script when the Alarm Viewer control detects a new unacknowledged alarm (that is, any alarm that transitions from a normal state to an unacknowledged state and meets the control's query and priority filtering criteria).

The NewAlarmEventMode property controls triggering the NewAlarm event.

- If you set NewAlarmEventMode property to 0, the NewAlarm event does not trigger. This is the default.
- If you set the NewAlarmEventMode property to 1, and a new alarm occurs:
  - An event is triggered.
  - The ActiveX event script associated with the NewAlarm event runs.
  - The NewAlarmEventMode property is set to 0.

    You must change the NewAlarmEventMode property back to 1 to process subsequent events.

    Use this setting if the application performs an action that should not be performed again until the

condition has been corrected or acknowledged. For example, when the event triggers, the ActiveX script can play an alarm sound until the alarm is acknowledged. Then the alarm sound can play again the next time a new alarm is received.

- If you set the NewAlarmEventMode property to 2, the NewAlarm event is active and continually triggers when at least one new unacknowledged alarm arrives. You do not need to change the value to process subsequent events. The new event triggers at most one time per second, regardless of how many additional new alarms arrive in the same second.

Acknowledging alarms in real time

When tag data transitions from a normal to an alarm state, a new instance of its alarm is generated. The InTouch Distributed Alarm system tracks each alarm instance through the following states:

- When the tag first enters the alarmed state
- When the alarm makes a sub-state transition, if the alarm is a multi-state alarm
- When the alarm returns to normal
- Whether the alarm is waiting for an acknowledgment
- When the alarm is acknowledged

The life cycle of an alarm instance ends when the tag value associated with the alarm returns to a normal, unalarmed state. A subsequent transition to the alarmed state generates a new alarm instance.

Understand alarm acknowledgement models

The InTouch HMI supports three models of alarm acknowledgement:

- For condition-oriented alarms, an acknowledgment counts against all entries into the alarmed state up to the time of the acknowledgment.
- For expanded summary alarms, an acknowledgment is only for a particular transition, whether to an alarmed state, to a sub-state, or a return to normal. All transitions into different alarm sub-states must be acknowledged before the overall alarm is considered acknowledged.
- For event-oriented alarms (as in OPC), an acknowledgment is accepted only if it refers to the most recent activation event.

Condition acknowledgement alarm model

For condition-oriented alarms, an acknowledgment counts against all entries into the alarmed state up to the time of the acknowledgment.

Acknowledgment is for the instance of the alarm. An alarm instance waits for an acknowledgement when it first enters the alarmed state. If the alarm is acknowledged and subsequently transitions to a new alarmed sub-state (for example from "Hi" to HiHi"), it begins waiting for another acknowledgement. Whenever the acknowledgement is received, it is accepted and applies to all transitions of the alarm that have occurred so far.

The alarm is considered acknowledged when the most recent instance is acknowledged.

Expanded summary alarm model

For expanded summary alarms, an acknowledgment is only for a particular transition, whether to an alarmed state, to a sub-state, or a return to normal. All transitions into different alarm sub-states must be acknowledged before the overall alarm is considered acknowledged.

The initial entry to the alarmed state must be acknowledged, and the return to normal must also be separately acknowledged.

Any transition to a new alarm sub-state is treated as a new occurrence that must be acknowledged, and whose return to normal must also be acknowledged. Sub-state transitions are treated as belonging to a "return to normal group," starting with the first entry into an alarmed state when the item was previously normal.

If the item returns to normal and subsequently enters the alarmed state again, a new return to normal group is created.

Each transition must be acknowledged individually and explicitly; and the alarm is considered acknowledged only when the item has returned to normal and all transitions in all pending return to normal groups have been acknowledged.

Note: The term "summary" means "awaiting acknowledgment." Alarms in this model are also known as "ring-back" alarms.

Expanded summary alarm records

For expanded summary alarms, when an alarm occurs, a record is generated in the alarm display object showing that an alarm condition has occurred. The record shows the date and time stamp of the alarm. This record remains visible until an operator acknowledges the alarm and a return-to-normal state has occurred. If the return-to-normal state occurs before the alarm is acknowledged, then two records are shown in the alarm object.

For example, a boiler’s temperature exceeds the high limit state and triggers an alarm. Later, the boiler returns to its normal temperature range before an operator can acknowledge the alarm. The alarm system generates a record that a high limit alarm occurred and an another record indicating the alarm was not acknowledged.

Using expanded summary alarms

When you define a tag and select **Expanded Summary** as its acknowledgement model, an operator must acknowledge that an alarm occurred even if the state triggering the alarm has returned to normal. Acknowledging an alarm changes the color of the alarm entry but does not change the time stamp. Alarms only clear from the display when they are acknowledged and the alarm has returned to a normal state.

Note: When you define a tag with the expanded summary acknowledgement mode, the **RTN Implies ACK** option in the Alarm Properties dialog box does not apply to the tag.

Event-based alarm model

For event-oriented alarms (as in OPC), an acknowledgment is accepted only if it refers to the most recent activation event.

An alarm instance begins waiting for an acknowledgement when it first enters the alarmed state. If the instance is acknowledged and subsequently transitions to a new alarmed sub-state, it begins waiting for another acknowledgement. Each subsequent transition is assigned a sequence number, and the acknowledgement must

have attached to it the sequence number of the transition to which it is responding.

The acknowledgement is accepted only if it is for the most recent transition. If it is accepted, it applies to all transactions of the alarm that have occurred so far. The alarm is considered acknowledged when the most recent instance is acknowledged.

A rejected acknowledgement may be logged for diagnostic purposes, but is not otherwise tracked in the system.

The event-oriented model ensures that if an alarm is changing between different states, the acknowledgement corresponds to current information. In systems with small latency times, this may look just like condition-oriented alarms. In other environments, such as the Internet, the features of this model may become important.

Check the acknowledgement model of a tag at runtime

Use the .AlarmAckModel dotfield to monitor the acknowledgement model associated with a tag.

.AlarmAckModel dotfield

Monitors the acknowledgment model associated with a tag as follows:

0 = condition (default)

1 = event oriented

2 = expanded summary

Category

Alarms

Usage

```
Tagname.AlarmAckModel
```

Parameter

Tagname

Any discrete, integer, real, indirect discrete and analog tag.

Remarks

This dotfield defaults to 0 (condition acknowledgment model).

Data Type

Analog (read-only)

Valid Values

0, 1 or 2

Example

The body of this IF-THEN statement is processed if the PumpStation tag is associated with an event alarm:

```
IF (PumpStation.AlarmAckModel == 1) THEN
   MyAlarmMessage="PumpStation is an Event alarm";
ENDIF;
```

See Also

.Alarm, .Ack, .UnAck, .AckDev, .AckDSC, .AckROC

Use dotfields to acknowledge alarms

You can create scripts that use dotfields to acknowledge all current alarms, selected alarms, or only alarms of a specific type.

Acknowledge alarms or alarm groups

You can create scripts that use the following dotfields to acknowledge alarms of specified local tags or alarms within specified alarm groups.

- .Ack dotfield
- .UnAck dotfield

You cannot acknowledge alarms that originate from the Application Server using these dotfields.

To acknowledge all local alarms of the running InTouch application, use the $System alarm group in combination with the appropriate .Ack dotfield.

.Ack dotfield

Monitors or controls the alarm acknowledgment status of all types of local alarms.

Category

Alarms

Usage

```
TagName.Ack=1;
```

Parameter

TagName

Any discrete, integer, real, indirect discrete and analog tag, or alarm group.

Remarks

Set this dotfield to a value of 1 to acknowledge any outstanding alarms associated with a specified tag or alarm group. When the specified tag is an alarm group, all unacknowledged alarms associated with the tags within the specified group are acknowledged. When the specified tag is of any type other than alarm group, only the unacknowledged alarm associated with that tag is acknowledged. Setting the .Ack dotfield to a value other than 1 has no meaning.

Data Type

Discrete (read/write)

Valid Values

1

Example

The following statement acknowledges an alarm associated with the Tag1 tag:

```
Tag1.Ack=1;
```

This next example would be used to acknowledge all unacknowledged alarms within the PumpStation alarm group:

```
PumpStation.Ack=1;
```

Note: The .ACK dotfield has an inverse dotfield called .UnAck. When an unacknowledged alarm occurs, .UnAck is set to 1. .UnAck can then be used with animation links or in condition scripts to trigger annunciators for any unacknowledged alarms.

See Also

Alarm, UnAck, AckDev, AckROC, AckDSC, AckValue, AlarmAckModel

.UnAck dotfield

Monitors or controls the alarm acknowledgment status of local alarms.

Category

Alarms

Usage

```
TagName.UnAck=0;
```

Parameter

TagName

Any discrete, integer, real, indirect discrete and analog tag, or alarm group.

Remarks

Set this dotfield to a value of 0 to acknowledge any outstanding alarms associated with the specified tag or alarm group. When the specified tag is an alarm group, all unacknowledged alarms associated with the tags within the specified group are acknowledged. When the specified tag is any other type, only the unacknowledged alarm associated with that tag is acknowledged. Setting this dotfield to a value other than 0 has no meaning.

Data Type

Discrete (read/reset) only

Valid Values

0

Example

The following statement acknowledges any alarm associated with the Tag1 tag.

```
Tag1.UnAck=0;
```

This statement acknowledges all unacknowledged alarms within the alarm group named PumpStation.

```
PumpStation.UnAck=0;
```

.UnAck has an inverse dotfield called .Ack. When an alarm has been acknowledged, the value of the .Ack dotfield is set to 1.

See Also

.Ack, Ack(), .Alarm, .AlarmAckModel

Acknowledge value alarms

You can create scripts that use the .AckValue dotfield to acknowledge all value alarms of a specified local tag or all value alarms in a specified alarm group.

.AckValue dotfield

Monitors and controls the acknowledgment of local value alarms.

Usage

```
TagName.AckValue=1;
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

Set the .**AckValue** dotfield to 1 to acknowledge any outstanding value alarms associated the specified tag/group. When the specified tag is an alarm group, all unacknowledged value alarms associated with the tags within the specified group are acknowledged. When the specified tag is of any type, only the unacknowledged value alarm associated with that tag is acknowledged.

Setting this dotfield to a value other than 1 has no meaning.

Data Type

Discrete (read/write)

Valid Values

1

Examples

The following statement acknowledges a value alarm associated with the Tag1 tag:

```
Tag1.AckValue=1;
```

This example acknowledges all unacknowledged value alarms within the alarm group named PumpStation:

```
PumpStation.AckValue=1;
```

An indirect alarm group (using GroupVar) can be used to acknowledge value alarms. For example, using an assignment such as:

```
StationAlarms.Name = "PumpStation";
```

Where StationAlarms is defined as an alarm group type tag and is then associated with PumpStation. Thus, the statement below is similar to the above examples, except it is used to acknowledge any unacknowledged value alarms within the alarm group PumpStation, which is currently associated with the StationAlarms alarm group tag.

```
StationAlarms.AckValue=1;
```

See Also

.Alarm, .AlarmValue, .Ack, .UnAck, .AckDev, .AckDSC, .AckROC, .AlarmAckModel

Acknowledge discrete alarms

You can create scripts that use the .AckDsc dotfield to acknowledge the discrete alarm of a specified tag or all discrete alarms of a specified alarm group.

.AckDsc dotfield

Acknowledges the discrete alarm of a specified tag or all discrete alarms of a specified alarm group.

Usage

```
TagName.AckDsc=1;
```

Parameter

TagName

Name assigned to the discrete tag or the name of an alarm group.

Remarks

Set to 1 to acknowledge an active discrete alarm associated with the specified tag or alarm group. When the specified tag is an alarm group, all unacknowledged discrete alarms associated with the tags within the specified group are acknowledged. When the specified tag is of any type other than alarm group, only the unacknowledged discrete alarm associated with that tag is acknowledged. Setting this dotfield to a value other than 1 has no meaning.

Data Type

Discrete (read/write)

Valid Values

0 or 1

Examples

The following statement verifies if Tag1 has an active discrete alarm associated with it:

```
IF (Tag1.AlarmDsc == 1) THEN
   MyAlarmMessage="The pumping station currently has an ALARM!";
ENDIF;
```

This dotfield is not linked to the .Ack or .UnAck dotfield. Therefore, even when an active alarm has been acknowledged, this dotfield remains equal to 1.

See Also

.Alarm, .AlarmDSC, .Ack, .UnAck, .AckDev, .AckDSC, .AckROC, .AckValue, .AlarmAckModel

Acknowledge deviation alarms

You can create scripts that use the .AckDev dotfield to acknowledge the minor or major deviation alarm of a specified local tag or the deviation alarms of a specified alarm group.

.AckDev dotfield

Acknowledges the minor or major deviation alarms of a specified local tag or all deviation alarms of a specified alarm group.

Category

Alarm

Usage

```
TagName.AckDev=1;
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

Set this dotfield to a value of 1 to acknowledge any outstanding deviation alarms associated with the specified tag or alarm group. When the specified tag is an alarm group, all unacknowledged deviation alarms associated with the tags within the specified group are acknowledged. Setting this dotfield to a value other than 1 has no meaning.

Data Type

Discrete (read/write)

Valid Values

1

Example

The following statement acknowledges a deviation alarm associated with the Tag1 tag:

```
Tag1.AckDev=1;
```

This next example would be used to acknowledge all unacknowledged deviation alarms within the alarm group named PumpStation:

```
PumpStation.AckDev=1;
```

See Also

.Alarm, .AlarmDev, .Ack, .UnAck, .AckDSC, .AckROC, .AckValue, .AlarmAckModel

Acknowledge rate-of-change alarms

You can create scripts that use the .AckROC dotfield to acknowledge the rate-of-change alarm of a specified local tag or the rate-of-change alarms of a specified alarm group.

.AckROC dotfield

Acknowledges the rate-of-change alarm of a specified local tag or all rate-of-change alarms of a specified alarm group.

Usage

```
TagName.AckROC=1;
```

Parameter

TagName

Name assigned to the integer, real, or indirect analog tag or the name of an alarm group.

Remarks

Set this dotfield to a value of 1 to acknowledge any outstanding rate-of-change alarms associated with a specified tag or alarm group. When the specified tag is an alarm group, all unacknowledged rate-of-change alarms associated with the tags within the specified group are acknowledged. When the specified tag is of any type other than alarm group, only the unacknowledged rate-of-change alarm associated with that tag is acknowledged. Setting this dotfield to a value other than 1 has no meaning.

Data Type

Discrete (read/write)

Valid Values

1

Examples

The following statement acknowledges a rate-of-change alarm associated with a tag named Tag1.

```
Tag1.AckROC=1;
```

This next example acknowledges all unacknowledged rate-of-change alarms within the alarm group named PumpStation:

```
PumpStation.AckROC=1;
```

See Also

.Alarm, .AlarmROC, .Ack, .UnAck, .AckDev, .AckDSC, .AckValue, .AlarmAckModel

Use script functions to acknowledge alarms

You can use the Ack() script function to acknowledge all alarms for a tag or group.

If you are using the Alarm Viewer control, you can acknowledge alarms using methods. For more information, see Acknowledge alarms.

If you are using the Distributed Alarm Display object, use script functions to acknowledge alarms. For more information, see Acknowledge alarms.

Ack() function

Acknowledges any unacknowledged InTouch alarm.

Category

Alarm

Syntax

```
Ack TagName;
```

Arguments

TagName

Any InTouch tag, alarm group, or group variable.

Remarks

This function can be applied to a tag or an alarm group.

Examples

The following statements can be used on a push button to acknowledge any unacknowledged alarm:

```
Ack $System; {All alarms}
Ack Tagname;
Ack GroupName;
```

See Also

almAckAll(), almAckGroup() almAckTag(), almAckDisplay(), almAckRecent(), almAckPriority(). almAckSelect(), almAckSelectedGroup(), almAckSelectedPriority(), almAckSelectedTag()

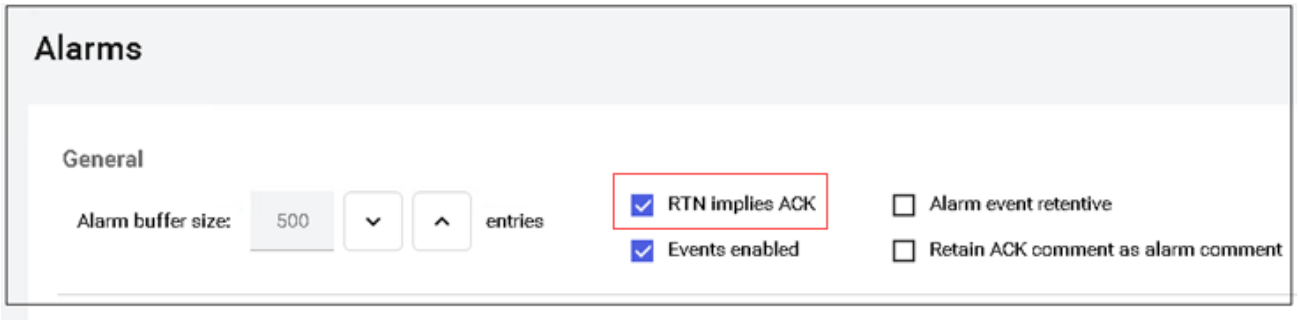
Use automatic acknowledgement when the tag value returns to normal

The InTouch HMI can automatically acknowledge an alarm when an alarmed tag's value returns to its normal state. This option does not apply to expanded summary alarms.

Configure alarm acknowledgement on return

1. Open an application in WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Alarms**.

   The **Alarms** configuration screen appears.
3. In the General area, select **RTN implies ACK** to have the InTouch HMI automatically acknowledge alarms whose values return to the normal state (RTN).

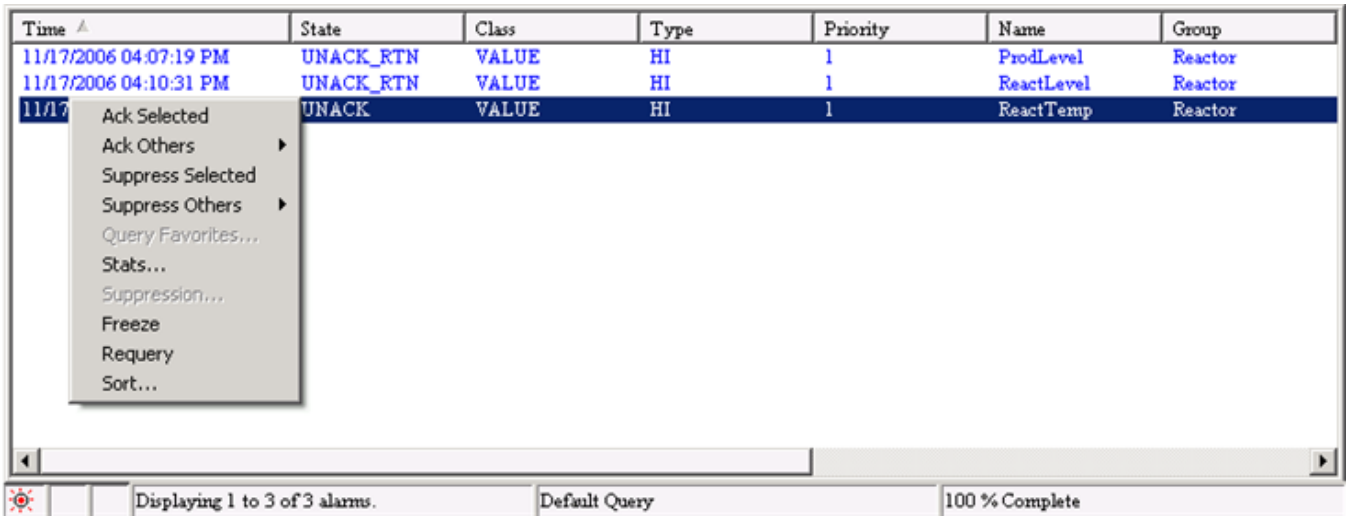


4. Select the **RTN implies ACK** check box
5. Select **OK**.

Use alarm clients to acknowledge alarms

Operators acknowledge alarms that appear in the Alarm Viewer control from WindowViewer.

In the display, the **State** column shows the current acknowledgement status of the alarm record you select. Also, the text of a record is color coded to indicate its acknowledgement status.



The alarms you acknowledge are removed from the display.

Acknowledge all alarms

1. Right-click in the display, point to **Ack Others**, and then select the appropriate command:
   - Select **Ack All** to acknowledge all current alarms.
   - Select **Ack Visible** to acknowledge all alarms visible in the display.

     The **Ack Comment** dialog box appears.
2. Type an optional acknowledgement comment and select **OK**.

Acknowledge selected alarms

1. Select one or more alarms.
2. Right-click and then select **Ack Selected**. The **Ack Comment** dialog box appears.
3. Type an optional acknowledgement comment and select **OK**.

Acknowledge alarms by group, tag, or priority

1. Select the alarm(s).
2. Right-click in the display, point to **Ack Others**, and then select the appropriate command:
   - Select **Ack Selected Groups** to acknowledge all alarms that belong to the selected alarm group(s).
   - Select **Ack Selected Tags** to acknowledge alarms for all tags that have the same name(s) as the selected alarm(s).
   - Select **Ack Selected Priorities** to acknowledge all alarms with the same priority or priorities as the selected alarm(s).

   The **Ack Comment** dialog box appears.
3. Type an optional acknowledgement comment and select **OK**.

Use alarm and acknowledgement comments

There are two kinds of comments: alarm comments and acknowledgement comments.

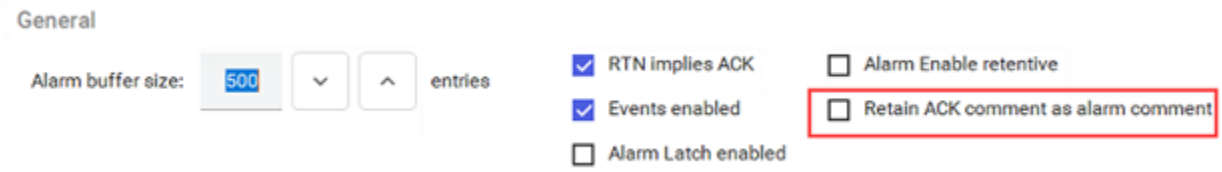
- The alarm comment is set when the new alarm instance occurs. The .AlarmComment dotfield is used for alarm comments and can be set or read in an InTouch script. You specify the default value for this comment in the tag's definition in the Tagname Dictionary. Alarm comments can be up to 131 characters.
- The acknowledgement comment is provided by the operator when he or she acknowledges the alarm.

You can use the acknowledgement comment to update the alarm comment in the tag database.

Allow alarm acknowledgement comments to update the .AlarmComment dotfield

1. Open an application in WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **Alarms**.

   The **Alarms** screen appears.



3. Select the **Retain ACK Comment As Alarm Comment** check box to update the tag's .AlarmComment dotfield and the Tagname Dictionary with the comments entered with alarm acknowledgments.

   If you do not select this check box, the acknowledgment comment is shown with the acknowledged alarm (in the database, printouts, and displays), but .AlarmComment does not change.
4. Select **OK**.

Dismiss alarms at runtime

When an alarm occurs, the run time operator (or system) can acknowledge the alarm to indicate that they are aware of the alarm. An acknowledged alarm that has returned to a normal value will continue to be displayed if the latching feature has been enabled for the Galaxy. With latching enabled, an acknowledged alarm that has returned to normal condition will be placed in the LATCHED alarm state.

LATCHED alarms can be displayed in the current alarms mode to show that the alarms did occur. Alarms go to the LATCHED state if:

- You ACK an alarm from UNACK_RTN state.

  Or
- You return an alarm from ACK state.

To view the LATCHED state, you have to enable the LATCHED state under the global settings. For more information, see the "Enable a Latched State" in the *AVEVA OMI* help.

You can dismiss the LATCHED alarms to remove the LATCHED alarms from the current mode of the Alarm Client Control grid. The dismissed LATCHED alarms would be visible in the recent mode of the Alarm Client Control.

To dismiss selected alarms

1. Select one or more alarms in **LATCHED** state.
2. Right-click anywhere on the grid and select **Dismiss Selected**.

The selected alarm is removed from the grid.

To dismiss other alarms

1. Select one or more alarms in **LATCHED** state.
2. Right-click anywhere on the grid, point to **Dismiss Others**, and select one of the following options:
   - **Dismiss All** to dismiss all alarms in alarm state
   - **Dismiss Visible** to dismiss all visible alarms
   - **Dismiss Selected Groups** Groups to dismiss alarms with the same provider names and group names of one or more selected active alarms
   - **Dismiss Selected Tags** to dismiss alarms with the same provider names, group names, and tag names of one or more selected active alarms

The relevant alarms are removed from the grid.

Use dotfields to dismiss alarms

You can create scripts that use dotfields to dismiss all alarms that are in LATCHED state.

Category

Alarms

Usage

```
TagName.Dismiss=1;
```

Parameter

TagName

Any discrete, integer, real, indirect discrete and analog tag, or alarm group.

Remarks

Set this dotfield to a value of 1 to dismiss any outstanding alarms associated with a specified tag or alarm group. When the specified tag is an alarm group, all LATCHED alarms associated with the tags within the specified group are dismissed. When the specified tag is of any type other than alarm group, only the LATCHED alarm associated with that tag is dismissed. Setting the .Dismiss dotfield to a value other than 1 has no meaning.

Data Type

Discrete (read/write)

Valid Values

1

Example

The following statement dismisses an alarm associated with the Tag1 tag:

```
Tag1.Dismiss=1;
```

This next example would be used to dismiss all LATCHED alarms within the PumpStation alarm group:

```
PumpStation.Dismiss=1;
```

Control alarm properties of tags and groups at runtime

You can use alarm dotfields to dynamically manage alarm conditions. Many of these dotfields are accessible using I/O, expressions, and scripts. Through I/O access, you can monitor and control a specific tag's alarm information using other Windows applications, such as Excel, or WindowViewer running on a remote node.

To access dotfields associated with a tag, use this syntax:

tag.dotfield

For example, if you want to allow run-time changes to the HiHi alarm limit on a tag named Analog_tag, you can create an Analog - User Input touch link to a button and enter Analog_tag.HiHiLimit as the expression in the link's dialog box. During run time, the operator simply selects the button and types in a new value for the HiHi alarm limit assigned to the Analog_tag.

The following table briefly describes each alarm dotfield.

Dotfield	Description
.Ack	Monitors/controls the alarm acknowledgment status of tags and alarm groups. .Ack has an inverse tag dotfield called .UnAck. When an unacknowledged alarm occurs, .UnAck is set to 1. The .UnAck dotfield can be used in animation links or condition scripts to trigger annunciators for any unacknowledged alarms.
.AckDev	Monitors/controls the alarm acknowledgment status of deviation type alarms active on an analog tag or alarm group.
.AckDsc	Monitors/controls the current acknowledgement status of a discrete tag.
.AckROC	Monitors/controls the alarm acknowledgment status of rate-of-change type alarms active on the tag.
.AckValue	Monitors the alarm acknowledgment status of value type alarms active on the tag.

Dotfield	Description
.Alarm	Signals that an alarm condition exists.
.AlarmAckModel	Monitors the acknowledgement model associated with the tag as follows: 0=condition (default) 1=event 2=expanded Applies to discrete or analog tags with alarms. Read only, but can be configured in WindowMaker.
.AlarmDev	Signals that a deviation alarm exists.
.AlarmDevCount	Tracks the total number of active deviation alarms for a given tag or alarm group.
.AlarmDevDeadband	Monitors/controls the deviation percentage deadband for both minor and major deviation alarms.
.AlarmDevUnAckCount	Tracks the number of unacknowledged deviation alarms active on a given tag or alarm group.
.AlarmDisabled	Disables/enables events and alarms. Applies to discrete and analog tags with alarms, or to alarm groups.
.AlarmDsc	Indicates that a discrete alarm condition is currently active.
.AlarmDscCount	Tracks the total number of discrete alarms active on a given tag or alarm group.
.AlarmDscDisabled	Indicates whether or not the tag can generate discrete alarms. **Note:** This dotfield is the same as .AlarmDisabled dotfield for a discrete tag.
.AlarmDscEnabled	Indicates whether or not the tag can generate discrete alarms. **Note:** This dotfield is the same as .AlarmEnabled dotfield for a discrete tag.
.AlarmDscInhibitor	Returns the name of the inhibitor tag assigned to the discrete alarm (if any) for this tag.
.AlarmDscUnAckCount	Tracks the total number of unacknowledged discrete alarms active for a given tag or alarm group.

Dotfield	Description
.AlarmEnabled	Disables/enables events and alarms.
.AlarmHiDisabled	Disables/enables the High limit for analog tags with alarms.
.AlarmHiHiDisabled	Disables/enables the HiHi limit for analog tags with alarms.
.AlarmHiHiEnabled	Disables/enables the HiHi limit for analog tags with alarms.
.AlarmHiHiInhibitor	Returns the inhibitor tag reference for the HiHi limit. Applies to analog tags with alarms. Read only but can be configured in WindowMaker.
.**AlarmHiInhibitor**	Returns the inhibitor tag reference for the High limit. Applies to analog tags with alarms. Read only but can be configured in WindowMaker.
.AlarmLoDisabled	Disables/enables the Low limit for analog tags with alarms.
.AlarmLoEnabled	Disables/enables the Low limit for analog tags with alarms.
.**AlarmLoInhibitor**	Returns the inhibitor tag reference for the Low limit. Applies to analog tags with alarms. Read only but can be configured in WindowMaker.
.AlarmLoLoDisabled	Disables/enables the LoLo limit for analog tags with alarms.
.AlarmLoLoEnabled	Disables/enables the LoLo limit for analog tags with alarms.
.**AlarmLoLoInhibitor**	Returns the inhibitor tag reference for the LoLo limit. Applies to analog tags with alarms. Read only but can be configured in WindowMaker.
.AlarmMajDevDisabled	Disables/enables the Major Deviation limit for analog tags with alarms.
.AlarmMajDevEnabled	Disables/enables the Major Deviation limit for analog tags with alarms.
.**AlarmMajDevInhibitor**	Returns the inhibitor tag reference for the Major Deviation limit. Applies to analog tags with alarms. Read only but can be configured in WindowMaker.

Dotfield	Description
.AlarmMinDevDisabled	Disables/enables the Minor Deviation limit for analog tags with alarms.
.AlarmMinDevEnabled	Disables/enables the Minor Deviation limit for analog tags with alarms.
.**AlarmMinDevInhibitor**	Returns the inhibitor tag reference for the Minor Deviation limit. Applies to analog tags with alarms. Read only but can be configured in WindowMaker.
.AlarmROC	Signals that a rate-of-change type alarm exists.
.AlarmROCCount	Tracks the total number of rate of change alarms active on a given tag or alarm group.
.AlarmROCDisabled	Disables/enables the rate of change limit for analog tags with alarms.
.AlarmROCEnabled	Disables/enables the rate of change limit for analog tags with alarms.
.**AlarmROCInhibitor**	Returns the inhibitor tag reference for the rate of change limit. Applies to analog tags with alarms. Read only but can be configured in WindowMaker.
.AlarmROCUnAckCount	Tracks the total number of unacknowledged rate of change alarms on a given tag or alarm group.
.AlarmTotalCount	Tracks the total number of alarms active for a given tag or alarm group.
.AlarmUnAckCount	Tracks the total number of unacknowledged alarms active for a tag or alarm group.
.AlarmUserDefNum1Set	Read/write real (floating point), default 0 and value not set. Applies to discrete tags with alarms, to analog tags with alarms, or to alarm groups. **Note:** The value of this dotfield is attached to the alarm, but ONLY if a value has been set, for example, by a script or a POKE.
.AlarmUserDefNum2	Read/write real (floating point), default 0 and value not set. Applies to discrete tags with alarms, to analog tags with alarms, or to alarm groups. **Note:** The value of this dotfield is attached to the alarm, but ONLY if a value has been set, for example, by a script or a POKE.

Dotfield	Description
.AlarmUserDefNum2Set	Read/write discrete. TRUE if a script has defined the .AlarmUserDefNum2 for the corresponding tag. To disassociate the value of .AlarmUserDefNum2 for the tag, set this dotfield to FALSE. The default is FALSE.
.AlarmUserDefStr	Read/write text string, default "" and value not set. Applies to discrete tags with alarms, to analog tags with alarms, or to alarm groups. **Note:** The value of this dotfield is attached to the alarm, but ONLY if a value has been set, for example, by a script or a POKE.
.AlarmUserDefStrSet	Read/write discrete. TRUE if a script has defined the .AlarmUserDefStr for the corresponding tag. To disassociate the value of .AlarmUserDefStr for the tag, set this dotfield to FALSE. The default is FALSE.
.AlarmValDeadband	Monitors/controls the value of an alarm's value deadband.
.AlarmValueCount	Tracks the total number of value alarms active on a given tagname or alarm group.
.AlarmValueUnAckCount	Tracks the total number of unacknowledged value alarms active on a given tagname or alarm group.
.DevTarget	Monitors/controls the target for minor and major deviation alarms.
.HiLimit, .HiHiLimit, .LoLimit, .LoLoLimit	Read/write analog tagname dotfields that monitors /controls the limits for value alarm checks. These dotfields are only valid for integer and real tags.
.HiStatus, .HiHiStatus, .LoStatus, .LoLoStatus	Read only discrete dotfields that determines whether an alarm of a specified type exists. These fields are only valid for integer and real tags.
.MajorDevPct	Read/write integer dotfield that monitors or controls the major percentage of deviation for alarm checking.
.MajorDevStatus	Read-only discrete dotfield that determines whether a major deviation alarm exists for the specified tagname.
.MinorDevPct	Read/write integer dotfield monitors and/or controls the minor percentage of deviation for alarm checking.
.MinorDevStatus	Read only discrete dotfield used to determine whether

Dotfield	Description
	a minor deviation alarm exists for the specified tagname.
.Name	Read/write message dotfield used to display the actual name of the tagname. For example, it can be used to determine the name of an Alarm Group that a Group Variable is pointing to, or the name of a TagID tags. It can also be written to change the Alarm Group that a Group Variable is pointing to.
.Normal	Read only discrete dotfield that is equal to 1 when there are no alarms for the specified tagname. This dotfield is valid for Alarm Groups and Group Variables as well as ordinary tags.
.ROCPct	Read/write dotfield used to monitor and/or control the rate of change for alarm checking.
.ROCStatus	Read only discrete dotfield used to determine whether a rate-of-change alarm exists for the specified tag.

Determine if tags or alarm groups are in an alarm condition

Use the following dotfields and a system tag to determine the status of alarms in a running application. You can find out if a new alarm occurred or whether a tag or alarm group is in alarm or is the normal state. You can also find out the status of discrete, deviation, and rate-of-change alarms.

- $NewAlarm System Tag
- $System system tag
- .Alarm dotfield
- .Normal Dotfield
- .AlarmDsc dotfield
- .AlarmDev dotfield
- .AlarmROC dotfield
- .LoStatus dotfield
- .LoLoStatus Dotfield
- .HiStatus dotfield
- .HiHiStatus dotfield
- .MinorDevStatus dotfield
- .MajorDevStatus dotfield
- .ROCStatus dotfield

$NewAlarm System Tag

Set to 1 if a new alarm occurs in a running application. The **$NewAlarm** system tag is not set for remote alarms.

Syntax

```
$NewAlarm=value;
```

Data Type

Discrete (read/write)

Valid Values

0 or 1

Remarks

Associate the **$NewAlarm** system tag with an animation object in an application window. For example, associate the tag with an acknowledgment button that an operator selects to reset the value of the tag to 0 and acknowledge the alarm. You can also link the **$NewAlarm** system tag to the **PlaySound** logic function to sound an audible warning when an alarm occurs.

Example

Add a button to an alarm acknowledgement window and attach the following action script that runs when the operator selects the button.

```
Ack $System;
$NewAlarm=0;
HideSelf;
```

When the operator selects on the button, all alarms are acknowledged, the **$NewAlarm** system tag is reset to 0, and the alarm acknowledgement window is hidden from view.

$System system tag

The default alarm group.

Category

alarms

Usage

```
$System
```

Remarks

By default, tags are assigned to this root alarm group. All defined alarm groups are descendants of $System.

Data Type

System alarm group

Example(s)

```
$System.Ack = 1; {Acknowledges All Alarms}
```

.Alarm dotfield

Returns 0 when a specified tag or alarm group is not currently in an alarm state. When an alarm occurs, the .**Alarm dotfield** returns 1. It remains at 1 until the alarm condition no longer exists. The .**Alarm** dotfield has an inverse dotfield called .**Normal**.

If the specified tag is the name of an alarm group, the **.Alarm** dotfield returns 1 if any of the tags that belong to the group are in an alarm state.

Category

Alarms

Usage

```
TagName.Alarm
```

Parameter

TagName

Any discrete, integer, real tag, indirect discrete and analog tag, or alarm group tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following statement verifies if **Tag1** has an active alarm associated with it:

```
IF (Tag1.Alarm == 1) THEN
```

The body of this IF-THEN statement is processed if active alarms exist within the **PumpStation** alarm group.

```
IF (PumpStation.Alarm == 1) THEN
   MyAlarmMessage="The pumping station currently has an ALARM!";
ENDIF;
```

This dotfield is not linked to the .**Ack** or .**UnAck** dotfields. Therefore, even when an active alarm has been acknowledged, .**Alarm** remains equal to 1.

.Normal dotfield

Returns 1 when a specified tag is not in an alarm condition. When an alarm occurs, the .**Normal dotfield** returns 0. The .**Normal** dotfield has an inverse dotfield called .**Alarm**.

Category

Alarms

Syntax

```
TagName.Normal
```

Parameter

TagName

Any discrete, integer, real tag, indirect discrete and analog tag, or alarm group tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following IF-THEN statement runs when there are no alarms associated with the **Tag1** tag. When there is one or more alarms active for **Tag1**, the "ELSE" body runs:

```
IF (Tag1.Normal==1) THEN
   MyOperatorMessage="Tag1 is OK - No alarms associated with it";
ELSE
   MyOperatorMessage="Tag1 has one or more alarms active!";
ENDIF;
```

.AlarmDsc dotfield

Indicates whether an alarm condition exists for a specified discrete tag or alarm group. The default value is 0. When a discrete alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the alarm condition no longer exists.

If the specified tag is the name of an alarm group, the .**AlarmDsc** dotfield is set to 1 if any of the tags within the group are in an active discrete alarm.

Category

Alarms

Usage

```
TagName.AlarmDsc
```

Parameter

TagName

Any discrete tag, indirect discrete tag, or alarm group.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following statement verifies if Tag1 has an active discrete alarm associated with it:

```
IF (Tag1.AlarmDsc == 1) THEN
   MyAlarmMessage="The pumping station currently has an ALARM!";
ENDIF;
```

This dotfield is not linked to the **.Ack** or **.UnAck** dotfields. Therefore, even when an active alarm has been acknowledged, the **.AlarmDsc** dotfield remains equal to 1.

See Also

.Ack, .UnAck, .Alarm, .AlarmDsc, .AckDsc

.AlarmDev dotfield

Indicates when a deviation alarm becomes active for the specified tag or alarm group. The default value is 0.

When a deviation alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the alarm condition no longer exists.

If the specified tag is the name of an alarm group, the **.AlarmDev** dotfield is set to 1 if any of the tags within the group are in an active alarm state.

Category

Alarms

Usage

```
TagName.AlarmDev
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following statement verifies if Tag1 has an active deviation alarm associated with it:

```
IF (Tag1.AlarmDev == 1) THEN
```

The body of this IF-THEN statement is processed if there are active deviation alarms within the PumpStation alarm group.

```
IF (PumpStation.AlarmDev == 1) THEN
   MyAlarmMessage="The pumping station currently has an ALARM!";
ENDIF;
```

This dotfield is not linked to the .Ack or .UnAck dotfields. Therefore, even when an active alarm has been acknowledged, this dotfield remains equal to 1.

See Also

.Ack, .UnAck, .Alarm, .AckDev

.AlarmROC dotfield

Indicates when a rate-of-change alarm condition becomes active for the specified tag or alarm group. The default

value is 0. When a rate-of-change alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the rate-of-change alarm condition no longer exists.

If the specified tag is the name of an alarm group, the **.AlarmROC** dotfield is set to 1 if any of the tags within the group are in a rate-of-change alarm state.

Category

Alarms

Usage

```
TagName.AlarmROC
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following statement verifies if Tag1 has an active rate-of-change alarm associated with it:

```
IF (Tag1.AlarmROC == 1) THEN
```

The body of this IF-THEN statement would be processed if there were active rate-of-change alarms within the alarm group named PumpStation.

```
IF (PumpStation.AlarmROC == 1) THEN
   MyAlarmMessage="The pumping station currently has an ALARM!";
ENDIF;
```

This dotfield is not linked to the .Ack or .UnAck dotfield. Therefore, even when an active rate-of-change alarm has been acknowledged, this dotfield remains equal to 1.

See Also

.Ack, .AckROC, .Alarm, .AlarmROCEnabled, .AlarmROCDisabled

.LoStatus dotfield

Indicates when a Low alarm condition becomes active for the specified tag or alarm group. The default value is 0.

When a Low alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the Low alarm condition no longer exists.

This dotfield is often used in conjunction with the **.Alarm** and **.Ack** dotfields to determine the specific alarm state of a particular tag.

Category

Alarms

Usage

```
TagName.LoStatus
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following IF-THEN runs when the .LoStatus (low alarm condition) for the MyTag tag is equal to 1.

```
IF (MyTag.LoStatus == 1) THEN
   OperatorMessage="MyTag has gone into Low Alarm";
ENDIF;
```

See Also

.Alarm, .AlarmValue, .Ack, .LoLimit, .LoSet, .AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled, .AlarmLoEnabled, .AlarmLoInhibitor

.LoLoStatus dotfield

Indicates when a LoLo alarm condition becomes active for the specified tag or alarm group. The default value is 0. When a LoLo alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the LoLo alarm condition no longer exists.

This dotfield is often used in conjunction with the **.Alarm** and **.Ack** dotfields to determine the exact nature of the alarm status of a particular tag within the system.

Category

Alarms

Usage

```
TagName.LoLoStatus
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following IF-THEN statement runs when the .LoLoStatus (LoLo alarm limit) for the MyTag tag is equal to 1.

```
IF (MyTag.LoLoStatus == 1) THEN
   OperatorMessage="MyTag has gone into LoLo Alarm";
ENDIF;
```

See Also

.Alarm, .AlarmValue, .Ack, .LoLoLimit, .LoLoSet, .AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled, .AlarmLoLoEnabled, .AlarmLoLoInhibitor

.HiStatus dotfield

Indicates when a High alarm condition becomes active for the specified tag or alarm group. The default value is 0. When a High alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the High alarm condition no longer exists.

This dotfield is often used in conjunction with the **.Alarm** and **.Ack** dotfields to determine the specific alarm state of a tag.

Category

Alarms

Usage

```
TagName.HiStatus
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

This script calls another script to stop a pump motor output if the MotorAmps tag goes into high limit alarm status.

```
IF (MotorAmps.HiStatus == 1) THEN
   CALL PumpShutdown( );
ENDIF;
```

See Also

.Alarm, .AlarmValue, .Ack, .HiLimit, .HiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiDisabled, .AlarmHiEnabled, .AlarmHiInhibitor

.HiHiStatus dotfield

Indicates when a HiHi alarm condition becomes active for the specified tag or alarm group. The default value is 0. When a HiHi alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the HiHi alarm condition no longer exists.

This dotfield is often used in conjunction with the **.Alarm** and **.Ack** dotfields to determine the specific alarm state of a tag.

Category

Alarms

Usage

```
TagName.HiHiStatus
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following IF-THEN statement runs when the .HiHiStatus (HiHi alarm) for the MyTag tag is 1.

```
IF (MyTag.HiHiStatus == 1) THEN
   OperatorMessage="MyTag has gone into HiHi Alarm";
ENDIF;
```

See Also

.Alarm, .AlarmValue, .Ack, .HiHiLimit, .HiHiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor

.MinorDevStatus dotfield

Indicates when a minor deviation alarm becomes active for the specified tag or alarm group. The default value is 0. When a minor deviation alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the minor deviation alarm condition no longer exists.

This dotfield is often used in conjunction with the **.Alarm** and **.Ack** dotfields to determine the specific alarm state of a tag.

Category

Alarms

Usage

```
TagName.MinorDevStatus
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following IF-THEN statement runs when the .MinorDevStatus (minor deviation alarm) for the tag MyTag is equal to 1.

```
IF (MyTag.MinorDevStatus == 1) THEN
   OperatorMessage="MyTag has gone into a Minor Deviation Alarm";
ENDIF;
```

See Also

.AckDev, .AlarmDev, .AlarmMinDevDisabled, .AlarmMinDevEnabled, .AlarmMinDevInhibitor, .MinorDevPct, .MajorDevStatus

.MajorDevStatus dotfield

Indicates when a major deviation alarm becomes active for the specified tag or alarm group. The default value is 0. When a major deviation alarm condition exists, the specified dotfield is set to 1. The value remains 1 until the major deviation alarm condition no longer exists.

This dotfield is often used in conjunction with the **.Alarm** and **.Ack** dotfields to determine the specific alarm state of a tag.

Category

Alarms

Usage

```
TagName.MajorDevStatus
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following IF-THEN statement runs when the .MajorDevStatus (Major deviation alarm) for the MyTag tag is equal to 1.

```
IF (MyTag.MajorDevStatus == 1) THEN
   OperatorMessage="MyTag has gone into a Major Deviation Alarm";
ENDIF;
```

See Also

.AckDev, .AlarmDev, .AlarmMajDevDisabled, .AlarmMajDevEnabled, .AlarmMajDevInhibitor, .MajorDevPct, .MajorDevSet, .MinorDevStatus

.ROCStatus dotfield

Indicates when a rate-of-change alarm becomes active for the specified tag or alarm group. The default value is 0. When a rate-of-change alarm condition exists for the specified tag, it is set to a value of 1. The value remains 1 until the rate-of-change alarm condition no longer exists.

This dotfield is often used in conjunction with the **.Alarm** and **.Ack** dotfields to determine the specific alarm state of a tag.

Category

Alarms

Usage

```
TagName.ROCStatus
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The following IF-THEN statement runs when the .ROCStatus (rate-of-change alarm) for the MyTag tag is equal to 1.

```
IF (MyTag.ROCStatus == 1) THEN
   OperatorMessage="MyTag has gone into a Rate-Of-Change alarm";
ENDIF;
```

See Also

.ROCPct, .ROCSet

Revert alarm status handling to InTouch 7.1 behavior

For InTouch 7.11 and later, the default behavior when a HiHi, LoLo, MinDev, or MajDev alarm occurs is to reset the In Alarm, ACK, and $NewAlarm status of Hi, Lo, MajDev, or MinDev alarms, respectively. For InTouch 7.1 and older versions, these alarm statuses were not affected by the occurence of a HiHi, LoLo, MinDev, or MajDev alarm.

To support legacy applications that expect Hi, Lo, MajDev, or MinDev alarms to maintain their In Alarm, ACK, and $NewAlarm status despite the occurrence of the other alarms, the InTouch.ini file can include an IT71StatusFlag parameter.

To force the alarm handling to operate as it did in InTouch 7.1 and older versions

- Enter the following line in the InTouch.ini file:

  ```
  IT71StatusFlags=1
  ```

To change the alarm handling back to the default operation for InTouch 7.11 and later

- Enter the following line in the InTouch.ini file:

  ```
  IT71StatusFlags=0
  ```

For more information about how to edit the InTouch.ini file to configure this and other InTouch operating parameters, see *AVEVA™ InTouch HMI Application Development Guide*.

Determine if alarm limits are set for tags

The following dotfields indicate whether alarm limits are set for a tag while an application is running.

- .LoLoSet dotfield
- .LoSet dotfield
- .HiSet dotfield
- .HiHiSet Dotfield
- .MinorDevSet dotfield
- .MajorDevSet dotfield
- .ROCSet dotfield

.LoLoSet dotfield

Indicates whether a LoLo alarm limit has been set for an integer or real tag.

Category

Alarms

Usage

```
TagName.LoLoSet
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The THEN block executes if the LoLo alarm limit is set for the MyTag tag:

```
IF (MyTag.LoLoSet== 1) THEN
   MsgTag="LoLo alarm limit has been set for MyTag";
ENDIF;
```

See Also

.Alarm, .AlarmValue, .Ack, .LoLoStatus, .LoLoLimit, .AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled, .AlarmLoLoEnabled, .AlarmLoLoInhibitor

.LoSet dotfield

Indicates whether a Low alarm limit has been set for an integer or real tag.

Category

Alarms

Usage

```
TagName.LoSet
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The THEN block executes if the Low alarm limit is set for the MyTag tag:

```
IF (MyTag.LoSet== 1) THEN
   MsgTag="Low alarm limit has been set for MyTag";
ENDIF;
```

See Also

.Alarm, .AlarmValue, .Ack, .LoStatus, .LoLimit, .AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled, .AlarmLoEnabled, .AlarmLoInhibitor

.HiSet dotfield

Indicates whether a High alarm limit has been set for an integer or real tag.

Category

Alarms

Usage

```
TagName.HiSet
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The THEN block executes if the High alarm limit is set for the MyTag tag:

```
IF (MyTag.HiSet== 1) THEN
   MsgTag="High alarm limit has been set for MyTag";
ENDIF;
```

See Also

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiLimit, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor

.HiHiSet Dotfield

Indicates whether a HiHi alarm limit has been set for an integer or real tag.

Category

Alarms

Usage

```
TagName.HiHiSet
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The THEN block executes if the HiHi alarm limit is set for the MyTag tag:

```
IF (MyTag.HiHiSet== 1) THEN
   MsgTag="HiHi alarm limit has been set for MyTag";
ENDIF;
```

See Also

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiLimit, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor

.MinorDevSet dotfield

Indicates whether a minor deviation alarm limit has been set for an integer or real tag.

Category

Alarms

Usage

```
TagName.MinorDevSet
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The THEN block executes if the minor deviation percentage alarm limit is set for the MyTag tag:

```
IF (MyTag.MinorDevSet== 1) THEN
   MsgTag="Minor deviation alarm limit has been set for MyTag";
ENDIF;
```

See Also

.AckDev, .AlarmDev, .AlarmMinDevDisabled, .AlarmMinDevEnabled, .AlarmMinDevInhibitor, .MinorDevPct, .MinorDevStatus

.MajorDevSet dotfield

Indicates whether a major deviation alarm limit has been set for an integer or real tag.

Category

Alarms

Usage

```
TagName.MajorDevSet
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The THEN block executes if the major deviation percentage alarm limit is set for the MyTag tag:

```
IF (MyTag.MajorDevSet== 1) THEN
   MsgTag="Major deviation alarm limit has been set for MyTag";
ENDIF;
```

See Also

.AckDev, .AlarmDev, .AlarmMajDevDisabled, .AlarmMajDevEnabled, .AlarmMajDevInhibitor, .MajorDevPct, .MajorDevStatus

.ROCSet dotfield

Indicates whether a rate-of-change alarm limit has been set for an integer or real tag.

Category

Alarms

Usage

```
TagName.ROCSet
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Data Type

Discrete (read-only)

Valid Values

0 or 1

Example

The THEN block executes if the rate-of-change alarm limit is set for the MyTag tag:

```
IF (MyTag.ROCSet == 1) THEN
   MsgTag="Rate-of-change alarm limit has been set for MyTag";
ENDIF;
```

See Also

.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiHiLimit, .HiLimit, .HiSet, .LoSet, .LoLoSet, .HiStatus, .HiHiStatus, .ROCPct, .ROCStatus

Enabling and Disabling Alarms for a Tag or Alarm Group

The InTouch HMI provides a set of dotfields that can enable or disable alarms that are set for a tag while an application is running.

Enable/disable all alarms

The **.AlarmEnabled** and **.AlarmDisabled** dotfields enable or disable alarms for a tag or alarm group based upon their respective values. The alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmEnabled**, a value of 1 enables alarms for a tag or alarm group. An **.AlarmDisabled** value of 1 disables alarms for a tag or alarm group.

When either dotfield enables alarms for an alarm group, all tags that belong to the group have alarming enabled.

When either dotfield disables alarms, all events and alarms are ignored. The alarms are not stored in alarm memory, nor are they written to disk.

.AlarmEnabled dotfield

Enables or disables alarms for a tag or an alarm group.

Category

Alarms

Usage

```
TagName.AlarmEnabled
```

Parameter

TagName

Any discrete, integer, real, indirect discrete, indirect analog tag, or alarm group.

Remarks

When **.AlarmEnabled** is set to 0, all events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

When the specified tag is an alarm group, all alarms associated with the tags within the specified alarm group are enabled.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1 = Enable alarms (default)

Example

The following example disables the alarms of the Tag1 tag:

```
Tag1.AlarmEnabled=0;
```

See Also

.AlarmDisabled

.AlarmDisabled dotfield

Enables or disables alarms for a tag or an alarm group.

Category

Alarms

Usage

```
TagName.AlarmDisabled
```

Parameter

TagName

Any discrete, integer, real, indirect discrete, indirect analog tag, or alarm group.

Remarks

When .AlarmDisabled is set to 1, all events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

When the specified tag is an alarm group, all alarms associated with the tags within the specified alarm group are disabled.

This is the opposite of the .AlarmEnabled dotfield.

Example

The following example enables the alarms of the Tag1 tag.

```
Tag1.AlarmDisabled=0;
```

See Also

.AlarmEnabled

Enable/disable LoLo alarms

The **.AlarmLoLoEnabled** and **.AlarmLoLoDisabled** dotfields enable or disable LoLo alarms for a tag or alarm group based upon their respective values. The LoLo alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmLoLoEnabled**, a value of 1 enables LoLo alarms for a tag or alarm group. An **.AlarmLoLoDisabled** value of 1 disables LoLo alarms for a tag or alarm group.

When either dotfield enables LoLo alarms for an alarm group, all tags that belong to the group have LoLo alarms enabled. When either dotfield disables LoLo alarms, all LoLo alarms are ignored. The alarms are not stored in alarm memory, nor are they written to disk.

.AlarmLoLoEnabled dotfield

Enables or disables LoLo condition events and alarms.

Category

Alarms

Usage

```
TagName.AlarmLoLoEnabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmLoLoEnabled is set to 0, all LoLo condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1 = Enable alarms (default)

Example

The following example disables the LoLo alarms of the Tag1 tag:

```
Tag1.AlarmLoLoEnabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled

.AlarmLoLoDisabled dotfield

Enables or disables LoLo condition events and alarms.

Category

Alarms

Usage

```
TagName.AlarmLoLoDisabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmLoLoDisabled is set to 1, all LoLo condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

Data Type

Discrete (read/write)

Valid Values

1 = Disable alarms

0 = Enable alarms (default)

Example

The following example enables LoLo alarms of the Tag2 tag:

```
Tag2.AlarmLoLoDisabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmLoLoEnabled

Enable/disable Low alarms

The **.AlarmLoEnabled** and **.AlarmLoDisabled** dotfields enable or disable Low alarms for a tag or alarm group based upon their respective values. The Low alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmLoEnabled**, a value of 1 enables Low alarms for a tag or alarm group. An **.AlarmLoDisabled** value of 1 disables Low alarms for a tag or alarm group.

When either dotfield enables Low alarms for an alarm group, all tags that belong to the group have Low alarms

enabled. When either dotfield disables Low alarms, all Low alarms are ignored. The alarms are not stored in alarm memory, nor are they written to disk.

.AlarmLoEnabled dotfield

Enables or disables Low condition events and alarms.

Category

Alarms

Usage

```
TagName.AlarmLoEnabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmLoEnabled is set to 0, all Low condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1 = Enable alarms (default)

Example

The following example disables the Low alarms of the Tag1 tag:

```
Tag1.AlarmLoEnabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled

.AlarmLoDisabled dotfield

Enables or disables Low condition events and alarms.

Category

Alarms

Usage

```
TagName.AlarmLoDisabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmLoDisabled is set to 1, all Low condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

Data Type

Discrete (read/write)

Valid Values

1 = Disable alarms

0 = Enable alarms (default)

Example

The following example enables Low alarms of the Tag2 tag:

```
Tag2.AlarmLoDisabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmLoEnabled

Enable/disable High alarms

The **.AlarmHiEnabled** and **.AlarmHiDisabled** dotfields enable or disable High alarms for a tag or alarm group

based upon their respective values. The High alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmHiEnabled**, a value of 1 enables High alarms for a tag or alarm group. An **.AlarmHiDisabled** value of 1 disables High alarms for a tag or alarm group.

When either dotfield enables High alarms for an alarm group, all tags that belong to the group have High alarms enabled. When either dotfield disables High alarms, all High alarms are ignored. The High alarms are not stored in alarm memory, nor are they written to disk.

.AlarmHiEnabled dotfield

Enables or disables High condition events and alarms.

Category

Alarms

Usage

```
TagName.AlarmHiEnabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmHiEnabled is set to 0, all High condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmHiDisabled dotfield.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1 = Enable alarms (default)

Example

The following example disables the High alarms of the tag Tag1:

```
Tag1.AlarmHiEnabled=0;
```

See Also

.AlarmHiDisabled, .AlarmEnabled

.AlarmHiDisabled dotfield

Enables or disables High condition events and alarms.

Category

Alarms

Usage

```
TagName.AlarmHiDisabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmHiDisabled is set to 1, all High condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmHiEnabled dotfield.

Data Type

Discrete (read/write)

Valid Values

1 = Disable alarms

0 = Enable alarms (default)

Example

The following example enables High alarms of the Tag2 tag:

```
Tag2.AlarmHiDisabled=0;
```

See Also

.AlarmHiEnabled, .AlarmDisabled

Enable/disable HiHi alarms

The **.AlarmHiHiEnabled** and **.AlarmHiHiDisabled** dotfields enable or disable HiHi alarms for a tag or alarm group based upon their respective values. The HiHi alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmHiHiEnabled**, a value of 1 enables HiHi alarms for a tag or alarm group. An **.AlarmHiHiDisabled** value of 1 disables HiHi alarms for a tag or alarm group.

When either dotfield enables HiHi alarms for an alarm group, all tags that belong to the group have HiHi alarms enabled. When either dotfield disables HiHi alarms, all HiHi alarms are ignored. The HiHi alarms are not stored in alarm memory, nor are they written to disk.

.AlarmHiHiEnabled Dotfield

Enables and/or disables HiHi condition events and alarms.

Category

Alarms

Usage

```
TagName.AlarmHiHiEnabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmHiHiEnabled is set to 0, all HiHi condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1= Enable alarms (default)

Example

The following example disables the HiHi alarms of the Tag1 tag:

```
Tag1.AlarmHiHiEnabled=0;
```

See Also

.AlarmHiHiDisabled, .AlarmEnabled

.AlarmHiHiDisabled dotfield

Enables or disables HiHi condition events and alarms.

Category

Alarms

Usage

```
TagName.AlarmHiHiDisabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmHiHiDisabled is set to 1, all HiHi condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmHiHiEnabled dotfield.

Data Type

Discrete (read/write)

Valid Values

1 = Disable alarms

0 = Enable alarms (default)

Example

The following example enables HiHi alarms of the Tag2 tag:

```
Tag2.AlarmHiHiDisabled=0;
```

See Also

.AlarmHiHiEnabled, .AlarmDisabled

Enable/disable discrete alarms

The **.AlarmDscEnabled** and **.AlarmDscDisabled** dotfields enable or disable discrete alarms for a tag or alarm group based upon their respective values. The discrete alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmDscEnabled**, a value of 1 enables discrete alarms for a tag or alarm group. An **.AlarmDscDisabled** value of 1 disables discrete alarms for a tag or alarm group.

When either dotfield enables discrete alarms for an alarm group, all tags that belong to the group have discrete alarms enabled. When either dotfield disables discrete alarms, all discrete alarms are ignored. The discrete alarms are not stored in alarm memory, nor are they written to disk.

.AlarmDscEnabled dotfield

Indicates indicates whether or not the tag can generate discrete alarms.

Category

Alarms

Usage

```
TagName.AlarmDscEnabled
```

Parameter

TagName

Any discrete or indirect discrete tag or alarm group.

Remarks

When .AlarmDscEnabled is set to 0, all discrete condition alarms and events are ignored. They are not stored in alarm memory, nor are they written to a disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmDscDisabled dotfield.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1= Enable alarms (default)

Example

The following example disables the discrete alarms of the Tag1 tag:

```
Tag1.AlarmDscEnabled=0;
```

See Also

.AlarmDscDisabled

.AlarmDscDisabled dotfield

Indicates whether or not the tag can generate discrete alarms.

Category

Alarms

Usage

```
TagName.AlarmDscDisabled
```

Parameter

TagName

Any discrete or indirect discrete tag or alarm group.

Remarks

When .AlarmDscDisabled is set to 1, all discrete condition alarms and events are ignored. They are not stored in alarm memory, nor are they written to a disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmDscEnabled dotfield.

Data Type

Discrete (read/write)

Valid Values

1 = Disable alarms

0 = Enable alarms (default)

Example

The following example enables discrete alarms of the Tag2 tag:

```
Tag2.AlarmDscDisabled=0;
```

Enabling/Disabling Minor Deviation Alarms

The **.AlarmMinDevEnabled** and **.AlarmMinDevDisabled** dotfields enable or disable minor deviation alarms for a tag or alarm group based upon their respective values. The minor deviation alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmMinDevEnabled**, a value of 1 enables minor deviation alarms for a tag or alarm group. An **.AlarmMinDevDisabled** value of 1 disables minor deviation alarms for a tag or alarm group.

When either dotfield enables minor deviation alarms for an alarm group, all tags that belong to the group have minor deviation alarms enabled. When either dotfield disables minor deviation alarms, all minor deviation alarms are ignored. The minor deviation alarms are not stored in alarm memory, nor are they written to disk.

.AlarmMinDevEnabled dotfield

Enables or disables minor deviation events and alarms.

Category

Alarms

Usage

```
TagName.AlarmMinDevEnabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When **.AlarmMinDevEnabled** is set to 0, all minor deviation events and alarms are ignored. They are not stored

in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1= Enable alarms (default)

Example

The following example disables the minor deviation alarms of the Tag1 tag:

```
Tag1.AlarmMinDevEnabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmMinDevDisabled

.AlarmMinDevDisabled dotfield

Enables or disables minor deviation events and alarms.

Category

Alarms

Usage

```
TagName.AlarmMinDevDisabled
```

Parameter

TagName

Any integer, real, or indirect analog tag, or alarm group.

Remarks

When **.AlarmMinDevDisabled** is set to 1, all minor deviation events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the **.AlarmMinDevEnabled** dotfield.

Data Type

Discrete (read/write)

Valid Values

1 = Disable alarms

0 = Enable alarms (default)

Example

The following example enables minor deviation alarms of the Tag2 tag:

```
Tag2.AlarmMinDevDisabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmMinDevEnabled

Enable/disable major deviation alarms

The **.AlarmMajDevEnabled** and **.AlarmMajDevDisabled** dotfields enable or disable major deviation alarms for a tag or alarm group based upon their respective values. The major deviation alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmMajDevEnabled**, a value of 1 enables major deviation alarms for a tag or alarm group. An **.AlarmMajDevDisabled** value of 1 disables major deviation alarms for a tag or alarm group.

When either dotfield enables major deviation alarms for an alarm group, all tags that belong to the group have major deviation alarms enabled. When either dotfield disables major deviation alarms, all major deviation alarms are ignored. The major deviation alarms are not stored in alarm memory, nor are they written to disk.

.AlarmMajDevEnabled Dotfield

Enables or disables major deviation events and alarms.

Category

Alarms

Usage

```
TagName.AlarmMajDevEnabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmMajDevEnabled is set to 0, all major deviation events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmMajDevDisabled dotfield.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1= Enable alarms (default)

Example

The following example disables major deviation alarms of the Tag1 tag:

```
Tag1.AlarmMajDevEnabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmMajDevDisabled

.AlarmMajDevDisabled dotfield

Enables or disables major deviation events and alarms.

Category

Alarms

Usage

```
TagName.AlarmMajDevDisabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmMajDevDisabled is set to 1, all major deviation events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmMajDevEnabled dotfield.

Data Type

Discrete (read/write)

Valid Values

1 = Disable alarms

0 = Enable alarms (default)

Example

The following example enables major deviation alarms of the Tag2 tag:

```
Tag2.AlarmMajDevDisabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmMajDevEnabled

Enable/Disable rate-of-change alarms

The **.AlarmROCEnabled** and **.AlarmROCDisabled** dotfields enable or disable rate-of-change alarms for a tag or alarm group based upon their respective values. The rate-of-change alarm states associated with both dotfields are the inverse of each other. In the case of **.AlarmROCEnabled**, a value of 1 enables rate-of-change alarms for a tag or alarm group. An **.AlarmROCDisabled** value of 1 disables rate-of-change alarms for a tag or alarm group.

When either dotfield enables rate-of-change alarms for an alarm group, all tags that belong to the group have rate-of-change alarms enabled. When either dotfield disables rate-of-change alarms, all rate-of-change alarms are ignored. The rate-of-change alarms are not stored in alarm memory, nor are they written to disk.

.AlarmROCEnabled dotfield

Enables or disables rate-of-change events and alarms.

Category

Alarms

Usage

```
TagName.AlarmROCEnabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmROCEnabled is set to 0, all rate-of-change condition events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmROCDisabled dotfield.

Data Type

Discrete (read/write)

Valid Values

0 = Disable alarms

1 = Enable alarms (default)

Example

The following example disables the rate-of-change alarms of the Tag1 tag:

```
Tag1.AlarmROCEnabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmROCDisabled

.AlarmROCDisabled dotfield

Disables or enables rate-of-change events and alarms.

Category

Alarms

Usage

```
TagName.AlarmROCDisabled
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

When .AlarmROCDisabled is set to 1, all rate-of-change events and alarms are ignored. They are not stored in alarm memory, nor are they written to disk. It is very important to re-enable events/alarms, whenever possible, to avoid data loss.

This is the opposite of the .AlarmROCEnabled property.

Data Type

Discrete (read/write)

Valid Values

1= Disable alarms

0= Enable alarms (default)

Example

The following example enables rate-of-change alarms of the Tag2 tag:

```
Tag2.AlarmROCDisabled=0;
```

See Also

.AlarmDisabled, .AlarmEnabled, .AlarmROCEnabled

Changing a tag's alarm limits

Use the following dotfields to change a tag’s alarm limits while an application is running. You can change the limit for LoLo, Low, High, and HiHi alarms, the major and minor deviation percentage and target, and the rate-of-change deviation.

- .LoLoLimit dotfield
- .LoLimit dotfield
- .HiLimit dotfield
- .HiHiLimit dotfield

- .MinorDevPct dotfield
- .MajorDevPct dotfield
- .DevTarget dotfield
- .ROCPct dotfield

.LoLoLimit dotfield

Changes a tag's LoLo alarm limit.

Category

Alarms

Usage

```
TagName.LoLoLimit
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Analog (read/write)

Valid Values

Must be in value range configured for the specified tag.

Example

This statement decreases the LoLo alarm limit for the MyTag1 tag by a value of 10:

```
MyTag1.LoLoLimit=MyTag1.LoLoLimit - 10;
```

See Also

.Alarm, .AlarmValue, .Ack, .LoLoStatus, .LoLoSet, .AlarmDisabled, .AlarmEnabled, .AlarmLoLoDisabled, .AlarmLoLoEnabled, .AlarmLoLoInhibitor

.LoLimit dotfield

Changes a tag's Low alarm limit.

Category

Alarms

Usage

```
Tagname.LoLimit
```

Parameter

Tagname

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Analog (read/write)

Valid Values

Must be in value range configured for the specified tag.

Example

This statement decreases the Low alarm limit for the MyTag tag by a value of 10:

```
MyTag.LoLimit=MyTag.LoLimit - 10;
```

See Also

.Alarm, .AlarmValue, .Ack, .LoStatus, .LoSet, .AlarmDisabled, .AlarmEnabled, .AlarmLoDisabled, .AlarmLoEnabled, .AlarmLoInhibitor

.HiLimit dotfield

Changes a tag's High alarm limit.

Category

Alarms

Usage

```
TagName.HiLimit
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Analog (read/write)

Valid Values

Must be in value range configured for the specified tag.

Example

This statement sets the High limit alarm for the **PumpTemp** tag to 212:

```
PumpTemp.HiLimit = 212;
```

See Also

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor

.HiHiLimit dotfield

Changes a tag's HiHi alarm limit.

Category

Alarms

Usage

```
TagName.HiHiLimit
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Analog (read/write)

Valid Values

Must be in value range configured for the specified tag.

Example

The following statement increases the HiHi alarm limit for the MyTag tag by a value of 5:

```
MyTag.HiHiLimit=MyTag.HiHiLimit + 5;
```

See Also

.Alarm, .AlarmValue, .Ack, .HiHiStatus, .HiHiSet, .AlarmDisabled, .AlarmEnabled, .AlarmHiHiDisabled, .AlarmHiHiEnabled, .AlarmHiHiInhibitor

.MinorDevPct dotfield

Changes a tag's minor deviation alarm limit.

Category

Alarms

Usage

```
TagName.MinorDevPct
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Real (read/write)

Valid Values

0 to 100

Example

The following statement sets the minor deviation limit property for the MyTag tag to 25 percent:

```
MyTag.MinorDevPct=25;
```

See Also

.AckDev, .AlarmDev, .AlarmMinDevDisabled, .AlarmMinDevEnabled, .AlarmMinDevInhibitor, .MinorDevSet, .MinorDevStatus

.MajorDevPct dotfield

Changes a tag's major alarm deviation limit.

Category

Alarms

Usage

```
TagName.MajorDevPct
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Real (read/write)

Valid Values

0 to 100

Example

The following statement sets the major deviation limit property for the MyTag tag to 25 percent:

```
MyTag.MajorDevPct=25;
```

See Also

.AckDev, .AlarmDev, .AlarmMajDevDisabled, .AlarmMajDevEnabled, .AlarmMajDevInhibitor, .MajorDevSet, .MajorDevStatus

.DevTarget dotfield

Changes the target for a tag's minor and major deviation alarms.

Category

Alarms

Usage

```
TagName.DevTarget
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Real (read/write)

Valid Values

Must be in value range specified for the tag.

Example

The following statement sets the deviation target for the MyTag tag to 500;

```
MyTag.DevTarget=500;
```

See Also

.AckDev, .AlarmDev, .AlarmMajDevDisabled, .AlarmMajDevEnabled, .AlarmMajDevInhibitor, .MajorDevSet, .MajorDevStatus

.ROCPct dotfield

Changes a tag's rate-of-change alarm limit.

Category

Alarms

Usage

```
TagName.ROCPct
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

The dotfield corresponds directly to the same field configured within the alarm section of the Tagname Dictionary.

Data Type

Integer (read/write)

Valid Values

0 to 100

Example

The following statement sets the rate-of-change alarm limit of the MyTag tag to 25 percent:

```
MyTag.ROCPct=25;
```

See Also

.ROCStatus, .ROCSet

Changing a tag's alarm deadbands

Use the following dotfields to change a tag's alarm deadband range while an application is running:

- .AlarmValDeadband dotfield
- .AlarmDevDeadband dotfield

.AlarmValDeadband dotfield

Changes a tag's deadband value while an InTouch application is running.

Category

Alarms

Usage

```
TagName.AlarmValDeadband
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Analog (read/write)

Valid Values

Must be in value range specified for the tag.

Example

The following statement changes the deadband for Tag1 tag to a value of 25:

```
Tag1.AlarmValDeadband=25;
```

See Also

.AlarmDevDeadband

.AlarmDevDeadband dotfield

Changes a tag's deviation percentage deadband for both minor and major deviation alarms.

Category

Alarms

Usage

```
TagName.AlarmDevDeadband
```

Parameter

TagName

Any integer, real, or indirect analog tag.

Remarks

If you want to continue to use the run-time value of this dotfield after an intentional or accidental shutdown of WindowViewer, select the **Retentive Parameters** option in the Tagname Dictionary.

Data Type

Integer (read/write)

Valid Values

0 to 100

Example

The following statement changes the deviation deadband percentage to 25 percent:

```
tag.AlarmDevDeadband=25;
```

See Also

.AlarmValDeadband, .AlarmDev

Changing the alarm comment associated with a tag

The **.AlarmComment** dotfield returns a comment text string that is associated with the alarm of a tag or alarm group.

.AlarmComment dotfields

Returns a comment text string that is associated with the alarm of a tag or alarm group. By default, it is empty in a new application.

Associate user-defined information with an alarm instance

You can attach three items to an alarm record: two numbers and a string. Use the following dotfields to add information to an alarm record.

- .AlarmUserDefNumX dotfields
- .AlarmUserDefStr dotfield

.AlarmUserDefNumX dotfields

To simplify setting user values, you can set these dotfields on an alarm group as well as on a specific tag. For example, InBatch could set the batch number in .AlarmUserDefNum1 all the way up at the $System alarm group, causing all alarms to have the batch number attached.

The .AlarmUserDefNum1 and .AlarmUserDefNum2 dotfields correspond to the User1 and User2 columns in the Alarm Viewer control, respectively.

If you set .AlarmUserDefNum1 on an alarm group, it applies to all alarms in that group and any of its sub-groups. You can also specifically set the value of .AlarmUserDefNum1 on a tag. In this case, it applies only to that tag and overwrites any setting of .AlarmUserDefNum1 in the tag's alarm group.

Category

Alarms

Usage

```
TagName.AlarmUserDefNum1
TagName.AlarmUserDefNum2
```

Parameter

TagName

Any discrete, integer, real, indirect discrete, indirect analog tag, or alarm group.

Remarks

This user-defined dotfield is enabled for a wide range of tags, particularly discrete tags, analog tags, and alarm groups (whether or not they have alarms defined). You can leave these items unset, set all of them, or set only some of them for any individual tag, group, or parent group.

The value of this dotfield is attached to the alarm, but ONLY if a value has been set, for example, by a script or a POKE.

Data Types

Analog (read/write)

Valid Values

Any real value and not set (default)

Examples

The following examples use constant values. However, you can use InTouch QuickScripts to copy the value of another tag to any of these user-defined fields. You can also use PtAcc to set or inspect them, or use InTouch as an I/O Server to get or set the values.

```
$System.AlarmUserDefNum1 = 4;
GroupA.AlarmUserDefNum1 = 27649;
```

In concept, the lowest-level setting prevails, when an alarm notification is sent to the Distributed Alarm system. That is, if the tag has .AlarmUserDefNum1 set to some value, the alarm record should be populated using that setting. However, if the tag doesn't have one, WindowViewer checks if the tag's alarm group has one, and so on up the line until the root group $System is reached. If no setting is found at any level, the entry in the alarm record will be left empty (zero for numbers, an empty string for strings).

Note: This hierarchical search is handled independently for each item. Therefore, if a tag has a setting for .AlarmUserDefNum2 but not for .AlarmUserDefNum1, but its parent group has a setting for .AlarmUserDefNum1, the tag will inherit the setting for .AlarmUserDefNum1 from its parent group.

See Also

.AlarmUserDefStr

.AlarmUserDefStr dotfield

The **.AlarmUserDefStr** dotfield is attached to the information recorded for each alarm by Alarm DB Logger in the alarm database. The .AlarmUserDefStr dotfield corresponds to database field User3. You can use the "user-defined" columns in a SELECT statement to select particular collections of alarms for database operations. For example, if $System.AlarmUserDefStr is set to a Batch String and is changed each time the Batch changes, a selection involving the database field User3 can be used to select alarms for particular batches.

Category

Alarms

Usage

```
Tagname.AlarmUserDefStr
```

Parameter

Tagname

Any discrete, integer, real, indirect discrete, or indirect analog tag, or alarm group.

Remarks

This user-defined dotfield is enabled for a wide range of tags, particularly discrete tags, analog tags, and alarm groups (whether or not they have alarms defined). You can leave these items unset, set all of them, or set only some of them for any individual tag, group, or parent group.

The value of this dotfield is attached to the alarm, but ONLY if a value has been set, for example, using a script or a POKE.

Data Type

Message (read/write)

Valid Values

NULL and any valid string

Examples

This example uses a constant value. However, you can use InTouch QuickScripts to copy the value of another tag to any of these user-defined fields. You can also use PtAcc to set or inspect them, or use InTouch as an I/O Server

to get or set the values.

```
Tag04.AlarmUserDefStr = "Joe";
```

In concept, the lowest-level setting prevails, when an alarm notification is sent to the Distributed Alarm system. That is, if the tag has .AlarmUserDefStr set to some value, the alarm record should be populated using that setting. However, if the tag doesn't have one, WindowViewer will check if the tag's alarm group has one, and so on up the line until the root group $System is reached. If no setting is found at any level, the entry in the alarm record will be left empty (zero for numbers, an empty string for strings).

Also note that this hierarchical search is handled independently for each item. Therefore, if a tag has a setting for .AlarmUserDefNum1 but not for .AlarmUserDefStr, but its parent group has a setting for .AlarmUserDefStr, that setting is used in the alarm record.

See Also

.AlarmUserDefNumX

Determine the inhibitor tag of a tag or alarm group

Use the following dotfields to determine the inhibitor tag of various types of alarms.

- .AlarmDscInhibitor dotfield
- .AlarmLoLoInhibitor dotfield
- .AlarmLoInhibitor dotfield
- .AlarmHiInhibitor dotfield
- .AlarmHiHiInhibitor dotfield
- .AlarmMinDevInhibitor dotfield
- .AlarmMajDevInhibitor dotfield
- .AlarmROCInhibitor dotfield

.AlarmDscInhibitor dotfield

Returns the name of the inhibitor tag assigned to a discrete alarm.

Category

Alarms

Usage

```
TagName.AlarmDscInhibitor
```

Parameter

TagName

Any discrete tag or alarm group.

Remarks

Configured in WindowMaker. Cannot be changed during run time.

Data Types

Message (read-only)

Examples

The .AlarmDSCInhibitor dotfield is used by setting .Name to an Indirect tag that is equal to the value of the .AlarmDscInhibitor tag then manipulating the value of the Indirect tag.

The following statement returns the name of the alarm inhibitor tag for a discrete alarm (assuming SomeIndirectTag is an analog indirect tag):

```
SomeIndirectTag.Name = AlarmedTag.AlarmDscInhibitor;
```

The inhibition state of the alarmed tag can be controlled by setting the value of the indirect tag as follows:

```
SomeIndirectTag = 1;
```

Turns on inhibition. Discrete alarms are disabled for AlarmedTag

```
SomeIndirectTag = 0;
```

Turns off inhibition
Discrete alarms can be generated for AlarmedTag

.AlarmLoLoInhibitor dotfield

Returns the name of the inhibitor tag assigned to a LoLo alarm.

Category

Alarms

Usage

```
TagName.AlarmLoLoInhibitor
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group tag.

Remarks

Configured in WindowMaker. Cannot be changed during run time.

Data Types

Message (read-only)

Examples

The **.AlarmLoLoInhibitor** dotfield is used by setting .Name to an Indirect tag that is equal to the value of the .AlarmLoLoInhibitor tag then manipulating the value of the Indirect tag.

The following statement returns the name of the alarm inhibitor tag for a LoLo alarm limit (assuming SomeIndirectTag is an analog indirect tag):

```
SomeIndirectTag.Name = AlarmedTag.AlarmLoLoInhibitor;
```

The inhibition state of the alarmed tag can be controlled by setting the value of the Indirect tag as follows:

```
SomeIndirectTag = 1;
```

Turns on inhibition
LoLo alarms are disabled for AlarmedTag

```
SomeIndirectTag = 0;
```

Turns off inhibition
LoLo alarms can be generated for AlarmedTag

See Also

.AlarmHiInhibitor, .AlarmHiHiInhibitor, .AlarmLoInhibitor

.AlarmLoInhibitor dotfield

Returns the name of the inhibitor tag assigned to a Low alarm.

Category

Alarms

Usage

```
TagName.AlarmLoInhibitor
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group tag.

Remarks

Configured in WindowMaker. Cannot be changed during run time.

Data Types

Message (read-only)

Examples

The **.AlarmLoInhibitor** dotfield is used by setting .Name to an Indirect tag that is equal to the value of the .AlarmLoInhibitor tag then manipulating the value of the Indirect tag.

The following statement returns the name of the alarm inhibitor tag for a Low alarm limit (assuming SomeIndirectTag is an analog indirect tag):

```
SomeIndirectTag.Name = AlarmedTag.AlarmLoInhibitor;
```

The inhibition state of the alarmed tag can be controlled by setting the value of the Indirect tag as follows:

```
SomeIndirectTag = 1;
```

Turns on inhibition
Low alarms are disabled for AlarmedTag

```
SomeIndirectTag = 0;
```

Turns off inhibition
Low alarms can be generated for AlarmedTag

See Also

.AlarmHiInhibitor, .AlarmHiHiInhibitor, .AlarmLoLoInhibitor

.AlarmHiInhibitor dotfield

Returns the name of the inhibitor tag assigned to a High alarm.

Category

Alarms

Usage

```
TagName.AlarmHiInhibitor
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group tag.

Remarks

Configured in WindowMaker. Cannot be changed during run time.

Data Types

Message (read-only)

Example

The **.AlarmHiInhibitor** dotfield is used by setting .Name to an Indirect tag that is equal to the value of the .AlarmHiInhibitor tag then manipulating the value of the Indirect tag.

The following statement returns the name of the alarm inhibitor tag for a High alarm limit (assuming SomeIndirectTag is an analog indirect tag):

```
SomeIndirectTag.Name = AlarmedTag.AlarmHiInhibitor;
```

The inhibition state of the alarmed tag can be controlled by setting the value of the Indirect tag as follows:

```
SomeIndirectTag = 1;
```

Turns on inhibition
High alarms are disabled for AlarmedTag

```
SomeIndirectTag = 0;
```

Turns off inhibition
High alarms can be generated for AlarmedTag

See Also

.AlarmHiHiInhibitor, .AlarmLoInhibitor, .AlarmLoLoInhibitor

.AlarmHiHiInhibitor dotfield

Returns the name of the inhibitor tag assigned to a HiHi alarm condition.

Category

Alarms

Usage

```
TagName.AlarmHiHiInhibitor
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group tag.

Remarks

Configured in WindowMaker. Cannot be changed during run time.

Data Types

Message (read-only)

Example

The .AlarmHiHiInhibitor dotfield is used by setting .Name to an Indirect tag that is equal to the value of the .AlarmHiHiInhibitor tag then manipulating the value of the Indirect tag. The following statement returns the name of the alarm inhibitor tag for a HiHi alarm limit (assuming SomeIndirectTag is an analog indirect tag):

```
SomeIndirectTag.Name = AlarmedTag.AlarmHiHiInhibitor;
```

The inhibition state of the alarmed tag can be controlled by setting the value of the indirect tag as follows:

```
SomeIndirectTag = 1;
```

Turns on inhibition
HiHi alarms are disabled for AlarmedTag

```
SomeIndirectTag = 0;
```

Turns off inhibition
HiHi alarms can be generated for AlarmedTag

See Also

.AlarmHiInhibitor, .AlarmLoInhibitor, .AlarmLoLoInhibitor

.AlarmMinDevInhibitor dotfield

Returns the name of the alarm inhibitor tag associated with a minor deviation alarm condition.

Category

Alarms

Usage

```
TagName.AlarmMinDevInhibitor
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group tag.

Remarks

Configured in WindowMaker. Cannot be changed during run time.

Data Types

Message (read-only)

Example

The .AlarmMinDevInhibitor dotfield is used by setting .Name to an Indirect tag that is equal to the value of the .AlarmMinDevInhibitor tag then manipulating the value of the Indirect tag. The following statement returns the name of the alarm inhibitor tag for a minor deviation alarm limit (assuming SomeIndirectTag is an analog indirect tag):

```
SomeIndirectTag.Name = AlarmedTag.AlarmMinDevInhibitor;
```

The inhibition state of the alarmed tag can be controlled by setting the value of the Indirect tag as follows:

```
SomeIndirectTag = 1;
```

Turns on inhibition
Minor deviation alarms are disabled for AlarmedTag

```
SomeIndirectTag = 0;
```

Turns off inhibition
Minor deviation alarms can be generated for AlarmedTag

See Also

.AlarmMajDevInhibitor

.AlarmMajDevInhibitor dotfield

Returns the name of the alarm inhibitor tag associated with a major deviation alarm condition.

Category

Alarms

Usage

```
TagName.AlarmMajDevInhibitor
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group tag.

Data Types

Message (read-only)

Example

The .AlarmMajDevInhibitor dotfield is used by setting .Name to an Indirect tag that is equal to the value of the .AlarmMajDevInhibitor tag then manipulating the value of the Indirect tag. The following statement returns the name of the alarm inhibitor tag for a major deviation alarm limit (assuming SomeIndirectTag is an analog indirect tag):

```
SomeIndirectTag.Name = AlarmedTag.AlarmMajDevInhibitor;
```

The inhibition state of the alarmed tag can be controlled by setting the value of the Indirect tag as follows:

```
SomeIndirectTag = 1;
```

Turns on inhibition
Major deviation alarms are disabled for AlarmedTag

```
SomeIndirectTag = 0;
```

Turns off inhibition
Major deviation alarms can be generated for AlarmedTag

See Also

.AlarmMinDevInhibitor

.AlarmROCInhibitor dotfield

Returns the name of the alarm inhibitor tag associated with a rate-of-change alarm condition.

Category

Alarms

Usage

```
TagName.AlarmROCInhibitor
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group tag.

Data Types

Message (read-only)

Example

The .AlarmROCInhibitor dotfield is used by setting .Name to an Indirect tag that is equal to the value of the .AlarmROCInhibitor tag then manipulating the value of the Indirect tag. The following statement returns the

name of the alarm inhibitor tag for a rate-of-change alarm limit (assuming SomeIndirectTag is an analog indirect tag):

```
SomeIndirectTag.Name = AlarmedTag.AlarmROCInhibitor;
```

The inhibition state of the alarmed tag can be controlled by setting the value of the Indirect tag as follows:

```
SomeIndirectTag = 1;
```

Turns on inhibition
Rate-of-change alarms are disabled for AlarmedTag

```
SomeIndirectTag = 0;
```

Turns off inhibition
Rate-of-change alarms can be generated for AlarmedTag

Count the number of active or unacknowledged alarms

Use the following dotfields to find out the number of active or unacknowledged alarms while the application is running.

Dotfield	Description
.AlarmTotalCount dotfield	Counts the number of alarms associated with a tag or alarm group.
.AlarmUnAckCount dotfield	Counts the number of unacknowledged alarms associated with a tag or alarm group.
.AlarmValueCount dotfield	Counts the number of value alarms associated with a tag.
.AlarmValueUnAckCount dotfield	Counts the number of unacknowledged value alarms associated with a tag.
.AlarmDscCount dotfield	Counts the number of discrete alarms.
.AlarmDscUnAckCount dotfield	Counts the number of unacknowledged discrete alarms.
.AlarmDevCount dotfield	Counts the number of deviation alarms.
.AlarmDevUnAckCount dotfield	Counts the number of unacknowledged deviation alarms.
.AlarmROCCount dotfield	Counts the number of rate-of-change alarms.
.AlarmROCUnAckCount Dotfield	Counts the number of unacknowledged rate-of-change alarms.

.AlarmTotalCount dotfield

Tracks the total number of active alarms for a specified tag or alarm group.

Category

Alarms

Usage

```
TagName.AlarmTotalCount
```

Parameter

TagName

Any type of tag or alarm group.

Remarks

The count includes value, deviation, rate-of-change, and discrete alarms. It includes both acknowledged and unacknowledged alarms.

Data Types

Integer (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is an analog tag configured for alarms. ATC is also an analog tag, which gets the total number of all active alarms (both UnAck and Ack) present in Tag1.

```
ATC = Tag1.AlarmTotalCount;
```

See Also

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDSCCount, .AlarmDSCUnAckCount, .AlarmValueCount, .AlarmUnAckCount, .AlarmValueUnAckCount, .AlarmROCCount, .AlarmROCUnACkCount

.AlarmUnAckCount dotfield

Tracks the total number of unacknowledged alarms for a specified tag or alarm group.

Category

Alarms

Usage

```
TagName.AlarmUnAckCount
```

Parameter

TagName

Any type of tag or alarm group.

Remarks

The count includes unacknowledged value, deviation, rate-of-change, and discrete alarms.

Data Types

Integer (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is an analog or discrete tag configured for alarms. AUC is an analog tag, which gets the total number of unacknowledged alarms present in Tag1.

```
AUC = Tag1.AlarmUnAckCount;
```

See Also

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmValueCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCCount, .AlarmROCUnACkCount

.AlarmValueCount dotfield

Tracks the total number of active value alarms for a specified tag or alarm group.

Category

Alarms

Usage

```
TagName.AlarmValueCount
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Remarks

This includes the count of HiHi, High, Low, and LoLo alarms. It includes both acknowledged and unacknowledged alarms. For non-expanded summary alarm tags, this count will not exceed 1. However, the count may vary with alarm groups.

Data Types

Integer (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is an analog tag configured for value alarms. AVC is also an analog tag, which gets the total number of all alarm values present in Tag1.

```
AVC = Tag1.AlarmValueCount;
```

See Also

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmROCCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmValueUnAckCount dotfield

Tracks the total number of unacknowledged value alarms for a specified tag or alarm group. This includes the count of HiHi, High, Low, and LoLo alarms.

Category

Alarms

Usage

```
TagName.AlarmValueUnAckCount
```

Parameter

TagName

Any integer, real, indirect analog tag, or alarm group.

Data Types

Integer (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is an analog tag configured for value alarms. AVUC is also an analog tag, which gets the total number of all unacknowledged value alarms present in Tag1.

```
AVUC = Tag1.AlarmValueUnAckCount;
```

See Also

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmROCCount, .AlarmTotalCount, .AlarmValueCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmDscCount dotfield

Tracks the total number of active discrete alarms for a specified tag or alarm group.

Category

Alarms

Usage

```
TagName.AlarmDscCount;
```

Parameter

TagName

Any discrete tag, indirect discrete tag, or alarm group.

Remarks

The count assigned to The **AlarmDscCount** dotfield includes both acknowledged and unacknowledged alarms.

For non-expanded summary alarm tags, this count is always 1. However, the count can vary for alarm groups.

Data Types

Integer (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is a discrete tag configured for discrete alarms. ADC is an analog tag, which gets the total number of active discrete alarms (both unacknowledged and acknowledged) present in Tag1.

```
ADC = Tag1.AlarmDSCCount;
```

See Also

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmValueCount, .AlarmROCUnAckCount, .AlarmTotalCount, .AlarmDscUnAckCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmDscUnAckCount dotfield

Tracks the total number of unacknowledged discrete alarms for a specified tag or alarm group.

Category

Alarms

Usage

```
TagName.AlarmDscUnAckCount
```

Parameter

TagName

Any discrete tag, indirect discrete tag, or alarm group.

Data Types

Integer (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is a discrete tag configured for discrete alarms. ADUC is an analog tag, which gets the total number of unacknowledged discrete alarms present in Tag1.

```
ADUC = Tag1.AlarmDscUnAckCount;
```

See Also

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmValueCount, .AlarmROCCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmDevCount dotfield

Tracks the total number of active deviation alarms for a specified tag or alarm group.

Category

Alarms

Usage

```
TagName.AlarmDevCount
```

Parameter

TagName

Any real tag, integer tag, indirect analog tag, or alarm group.

Remarks

This includes the count of minor and major deviation alarms. It includes both acknowledged and unacknowledged alarms. For non-expanded summary alarm tags, this count is always 1. However, the count can vary with alarm groups.

Data Types

Analog (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is an analog tag configured for Deviation alarms. ADC is also an analog tag, which gets the total number of

active deviation (both unacknowledged and acknowledged) alarms present in Tag1.

```
ADC=Tag1.AlarmDevCount;
```

See Also

.AlarmDSCCount, .AlarmValueCount, .AlarmROCUnAckCount, .AlarmTotalCount, .AlarmDSCUnAckCount, .AlarmValueUnAckCount, .AlarmDevUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmDevUnAckCount dotfield

Tracks the total number of unacknowledged deviation alarms for a specified tag or alarm group. This includes the count of minor and major deviation alarms.

Category

Alarms

Usage

```
TagName.AlarmDevUnAckCount
```

Parameter

TagName

Any real tag, integer tag, indirect analog tag, or alarm group.

Data Types

Analog (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is an analog tag configured for Deviation alarms. ADUC is also an analog tag, which gets the total number of unacknowledged deviation alarms present in Tag1.

```
ADUC = Tag1.AlarmDevUnAckCount;
```

See Also

.AlarmDevCount, .AlarmDSCCount, .AlarmValueCount, .AlarmROCUnAckCount, .AlarmTotalCount, .AlarmDSCUnAckCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmROCCount dotfield

Tracks the total number of active rate-of-change alarms for a specified tag or alarm group. It includes both acknowledged and unacknowledged alarms. For non-expanded summary alarm tags, this count will always be 1. However, the count may vary with alarm groups.

Category

Alarms

Usage

```
TagName.AlarmROCCount
```

Parameter

TagName

Any real tag, integer tag, indirect analog tag, or alarm group.

Data Types

Integer (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is an analog tag configured for rate-of-change alarms. ARC is also an analog tag, which gets the total number of active rate-of-change alarms (both unacknowledged and acknowledged) present in Tag1.

```
ARC = Tag1.AlarmROCCount;
```

See Also

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmValueCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCUnAckCount, .AlarmUnAckCount

.AlarmROCUnAckCount Dotfield

Tracks the total number of unacknowledged rate-of-change alarms for a specified analog tag or alarm group.

Category

Alarms

Usage

```
TagName.AlarmROCUnAckCount
```

Parameter

TagName

Any real tag, integer tag, indirect analog tag, or alarm group.

Data Types

Integer (read-only)

Valid Values

0 or any positive integer

Example

Tag1 is an analog tag configured for rate-of-change alarms. ARUC is also an analog tag, which gets the total number of unacknowledged rate-of-change alarms present in Tag1.

```
ARUC = Tag1.AlarmROCUnAckCount;
```

See Also

.AlarmDevCount, .AlarmDevUnAckCount, .AlarmDscCount, .AlarmDscUnAckCount, .AlarmValueCount, .AlarmTotalCount, .AlarmValueUnAckCount, .AlarmROCCount, .AlarmUnAckCount

Maintain

Migrate and upgrade applications

This section describes about migrating and upgrading different InTouch applications.

Move from a legacy application to the new standalone application

Prior to System Platform 2020, InTouch HMI users could create the following types of applications:

- Standalone
- Modern
- Managed
- Published

Standalone applications were built using legacy symbols and controls. Modern applications supported the use of Industrial Graphics (formerly known as ArchestrA graphics/symbols) in addition to legacy symbols. Managed applications were built using the IDE and Galaxy Objects. Standalone applications could be published into a package and then distributed to other nodes, resulting in Published applications..

In InTouch HMI 2020, modern applications have been redesigned as more comprehensive standalone applications. The new standalone application offers many improvements over legacy standalone applications.

- Easy distribution – Copy and paste the application folder to a different node. No import or export operations needed.
- Use of Industrial Graphics – The new standalone application combines the ease of use of earlier legacy applications with modern industrial graphics.
- Ready for the cloud – Applications created on-premise nodes can now be viewed on a HTML5 compliant browser.
- Light weight – The applications files are light weight and allow for better performance and use.

There is no change in behavior of Managed and Published applications.

Migrate and upgrade older applications

To support applications created in earlier versions of InTouch HMI, you can use two workflows to transition to the new standalone application.

- In-place migration of older modern applications: If the node contains old Modern applications and the product version on the node is upgraded, then you migrate the application using the Application Manager.
- Importing .aapkg files of modern applications exported from earlier versions of InTouch HMI.

Migrate earlier InTouch applications to the current version

You can migrate applications developed with older versions of the InTouch HMI to the current version. When you attempt to open an older application with either WindowMaker or WindowViewer, you are shown the Application Migration dialog box. Here you can:

- Choose to convert the application resolution.
- Create a backup copy before migrating the old application to the current version of the InTouch HMI.

You can migrate existing standalone, modern, or published InTouch applications to the current InTouch version. You must specify the folder to create the backup copy and if you want to exclude any files from the backup.

1. From the application list, double-click on an application.

   The **Application Migration** dialog box appears.
2. To convert the application resolution from the original to the current resolution, select the **Convert the application resolution from** <existing resolution> **to** <new resolution> checkbox.
3. To change the default backup path (<Application Directory>\Bak), clear the check box for **Backup the application before migration**. Then, type the path to the folder in the **Backup Path** box where you want to save the backup. If the folder does not exist, you must create it, and then create the backup.
4. In the **Ignore Files** box, you can specify any files that you want to exclude from the backup. By default, all files in the application directory are backed up. Type the file names you want to exclude separated by a semicolon (;). Or, use standard wild card characters ('*' and '?') to exclude a set of files by the common characters in their names.
5. After configuring the necessary options, select **OK**.

   Import-Sp6_5

Convert legacy alarm displays

When you open an application built with a version before InTouch 7.11 in WindowViewer, a dialog box appears prompting you to run WindowMaker to convert the application. If you continue with the conversion, all of the Standard Alarm Objects are converted to Distributed Alarm Objects with default values. Colors, fonts, expressions, and alarm query settings are not preserved.

Manage application settings

InTouch application settings, such as the application path, are stored in the Win.ini file. The Win.ini file is located in the below directory:

C:\Users\<User Name>\AppData\Local\Wonderware

WindowMaker runs as an administrative user and WindowViewer can run as an administrative or standard user. The standard user cannot access the Win.ini directory of the administrative user profile. Therefore, as the application developer, you need to copy the common Win.ini attributes to the standard user’s Win.ini profile when you develop the application. This ensures that all the attributes that are set under the administrative user are also available when WindowViewer is started by the standard user. You must copy the attributes each time you make changes to the common Win.ini attributes.

Import InTouch applications

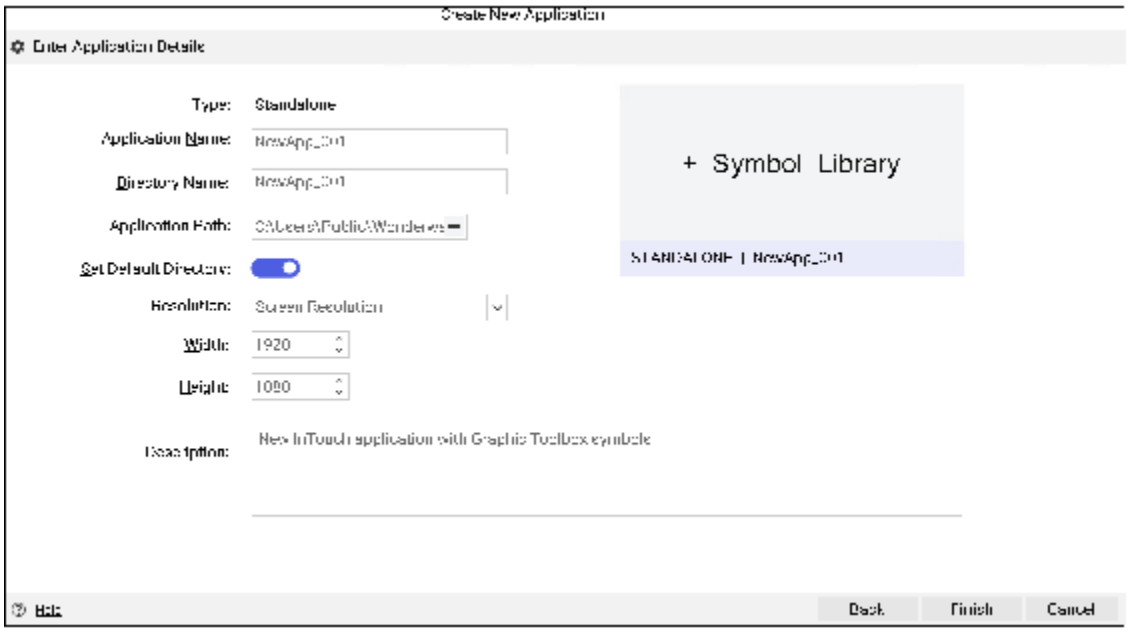
You can import existing modern applications using the Application Manager, which will be converted to a Standalone application. Standalone applications can be copied from one node to another and found using the Find Applications option, they do not need to be imported or exported.

Import an existing modern application:

1. On the **File** menu, in the **Import** group, select **Import**.

   The **Create New Application: Select an application to import** screen appears.
2. Use the **Find Applications** section to search for the application you want to import. Search for a folder or a file to import.
3. Select the application and select **Next**.

   The Enter Application Details screen appears.



4. Make any changes to the settings.

   Select **Finish**.

   A new application is created and displayed in the Application Manager.

Note: All modern applications are imported as a standalone application.

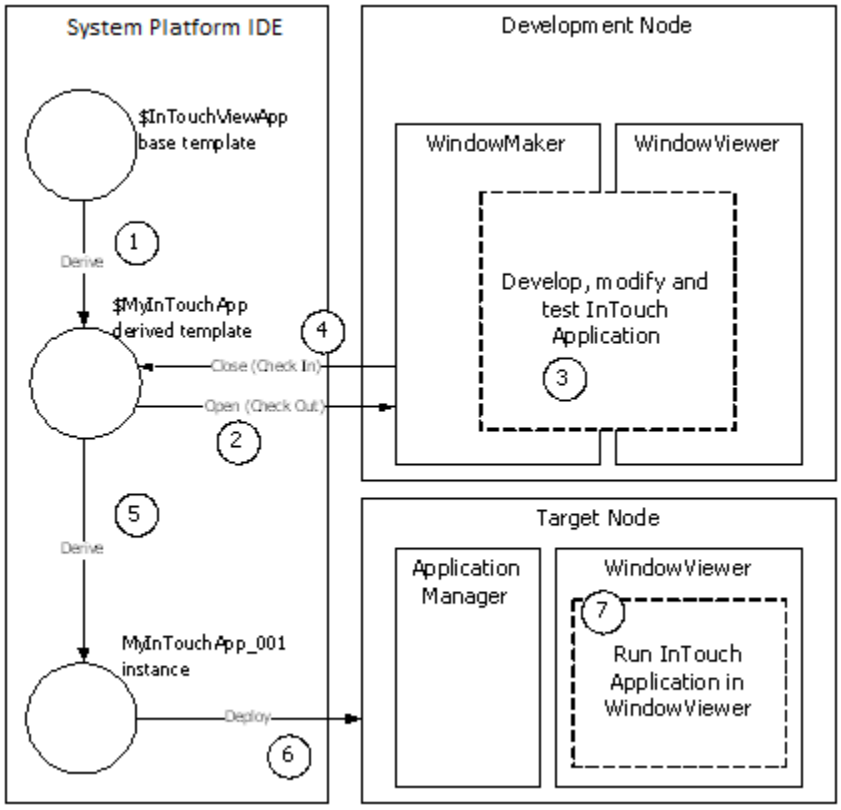
Manage InTouch applications using the System Platform IDE

You can manage your InTouch applications using the IDE. The following procedure shows you how to do this in a general way. For more specific information, see Manage InTouch applications with the IDE.

The InTouch functionality in the System Platform IDE is handled by two AutomationObjects:

- The InTouchViewApp object represents an InTouch application at design time and run time.
- The ViewEngine object controls how an InTouch application runs on a target node in a Galaxy.

The following diagram shows how InTouch applications are managed using the System Platform IDE:



Use the IDE to manage your InTouch applications

1. Create a managed InTouch application in the System Platform IDE.
2. Open it in WindowMaker.
3. Configure your InTouch application in WindowMaker. You can switch to WindowViewer to test the application.
4. Save the InTouch application and close WindowMaker and WindowViewer.
5. Determine which nodes to deploy the InTouch application to.
6. Deploy the InTouch application to the target nodes in the Galaxy.
7. Run the InTouch application in WindowViewer on the target nodes.

InTouchViewApp object

Application Server manages your InTouch applications with a specific type of Application Server object called the InTouchViewApp object.

An InTouchViewApp template references one specific managed InTouch application at design time and cannot be executed at run time.

You must create an instance of the InTouchViewApp template. This instance can be deployed to a target node. The target node is the node on which the managed InTouch application runs in WindowViewer.

To distribute an InTouch application, you create multiple instances of the same template and deploy them to

multiple nodes.

Optionally, you can:

- Export and import the InTouchViewApp object to exchange managed InTouch applications across Galaxies.
- Export and import tag dictionary data as .csv files.
- Export and import windows between different types of InTouch applications.
- Publish the managed InTouch application. The published InTouch application runs like a standalone InTouch application but can contain Industrial Graphics.
- Use the attributes of the deployed InTouchViewApp object to read from and write to InTouch tags with ArchestrA attributes.

Use the InTouchViewApp object

1. Derive an InTouchViewApp template from the $InTouchViewApp base template.
2. Associate the derived template with an InTouch application by creating a new InTouch application or importing a standalone InTouch application.
3. Open the application in WindowMaker.
4. Configure the application in WindowMaker and test it in WindowViewer.
5. Save and close WindowMaker. The InTouchViewApp template is checked in.
6. Derive instances from the InTouchViewApp template.
7. Deploy these instances to selected target nodes within the Galaxy.
8. Run Application Manager on the target nodes and run the managed InTouch applications in WindowViewer.

Associate an InTouchViewApp template with an InTouch application

After you create a new InTouchViewApp template, you can associate an InTouchViewApp template with an InTouch application by:

- Creating a new InTouch application.
- Importing a standalone InTouch application.

The InTouchViewApp template does not contain the InTouch application data itself, such as the tag configuration and values, but simply references the application.

Edit a managed InTouch application

You edit a managed InTouch application with WindowMaker as you do for a standalone InTouch application, except that you open the editor of the InTouchViewApp template to start the associated InTouch application in WindowMaker.

When you close WindowMaker after making changes to the InTouch application, the InTouchViewApp object is automatically checked in.

Test a managed InTouch application

You can test a managed InTouch application with WindowViewer as you can with standalone InTouch applications.

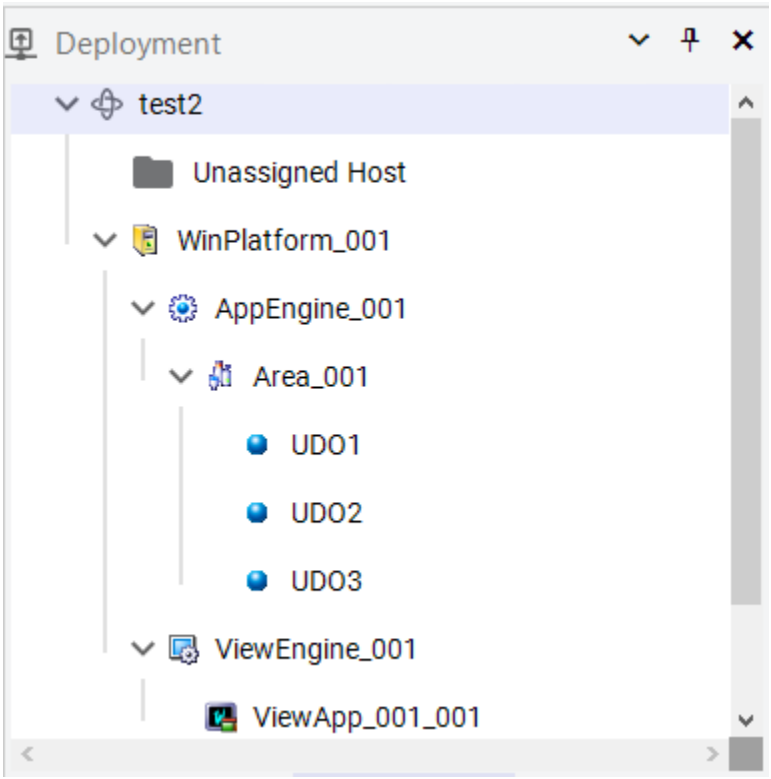
You can fast switch between WindowMaker and WindowViewer to test a managed application if you opened WindowMaker from the System Platform IDE.

If a managed InTouch application contains references to Application Server data, such as `galaxy:UDA`, then a WinPlatform object needs to be deployed to the node you are editing the InTouch application on. Otherwise, the data shows blank values.

Deploy the InTouchViewApp object

After you derive an instance of your InTouchViewApp template, you can assign it to the target platform under a ViewEngine object.

You cannot assign multiple InTouchViewApp instances that have the same parent under one ViewEngine. Instead create a second ViewEngine instance to host additional InTouchViewApp instances with the same parent.



After you deploy the InTouchViewApp object, you can open the InTouch Application Manager on the target node. The associated managed InTouch application appears in the list together with the time stamp of its last deployment in the **Date Modified** column.

When you deploy the InTouchViewApp instance to a target node, the InTouch application is contained in:

- A folder on the development node. This contains the source for the InTouchViewApp template.
- A folder on the target node from which the InTouch application runs. This contains an instance copy of the InTouch application.

Export and import an InTouchViewApp object

You can export the InTouchViewApp object. You do this, for example, to use the managed InTouch application together with its hosting InTouchViewApp object in other Galaxies.

When you export the object, a package file (.aaPKG) is created containing information about the object, the associated managed InTouch application, and any Industrial graphics the application uses.

When you import an InTouchViewApp object, the System Platform IDE also imports the managed InTouch application.

Attributes of the InTouchViewApp object

You can use the Application Server attributes of the InTouchViewApp object to access the run-time data of the tags of the associated InTouch application. This is useful for reading and writing InTouch data directly in the Galaxy name space and provides the same functionality as the InTouchProxy object.

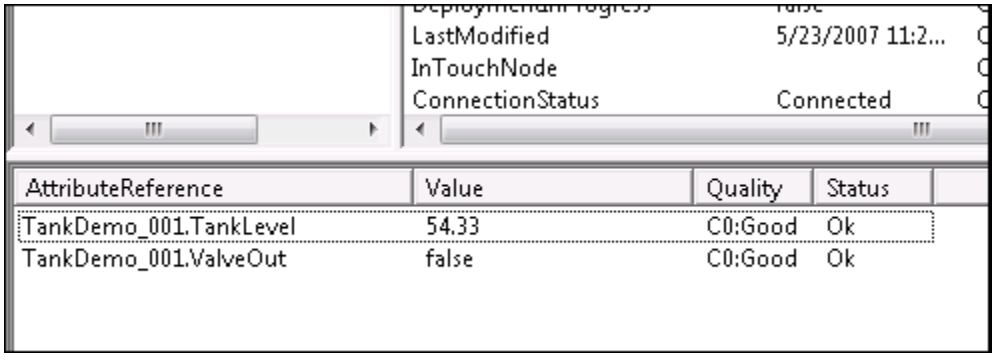
In this example, a deployed managed InTouch application uses a real tag TankLevel to report the fill level of a tank and a discrete tag ValveOut to control the state of a valve.

Read and write the InTouch tags from the InTouchViewApp object instance

1. Right-click the deployed InTouchViewApp object, and then select **Monitor**.

   The **Object Viewer** screen appears.
2. Right-click in the Watch area, and then select **Add Attribute Reference**.

   The **Add Attribute Reference** dialog box appears.
3. In the **Attribute Reference** box, type the name of the InTouchViewApp object followed by a dot and the name of the InTouch tag you want to read or write. For example, `TankDemo_001.TankLevel`.
4. Select **OK**. The attribute is added to the Watch area.
5. Repeat steps 2 through 4 for any other InTouch tags you want to read or write.
6. You can now view the InTouch tag value.



7. Do the following to write to an InTouch tag value,
   a. Double-click it. The **Modify Value** dialog box appears.
   b. Type a new value and select **OK**. The value is written back to the tag of the running InTouch application.

Differences between the InTouchViewApp object and other AutomationObjects

The InTouchViewApp object is unlike other AutomationObjects. You cannot perform some operations you can normally do with other AutomationObjects.

- If you try to configure an InTouchViewApp instance, a message appears asking if you want to open its parent template instead. You cannot configure the instance directly, only the parent template.
- If you try to open more than one InTouchViewApp template for configuration at a time on one node, IDE prevents you from doing so. Close WindowMaker, WindowViewer, and Application Manager and retry. Alternately, you can edit the InTouchViewApp object on a different node with InTouch WindowMaker.
- If you close the IDE while editing an InTouch application with WindowMaker, WindowMaker prompts you to save any changes. It then closes and the InTouchViewApp template is checked in.
- If you close the IDE while testing an InTouch application with WindowViewer, WindowViewer closes.

If you want to:

- Change the association between the InTouchViewApp and the InTouch application, create a new derived InTouchViewApp template instead.
- Use ArchestrA security (Galaxy security) in InTouch, deploy a WinPlatform instance to the node on which the deployed managed InTouch application is running.

ViewEngine object

The ViewEngine is an Application Server object that hosts and runs deployed InTouchViewApp objects.

To deploy an InTouchViewApp instance to a target platform, you need to assign it to a ViewEngine object first. The ViewEngine object is then assigned to the target WinPlatform object.

The ViewEngine fulfils the same functions for the InTouchViewApp instances as the AppEngine instance does for the Application Objects. The ViewEngine:

- Sets up and initializes the InTouchViewApp objects when they are initially deployed and started, so that they can communicate with other objects in the Galaxy.
- Performs diagnostics on attributes that can be monitored, alarmed, and historized.
- Historizes data to the Historian.

You can use different ViewEngine objects to:

- Historize data to different Historians.
- Interact with deployed InTouch applications at different scan rates. This sets at which frequency InTouch tag data can interact with the Galaxy name space.

A platform can host multiple ViewEngine objects. Every InTouchViewApp must be assigned to a ViewEngine.

You cannot create multiple instances of the same InTouchViewApp template to run under the same ViewEngine object. But you can run multiple instances of the same template under different ViewEngine objects.

Export and import InTouch components

You can build InTouch applications more quickly by importing or exporting some or all of the components of an existing application. You can import tag definitions, windows, scripts, application style libraries, Industrial graphics, client controls, localization strings, HTML5 widgets, and script function libraries from your existing application to a new application. Tag definitions are imported and exported from the Application Manager, other components are imported and exported via WindowMaker.

Export and import tag data associated with a managed InTouch application

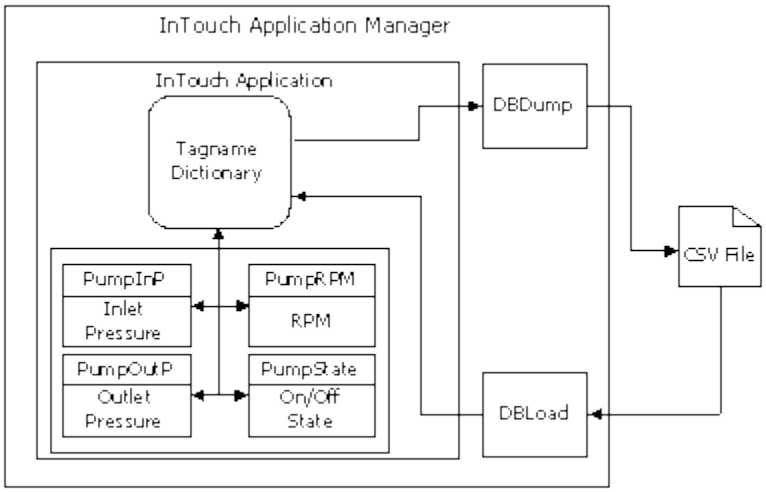
You can export tag data that is associated with a managed InTouch application to a .csv file. This is equivalent to the DB Dump function of the InTouch Application Manager.

You can import the exported tag data from a .csv file back into a managed InTouch application in the same way as the DB Load function.

The exported .csv files from a managed InTouch application and those of a standalone InTouch application are fully interchangeable.

Export tag definitions

The figure below shows the steps to export and import tag definitions between an interim export file and an application's Tagname Dictionary.



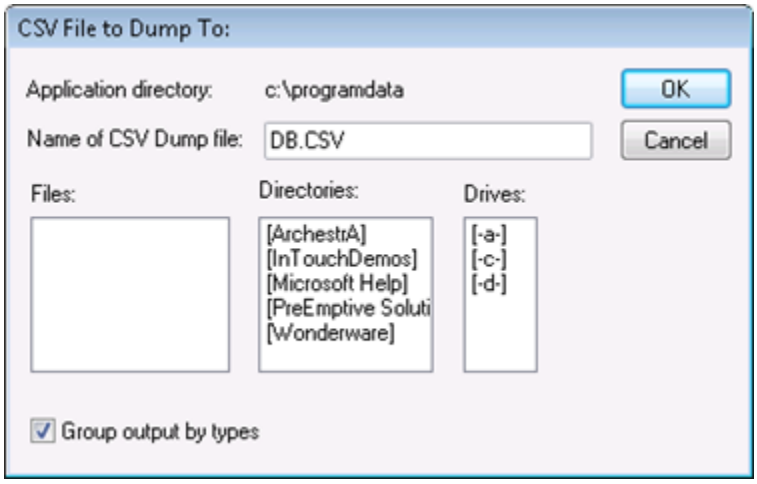
You use the DBDump utility within the Application Manager to export the contents of the Tagname Dictionary to a Comma Separated Value (CSV) file. You can view and edit the exported file with Microsoft Notepad or Microsoft Excel. After making edits, you then import the tag definitions to an InTouch application with the DBLoad utility, which is also an Application Manager utility.

You must convert an application to the current version of the InTouch HMI software before you can export the tag definitions.

Export tag definitions

1. Close WindowMaker and WindowViewer.

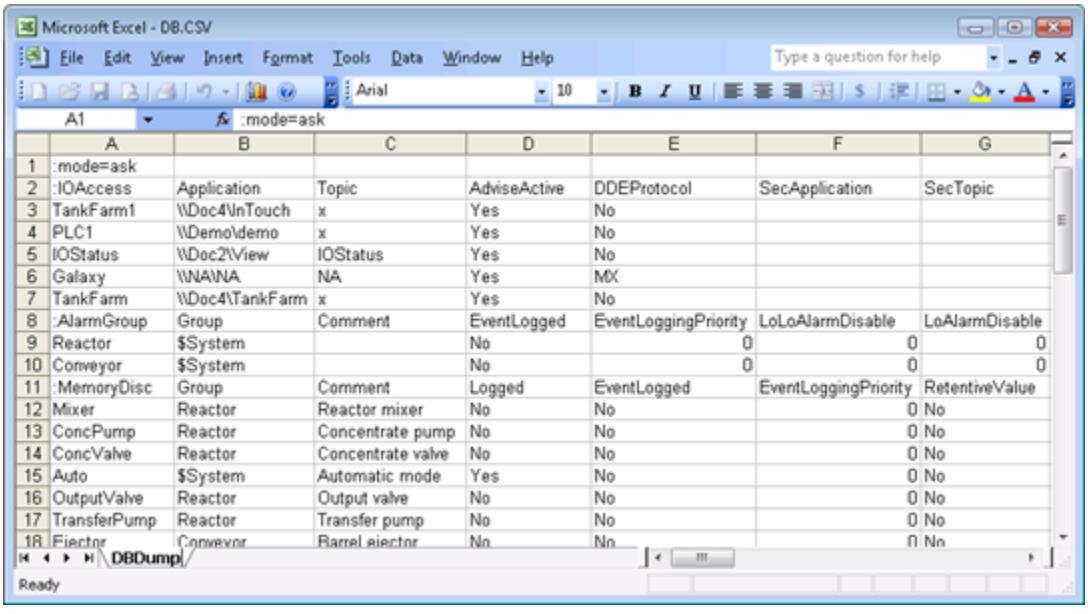
2. Start Application Manager. The **Application Manager** dialog box shows a list of InTouch applications.
3. Select the application from the list.
4. On the **File** menu, in the **Data** group, select **DBDump**. The **CSV File to Dump To**: dialog box appears.



5. In the **Name of CSV Dump file** box, type a name for the file with a .csv file name extension.
6. Select the type of data grouping in the export file.
   - Select the **Group output by types** check box to group the data by the types of tags in the export file. This is the default.
   - Clear **Group output by types** to save the output to the export file alphabetically by tag name.
7. Select **OK** to save the contents of the Tagname Dictionary to the selected file. A message appears indicating the contents were saved successfully to the file.

View exported tag definitions

If you use Microsoft Excel to view an export file created with the DBDump utility, each data record appears in a separate spreadsheet cell.



	A	B	C	D	E	F	G
1	:mode=ask						
2	:IOAccess	Application	Topic	AdviseActive	DDEProtocol	SecApplication	SecTopic
3	TankFarm1	\\Doc4\InTouch	x	Yes	No		
4	PLC1	\\Demo\demo	x	Yes	No		
5	IOStatus	\\Doc2\View	IOStatus	Yes	No		
6	Galaxy	\\NA\NA	NA	Yes	MX		
7	TankFarm	\\Doc4\TankFarm	x	Yes	No		
8	:AlarmGroup	Group	Comment	EventLogged	EventLoggingPriority	LoLoAlarmDisable	LoAlarmDisable
9	Reactor	$System		No	0	0	0
10	Conveyor	$System		No	0	0	0
11	:MemoryDisc	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue
12	Mixer	Reactor	Reactor mixer	No	No	0	No
13	ConcPump	Reactor	Concentrate pump	No	No	0	No
14	ConcValve	Reactor	Concentrate valve	No	No	0	No
15	Auto	$System	Automatic mode	Yes	No	0	No
16	OutputValve	Reactor	Output valve	No	No	0	No
17	TransferPump	Reactor	Transfer pump	No	No	0	No
18	Ejector	Conveyor	Barrel ejector	No	No	0	No

The file consists of keywords, their attributes, and data from the Tagname Dictionary arranged in column order beneath keyword attributes.

Notice the :MemoryDisc keyword in the example of the Excel spreadsheet. This keyword identifies memory discrete tags that were exported from a Tagname Dictionary. On the same spreadsheet row, the attributes of a memory discrete tag appear in separate spreadsheet columns. For example, the Logged attribute column shows whether a memory discrete tag's data is logged or not.

Immediately beneath the keyword and attributes row are the exported tags and their associated properties. In the example of the Excel spreadsheet, OutputValve is a memory discrete tag whose data is not logged.

You can view or edit the export file created by DBDump with any program that supports the .csv file format. Typically, Excel is used because its columnar spreadsheet format makes it easy to organize tag data. But, you can also use Microsoft Notepad if you prefer to view or edit the file's contents in its native comma-delimited string format.

Import tag definitions

You can use the DBLoad utility within the Application Manager to import a .csv file of tag definitions into an application's Tagname Dictionary. You can import a definition file that you originally created with the DBDump utility. Or, you can create your own import file.

You can also use the DBLoad utility to create SuperTag instances. For more information, see Create SuperTag instances.

Tagname dictionary import file format

You can manually create DBLoad import files with any program that supports a .csv file format. If you use Excel to create an import file, each entry is placed in a separate spreadsheet cell. This makes it much easier to read, and there is less chance of error.

For more information on creating import files, see Create an import file template.

The DBLoad import file contains a set of keywords that organize Access Names, alarm groups, and tag data within the file.

- A colon (:) precedes all keywords.
- To continue a line, enter a backslash (\) at the end of the line.
- To enter comments, precede them with a semi-colon (;).

The following table lists the keywords within a DBLoad import file. The table lists the keywords in the order they are specified when you create the file with DBDump. But you can specify keywords in any order within the file.

Keyword	Description
:mode	Specifies how duplicate tag records are handled when importing the contents of the DBLoad file to an application's Tagname Dictionary.
:IOAccess	Access names defined for the InTouch application.
:AlarmGroup	Alarm groups defined for the InTouch application.

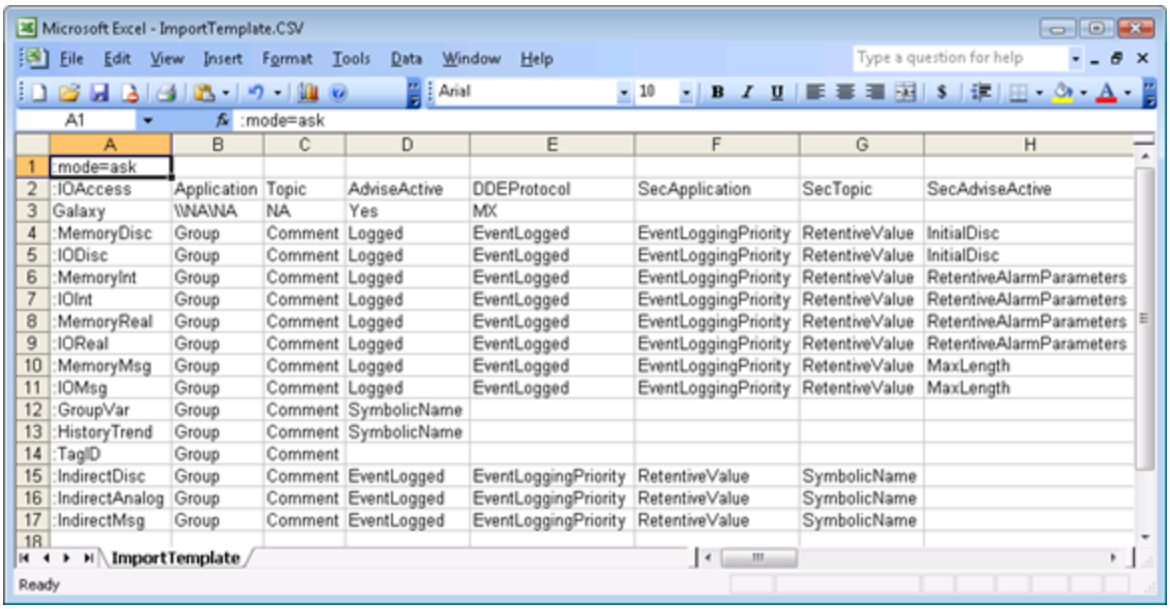
Keyword	Description
:MemoryDisc	Memory discrete tags.
:IODisc	I/O discrete tags.
:MemoryInt	Memory integer tags.
:IOInt	I/O integer tags.
:MemoryReal	Memory real tags.
:IOReal	I/O real tags.
:MemoryMsg	Memory message tags.
:IOMsg	I/O message tags.
:GroupVar	Group Var tags.
:HistoryTrend	Hist Trend tags.
:TagID	Tag ID tags.
:IndirectDisc	Indirect discrete tags.
:IndirectAnalog	Indirect analog tags.
:IndirectMsg	Indirect message tags.

Each keyword includes a set of associated attributes that specify the properties of Access Names, alarm groups, and tags. For example, the :IOAccess keyword includes attributes to specify the application, topic, and communication protocol, which are properties of every InTouch Access Name.

Create an import file template

You can manually create Tagname Dictionary import files with any application that supports the .csv file format. But, creating an entire import file can be time consuming and prone to errors. Using an existing .csv file as a template is faster and more reliable.

The following figure shows a template import file created by DBDump. The figure shows a file created from an InTouch application that has no windows nor tags. The resulting file only includes the required keywords and attributes without tag data.



	A	B	C	D	E	F	G	H
1	:mode=ask							
2	:IOAccess	Application	Topic	AdviseActive	DDEProtocol	SecApplication	SecTopic	SecAdviseActive
3	Galaxy	\\NA\NA	NA	Yes	MX			
4	:MemoryDisc	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	InitialDisc
5	:IODisc	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	InitialDisc
6	:MemoryInt	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	RetentiveAlarmParameters
7	:IOInt	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	RetentiveAlarmParameters
8	:MemoryReal	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	RetentiveAlarmParameters
9	:IOReal	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	RetentiveAlarmParameters
10	:MemoryMsg	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	MaxLength
11	:IOMsg	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	MaxLength
12	:GroupVar	Group	Comment	SymbolicName				
13	:HistoryTrend	Group	Comment	SymbolicName				
14	:TagID	Group	Comment					
15	:IndirectDisc	Group	Comment	EventLogged	EventLoggingPriority	RetentiveValue	SymbolicName	
16	:IndirectAnalog	Group	Comment	EventLogged	EventLoggingPriority	RetentiveValue	SymbolicName	
17	:IndirectMsg	Group	Comment	EventLogged	EventLoggingPriority	RetentiveValue	SymbolicName	
18								

After creating a template, you then manually add tag data beneath the keyword that identifies the type of tag. You insert the properties of your tags in the corresponding attribute columns associated with the tag type keywords.

Create a template import file

1. Open the Application Manager.
2. Create a new InTouch application.

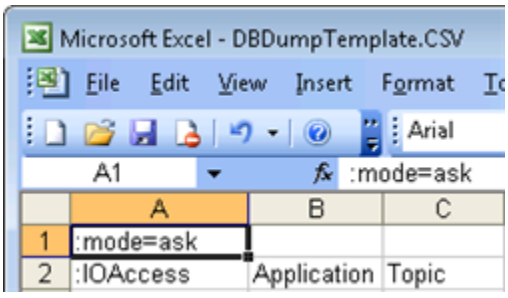
   For more information about the steps to create an application, see Create an InTouch application.
3. Select the new application from the list shown in Application Manager.
4. Export the contents of the application's Tagname Dictionary with the DBDump utility.

   For more information about exporting tags, see Export tag definitions.
5. Edit the file to insert tag data that you want to import.

Set the operating mode for dictionary import files

You must specify how DBLoad handles duplicate tag records while loading data from the import file into an application's Tagname Dictionary.

If you use a import file template created with DBDump, the first line of the file contains the :**mode** keyword. For example, you can assign the value ask to the :mode keyword in cell A1 of the Excel application.



You can assign the following values to a :mode keyword:

```
:MODE=REPLACE
:MODE=UPDATE
:MODE=ASK
:MODE=IGNORE
:MODE=TERMINATE
:MODE=TEST
```

:MODE=REPLACE

If a duplicate tag is encountered, the DBLoad utility deletes the existing tag in the Tagname Dictionary and replaces it with the tag from the import file with the same name.

:MODE=UPDATE

If a duplicate tag is encountered, the DBLoad utility overwrites the existing tag definition in the Tagname Dictionary only with data explicitly specified from the import file. All other data associated with the tag remains unchanged in the Tagname Dictionary.

Fields are considered explicitly defined if the field is in the record and entered by you or is set by the ":KEYWORD=value" mechanism. If a field is not specified in the record, and the keyword has been reset using the ":KEYWORD=" command, the current field value is not updated.

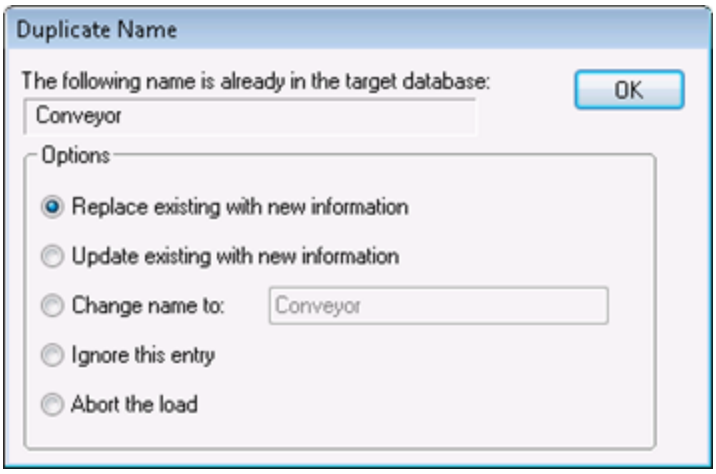
The following is an example of what occurs when an import file in the update mode is loaded/merged into the Tagname Dictionary:

```
:Mode=update
:Group=Group1
:IODisc,Group,DConversion
Tagname1,Group2,
; Tagname1's Group updated to Group2 only
Tagname2,,
; Tagname2's Group updated to Group1 and the DConversion left as is
Tagname3,,Reverse
; Tagname3's Group updated to Group1 and the DConversion to "Reverse"
; the following line "resets" the Group field to its default value
:Group=
; Data field "Group" is reset to its default value
Tagname4,,
; Tagname4 will be left alone
```

The tag types must be compatible if the type is being changed and the tag is in use. For example, an existing historical trend tag cannot be changed to an I/O Integer if the tag is in use by the application. Also, a tag cannot be changed to ReadOnly=yes if the tag is being used on an input link in the application. Because of these restrictions, update the use counts for the target application before running the DBLoad utility.

:MODE=ASK

DBLoad stops when a duplicate tag is encountered while loading the Tagname Dictionary. The **Duplicate Name** dialog box appears and shows a list of options to handle duplicate tags. This is the default import mode.



Options for handing duplicates are:

- Select **Replace existing with new information** to replace the existing tag record with the record from the import file.
- Select **Update existing with new information** to overwrite the existing tag record with only the fields that are explicitly defined in the import file.
- Select **Change Name to** replace the name of the imported tag with the name you type in the box of the **Duplicate Name** dialog box.
- Select **Ignore this entry** to ignore the tag and continue importing the contents of the file.
- Select **Abort the Load** to cancel the import process.

:MODE=IGNORE

The DBLoad import utility ignores the duplicate tag and continues processing the remaining records of the import file.

:MODE=TERMINATE

The DBLoad import operation stops when a duplicate tag is encountered.

:MODE=TEST

DBLoad scans the import file for errors and does not attempt to load tag definitions into the Tagname Dictionary. DBLoad generates a report that identifies any format errors by line number and location in the import file.

Run DBLoad with :**mode=test** first to identify any errors in the import file. After correcting any errors, change the **mode** keyword value to :**mode=replace** or :**mode=update** before running DBLoad.

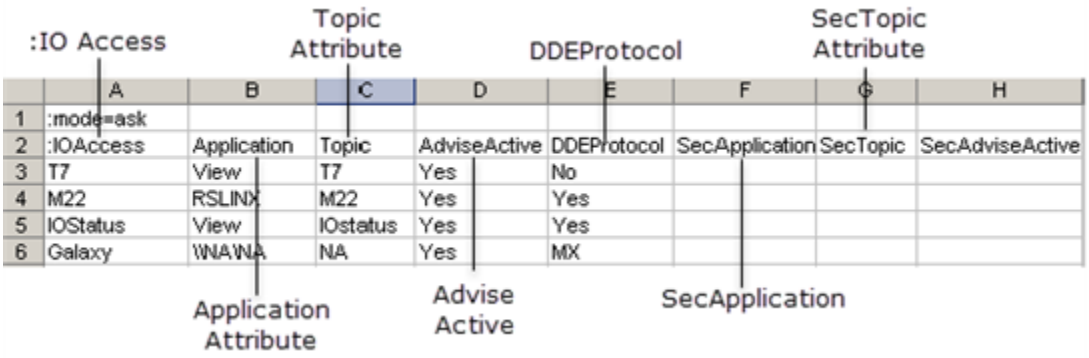
Set access names and alarm groups

The DBLoad import file includes keywords that specify an InTouch application's defined Access Names and alarm groups.

:IOAccess keyword attributes

The **:IOAccess** keyword identifies the Access Names defined for an InTouch application. The **:IOAccess** keyword includes a set of attributes that describe the characteristics of a defined InTouch Access Name.

The following figure shows how Access Names are defined in an Excel spreadsheet with the :IOAccess keyword. Attributes are specified left to right in separate spreadsheet columns.



	A	B	C	D	E	F	G	H
1	:mode=ask							
2	:IOAccess	Application	Topic	AdviseActive	DDEProtocol	SecApplication	SecTopic	SecAdviseActive
3	T7	View	T7	Yes	No			
4	M22	RSLINX	M22	Yes	Yes			
5	IOStatus	View	IOstatus	Yes	Yes			
6	Galaxy	WNAWA	NA	Yes	MX			

The following table shows the list of attributes associated with the :IOAccess keyword. The table lists the attributes in the order they are specified when using a template import file created with the DBDump utility.

String Position	Attributes	Acceptable Values	Default Values
1	Application	Application name defined for the Access Name	None
2	Topic	Topic name defined for the Access Name	None
3	AdviseActive	What information to poll from the server No = Advise all items Yes = Advise only active items	Yes
4	DDEProtocol	Communication protocol defined for the Access Name No = Suitelink Yes = DDE MX = Message Exchange	No
5	SecApplication	Application name defined for the secondary source of the Access Name	None

String Position	Attributes	Acceptable Values	Default Values
6	SecTopic	Topic name defined for the secondary source of the Access Name.	None
7	SecAdviseActive	When to poll information stored on the secondary server NO = Advise all items YES = Advise only active items	None
8	SecDDEProtocol	Communication protocol defined for the secondary source of the Access Name NO = Suitelink YES = DDE MX = Message Exchange	None
9	FailoverExpression	Failover expression that switches the Access Name to the secondary source when TRUE	None
10	FailoverDeadband	Integer number of seconds before starting failover to the secondary source defined by the Access Name	None
11	DFOFlag	Disable Failover flag Yes = Disable Failover flag set No = Disable Failover flag not set	None
12	FBDFlag	Switch back to Primary flag YES = Switch back to the Primary source after the failover condition clears NO = Do not switch back to the Primary source after the failover	None

String Position	Attributes	Acceptable Values	Default Values
		condition clears	
13	FailbackDeadband	Integer number of seconds before switching back to the primary Access Name source after the failover condition clears	No value

:AlarmGroup keyword attributes

The DBLoad import file contains a keyword that identifies the alarm groups defined for an InTouch application. The **:AlarmGroup** keyword includes a set of attributes that describe the characteristics of the InTouch application's alarm groups.

The following table shows the list of attributes associated with the :AccessGroup keyword. The table lists the attributes in the order they are specified when using a template import file created with the DBDump utility.

String Position	Attributes	Acceptable Values	Default Values
1	Group	Name of the alarm group	$System
2	Comment	Comment assigned to the alarm group Any text string	None
3	EventLogged	Event logging enabled or disabled Yes or On = Event logging enabled No or Off = Event logging disabled	No
4	EventLoggingPriority	Priority assigned to events 1 to 999, 0 = not logged	0

String Position	Attributes	Acceptable Values	Default Values
5	LoLoAlarmDisable	LoLo alarm disabled or enabled 0 = LoLo alarm enabled 1 = LoLo alarm disabled	0
6	LoAlarmDisable	Low alarm disabled or enabled 0 = Low alarm enabled 1 = Low alarm disabled	0
7	HiAlarmDisable	High alarm disabled or enabled 0 = High alarm enabled 1 = High alarm disabled	0
8	HiHiAlarmDisable	HiHi alarm disabled or enabled 0 = HiHi alarm enabled 1 = HiHi alarm disabled	0
9	MinDevAlarmDisable	Minor Deviation alarm disabled or enabled 0 = Minor Deviation alarm enabled 1 = Minor Deviation alarm disabled	0
10	MajDevAlarmDisable	Major Deviation alarm disabled or enabled 0 = Major Deviation alarm enabled 1 = Major Deviation alarm disabled	0
11	RocAlarmDisable	Rate of Change alarm disabled or enabled 0 = ROC alarm enabled 1 = ROC alarm disabled	0

String Position	Attributes	Acceptable Values	Default Values
12	DSCAlarmDisable	Discrete alarms disabled or enabled 0 = Discrete alarm enabled 1 = Discrete alarm disabled	0
13	LoLoAlarmInhibititor	Name of the tag used to inhibit LoLo alarms Tag reference: any discrete or analog tag	None
14	LoAlarmInhibitor	Name of the tag used to inhibit Low alarms Tag reference: any discrete or analog tag	None
15	HiAlarmInhibitor	Name of the tag used to inhibit High alarms Tag reference: Any discrete or analog tag	None
16	HiHiAlarmInhibitor	Name of the tag used to inhibit HiHi alarms Tag reference: Any discrete or analog tag	None
17	MinDevAlarmInhibitor	Name of the tag used to inhibit Minor Deviation alarms Tag reference: Any discrete or analog tag	None
18	MajDevAlarmInhibitor	Name of the tag used to inhibit Major Deviation alarms Tag reference: Any discrete or analog tag	None
19	RocAlarmInhibitor	Name of the tag used to inhibit Rate of Change alarms Tag reference: Any discrete or analog tag	None

String Position	Attributes	Acceptable Values	Default Values
20	DSCAlarmInhibitor	Name of the tag used to inhibit discrete alarms Tag reference: any discrete or analog tag	None

Define tag type keywords and attributes

Tag records begin with a keyword line that identifies the type of tag. Each tag keyword includes a unique set of attributes to specify the characteristics of the data associated with the type of tag.

In the following example, the :**IODisc** keyword identifies the I/O discrete tag type. The remaining values in the keyword line identify the attributes of the data associated with an I/O discrete tag. This example shows the contents of the file with Notepad in its native comma-delimited string format.

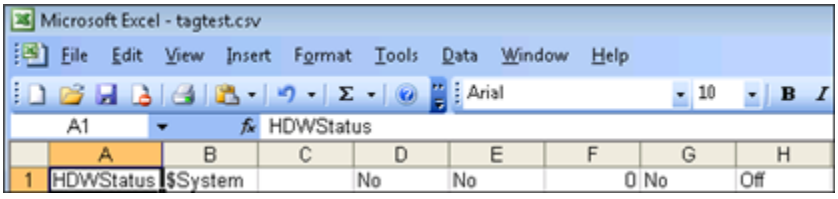
```
:IODisc,Group,Comment,Logged,EventLogged, EventLoggingPriority,RetentiveValue,InitialDis,
OffMsg,OnMsg,AlarmState,AlarmPri,DConversion, AccessName,ItemUseTagname,ItemName,ReadOnly,
AlarmComment,AlarmAckModel,DSCAlarmDisable, DSCAlarmInhibitor,SymbolicName
```

Beneath the tag type keyword line, individual rows specify the tags of that type with a set of attribute values. In the following example, the **HDWStatus** tag belongs to the I/O Discrete tag type in the import file.

```
"HDWStatus","$System","",No,No,0,No,Off,"","",,1,Direct,"HistdataViewstr",No,"Status",No,""
,0,0,"",""
```

The record uses quotation marks to identify a blank string.

The following figure shows the same import file data in an Excel spreadsheet. The **Comment** cell is blank because no tag comment is specified in the import file.



Tag keyword attributes

The following table lists all attributes associated with InTouch tag keywords. The table include columns that describe the type of data associated with each tag attribute and its default value.

These tag attributes can be specified in any order in your DBLoad import file as long as the accompanying tag data matches its corresponding attribute. For example, if you insert a :IODisc keyword in an Excel import file, then all I/O discrete tags' engineering units must be placed in the same Excel column as the EngUnits attribute.

Attribute	Acceptable Value	Default Value
AccessName	InTouch Access Name assigned to tag	None
AlarmAckModel	Alarm acknowledgement model Integer 0 = Condition 1 = Event Oriented 2 = Expanded Summary	0
AlarmComment	Alarm comment assigned to the tag Text string	None
AlarmDevDeadband	Tag deviation alarm deadband Real	None
AlarmPri	Alarm priority assigned to the tag 1 to 999	1
AlarmState	Tag alarm state On, Off, or None	None
AlarmValueDeadband	Tag alarm deadband Real	0
Comment	Comment assigned to the tag Text string	None
Conversion	Tag value conversion Linear or Square Root	Linear
Deadband	Value deadband assigned to the tag Real	0
DevTarget	Tag deviation target value Real	0
DSCAlarmDisable	Discrete alarms disabled or enabled 0 = Discrete alarm enabled 1 = Discrete alarm disabled	0
DSCAlarmInhibitor	Name of the tag used to inhibit a discrete alarm	None

Attribute	Acceptable Value	Default Value
EngUnits	Engineering units assigned to tag Text string	None
EventLogged	Event logging enabled or disabled Yes or On = Event logging enabled No or Off = Event logging disabled	No
EventLogging	Tag event logging enabled or disabled No or Off = Logging disabled Yes or On = Logging enabled	No
EventLoggingPriority	Priority assigned to events 1 to 999, 0 = not logged	0
Group	Name of the alarm group in which the tag belongs	$System
HiAlarmDisable	High alarm disabled or enabled 0 = High alarm enabled 1 = High alarm disabled	0
HiAlarmInhibitor	Name of the tag used to inhibit High alarm Any discrete or analog tag	None
HiAlarmPri	Priority assigned to High alarm 1 to 999	1
HiAlarmState	High alarm enabled or disabled No or Off = Disabled Yes or On = Enabled	No
HiAlarmValue	High alarm point assigned to tag Real	0
HiHiAlarmDisable	HiHi alarm disabled or enabled 0 = HiHi alarm enabled 1 = HiHi alarm disabled	0
HiHiAlarmInhibitor	Name of the tag used to inhibit HiHi alarm Any discrete or analog tag	None

Attribute	Acceptable Value	Default Value
HiHiAlarmPri	Priority assigned to HiHi alarm 1 to 999	1
HiHiAlarmState	HiHi alarm enabled or disabled No or Off = Disabled Yes or On = Enabled	No
HiHiAlarmValue	HiHi alarm point assigned to tag Real	0
InitialDisc	Initial value assigned to discrete tag 0, Off, False, or No = Off 1, On, True, or Yes = On	0
InitialMessage	Initial tag message Text string	None
InitialValue	Initial value assigned to the tag Real	0
ItemName	Name of the item assigned to the tag Text string	None
ItemUseTagname	**Use Tagname as Item Name** option enabled or disabled No or False = Disabled Yes or True = Enabled	No
LoAlarmDisable	Low alarm disabled or enabled 0 = Low alarm enabled 1 = Low alarm disabled	0
LoAlarmInhibitor	Name of the tag used to inhibit Low alarm Any discrete or analog tag	None
LoAlarmPri	Priority assigned to Low alarm 1 to 999	1
LoAlarmState	Low alarm enabled or disabled No or Off = Disabled Yes or On = Enabled	No

Attribute	Acceptable Value	Default Value
LoAlarmValue	Low alarm point assigned to tag Real	0
LogDeadband	Logging deadband assigned to the tag Real	0
Logged	Tag value logging enabled or disabled No or Off = Logging disabled Yes or On = Logging enabled	No
LoLoAlarmDisable	LoLo alarm disabled or enabled 0 = LoLo alarm enabled 1 = LoLo alarm disabled	0
LoLoAlarmInhibitor	Name of the tag used to inhibit LoLo alarm Any discrete or analog tag	None
LoLoAlarmPri	Priority assigned to LoLo alarm 1 to 999	1
LoLoAlarmState	LoLo alarm enabled or disabled No or Off = Disabled Yes or On = Enabled	No
LoLoAlarmValue	LoLo alarm point assigned to tag Real	0
MajDevAlarmDisable	Major Deviation alarm disabled or enabled 0 = Major Deviation alarm enabled 1 = Major Deviation alarm disabled	0
MajDevAlarmInhibitor	Name of the tag used to inhibit Major Deviation alarm Any discrete or analog tag	None
MajorDevAlarmPri	Priority assigned to Major Deviation alarm 1 to 999	1

Attribute	Acceptable Value	Default Value
MajorDevAlarmState	Major deviation alarm enabled or disabled No or Off = Disabled Yes or On = Enabled	No
MajorDevAlarmValue	Major deviation alarm percentage assigned to tag Real	0
MaxEU	Maximum engineering units value assigned to the tag Real	32767
MaxLength	Maximum message length Real	131
MaxRaw	Maximum raw value assigned to tag Real	32767
MaxValue	Maximum value assigned to the tag Real	32767
MinDevAlarmDisable	Minor Deviation alarm disabled or enabled 0 = Minor Deviation alarm enabled 1 = Minor Deviation alarm disabled	0
MinDevAlarmInhibitor	Name of the tag used to inhibit Minor Deviation alarm Any discrete or analog tag	None
MinEU	Minimum engineering units value assigned to the tag Real	-32768
MinorDevAlarmPri	Priority assigned to Minor Deviation alarm 1 to 999	1
MinorDevAlarmState	Minor deviation alarm enabled or disabled No or Off = Disabled Yes or On = Enabled	No

Attribute	Acceptable Value	Default Value
MinorDevAlarmValue	Minor deviation alarm percentage assigned to tag Real	0
MinRaw	Minimum raw value assigned to tag Real	-32768
MinValue	Minimum value assigned to the tag Real	-32768
OffMsg	Discrete tag Off message Text string	None
OnMsg	Discrete tag On message Text string	None
ReadOnly	Tag value read only or read/write Yes = Read Only No = Read/Write	No
RetentiveAlarmParameters	Tag Retentive Parameters enabled or disabled No or Off = Disabled Yes or On = Enabled	No
RetentiveValue	Tag Retentive Value enabled or disabled 0, Off, False, or No = Disabled 1, On, True, or Yes = Enabled	No
RocAlarmDisable	Rate of Change alarm disabled or enabled 0 = ROC alarm enabled 1 = ROC alarm disabled	0
RocAlarmInhibitor	Name of the tag used to inhibit Rate of Change alarm Any discrete or analog tag	None
ROCAlarmPri	Priority assigned to Rate of Change alarm 1 to 999	1

Attribute	Acceptable Value	Default Value
ROCAlarmState	Rate of Change alarm enabled or disabled No or Off = Disabled Yes or On = Enabled	No
ROCAlarmValue	Change in tag value by percent Real	0
ROCTimeBase	Measurement period to calculate rate of change Sec, Min or Hr	Min
SymbolicName	Symbolic name assigned to input data blocks by the S7 Tag Creator product. Symbolic names are listed in the S7 Tag Creator Symbol Table.	None

:MemoryDisc keyword attributes

The DBLoad import file includes the :MemoryDisc keyword to define memory discrete tags that can be imported to the Tagname Dictionary. The following table lists the attributes of the :MemoryDisc keyword associated with the properties of a memory discrete tag.

The table shows the order that :MemoryDisc keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	Logged
4	EventLogged
5	EventLoggingPriority
6	RetentiveValue
7	InitialDisc
8	OffMsg
9	OnMsg

String Position	Attribute
10	AlarmState
11	AlarmPri
12	AlarmComment
13	AlarmAckModel
14	DSCAlarmDisable
15	DSCAlarmInhibitor
16	SymbolicName

:IODisc keyword attributes

The DBLoad import file includes the :IODisc keyword to define I/O discrete tags that can be imported to the Tagname Dictionary. The following table lists the attributes of the :IODisc keyword associated with the properties of an I/O discrete tag.

The table shows the order that :IODisc keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	Logged
4	EventLogged
5	EventLoggingPriority
6	RetentiveValue
7	InitialDisc
8	OffMsg
9	OnMsg
10	AlarmState
11	AlarmPri
12	Conversion
13	AccessName

String Position	Attribute
14	ItemUseTagname
15	ItemName
16	ReadOnly
17	AlarmComment
18	AlarmAckModel
19	DSCAlarmDisable
20	DSCAlarmInhibitor
21	SymbolicName

:MemoryInt keyword attributes

The DBLoad import file includes the :MemoryInt keyword to define memory integer tags that can be imported to the Tagname Dictionary. The following table lists the attributes of the :MemoryInt keyword associated with the properties of a memory integer tag.

The table shows the order that :MemoryInt keyword attributes are specified when the DBDump utility is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	Logged
4	EventLogged
5	EventLoggingPriority
6	RetentiveValue
7	RetentiveAlarmParameters
8	AlarmValueDeadband
9	AlarmDevDeadband
10	EngUnits
11	InitialValue

String Position	Attribute
12	MinValue
13	MaxValue
14	Deadband
15	LogDeadband
16	LoLoAlarmState
17	LoLoAlarmValue
18	LoLoAlarmPri
19	LoAlarmState
20	LoAlarmValue
21	LoAlarmPri
22	HiAlarmState
23	HiAlarmValue
24	HiAlarmPri
25	HiHiAlarmState
26	HiHiAlarmValue
27	HiHIAlarmPri
28	MinorDevAlarmState
29	MinorDevAlarmValue
30	MinorDevAlarmPri
31	MajorDevAlarmState
32	MajorDevAlarmValue
33	MajorDevAlarmPri
34	DevTarget
35	ROCAlarmState
36	ROCAlarmValue
37	ROCAlarmPri

String Position	Attribute
38	ROCTimeBase
39	AlarmComment
40	AlarmAckModel
41	LoLoAlarmDisable
42	LoAlarmDisable
43	HiAlarmDisable
44	HiHiAlarmDisable
45	MinDevAlarmDisable
46	MajDevAlarmDisable
47	RocAlarmDisable
48	LoLoAlarmInhibitor
49	LoAlarmInhibitor
50	HiAlarmInhibitor
51	HiHiAlarmInhibitor
52	MinDevAlarmInhibitor
53	MajDevAlarmInhibitor
54	RocAlarmInhibitor
55	SymbolicName

:IOInt keyword attributes

The DBLoad import file includes the :IOInt keyword to define I/O integer tags that can be imported to the Tagname Dictionary. The following table lists the attributes of the :IOInt keyword associated with the properties of an I/O integer tag.

The table shows the order that :IOInt keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment

String Position	Attribute
3	Logged
4	EventLogged
5	EventLoggingPriority
6	RetentiveValue
7	RetentiveAlarmParameters
8	AlarmValueDeadband
9	AlarmDevDeadband
10	EngUnits
11	InitialValue
12	MinEU
13	MaxEU
14	Deadband
15	LogDeadband
16	LoLoAlarmState
17	LoLoAlarmValue
18	LoLoAlarmPri
19	LoAlarmState
20	LoAlarmValue
21	LoAlarmPri
22	HiAlarmState
23	HiAlarmValue
24	HiAlarmPri
25	HiHiAlarmState
26	HiHiAlarmValue
27	HiHIAlarmPri
28	MinorDevAlarmState

String Position	Attribute
29	MinorDevAlarmValue
30	MinorDevAlarmPri
31	MajorDevAlarmState
32	MajorDevAlarmValue
33	MajorDevAlarmPri
34	DevTarget
35	ROCAlarmState
36	ROCAlarmValue
37	ROCAlarmPri
38	ROCTimeBase
39	AlarmComment
39	MinRaw
40	MaxRaw
41	Conversion
42	AccessName
43	ItemUseTagname
44	ItemName
45	ReadOnly
46	AlarmComment
47	AlarmAckModel
48	LoLoAlarmDisable
49	LoAlarmDisable
50	HiAlarmDisable
51	HiHiAlarmDisable
52	MinDevAlarmDisable
53	MajDevAlarmDisable

String Position	Attribute
54	RocAlarmDisable
55	LoLoAlarmInhibitor
56	LoAlarmInhibitor
57	HiAlarmInhibitor
58	HiHiAlarmInhibitor
59	MinDevAlarmInhibitor
60	MajDevAlarmInhibitor
61	RocAlarmInhibitor
62	SymbolicName

:MemoryReal keyword attributes

The DBLoad import file includes the :MemoryReal keyword to define memory real tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :MemoryReal keyword associated with the properties of a memory real tag.

The table shows the order that :MemoryReal keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	Logged
4	EventLogged
5	EventLoggingPriority
6	RetentiveValue
7	RetentiveAlarmParameters
8	AlarmValueDeadband
9	AlarmDevDeadband
10	EngUnits
11	InitialValue

String Position	Attribute
12	MinValue
13	MaxValue
14	Deadband
15	LogDeadband
16	LoLoAlarmState
17	LoLoAlarmValue
18	LoLoAlarmPri
19	LoAlarmState
20	LoAlarmValue
21	LoAlarmPri
22	HiAlarmState
23	HiAlarmValue
24	HiAlarmPri
25	HiHiAlarmState
26	HiHiAlarmValue
27	HiHiAlarmPri
28	MinorDevAlarmState
29	MinorDevAlarmValue
30	MinorDevAlarmPri
31	MajorDevAlarmState
32	MajorDevAlarmValue
33	MajorDevAlarmPri
34	DevTarget
35	ROCAlarmState
36	ROCAlarmValue
37	ROCAlarmPri

String Position	Attribute
38	ROCTimeBase
39	AlarmComment
40	AlarmAckModel
41	LoLoAlarmDisable
42	LoAlarmDisable
43	HiAlarmDisable
44	HiHiAlarmDisable
45	MinDevAlarmDisable
46	MajDevAlarmDisable
47	RocAlarmDisable
48	LoLoAlarmInhibitor
49	LoAlarmInhibitor
50	HiAlarmInhibitor
51	HiHiAlarmInhibitor
52	MinDevAlarmInhibitor
53	MajDevAlarmInhibitor
54	RocAlarmInhibitor
55	SymbolicName

:IOReal keyword attributes

The DBLoad import file includes the :IOReal keyword to define I/O real tags that can be imported to the Tagname Dictionary. The following table lists the attributes of the :IOReal keyword associated with the properties of an I/O real tag.

The table shows the order that :IOReal keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment

String Position	Attribute
3	Logged
4	EventLogged
5	EventLoggingPriority
6	RetentiveValue
7	RetentiveAlarmParameters
8	AlarmValueDeadband
9	AlarmDevDeadband
10	EngUnits
11	InitialValue
12	MinEU
13	MaxEU
14	Deadband
15	LogDeadband
16	LoLoAlarmState
17	LoLoAlarmValue
18	LoLoAlarmPri
19	LoAlarmState
20	LoAlarmValue
21	LoAlarmPri
22	HiAlarmState
23	HiAlarmValue
24	HiAlarmPri
25	HiHiAlarmState
26	HiHiAlarmValue
27	HiHiAlarmPri
28	MinorDevAlarmState

String Position	Attribute
29	MinorDevAlarmValue
30	MinorDevAlarmPri
31	MajorDevAlarmState
32	MajorDevAlarmValue
33	MajorDevAlarmPri
34	DevTarget
35	ROCAlarmState
36	ROCAlarmValue
37	ROCAlarmPri
38	ROCTimeBase
39	MinRaw
40	MaxRaw
41	Conversion
42	AccessName
43	ItemUseTagname
44	ItemName
45	ReadOnly
46	AlarmComment
47	AlarmAckModel
48	LoLoAlarmDisable
49	LoAlarmDisable
50	HiAlarmDisable
51	HiHiAlarmDisable
52	MinDevAlarmDisable
53	MajDevAlarmDisable
54	RocAlarmDisable

String Position	Attribute
55	LoLoAlarmInhibitor
56	LoAlarmInhibitor
57	HiAlarmInhibitor
58	HiHiAlarmInhibitor
59	MinDevAlarmInhibitor
60	MajDevAlarmInhibitor
61	RocAlarmInhibitor
62	SymbolicName

:MemoryMsg keyword attributes

The DBLoad import file includes the :MemoryMsg keyword to define memory message tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :MemoryMsg keyword associated with the properties of a memory message tag.

The table shows the order that :MemoryMsg keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	Logged
4	EventLogged
5	EventLoggingPriority
6	RetentiveValue
7	MaxLength
8	InitialMessage
9	AlarmComment
10	SymbolicName

:IOMsg keyword attributes

The DBLoad import file includes the :IOMsg keyword to define I/O message tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :IOMsg keyword associated with the properties of an I/O message tag.

The table shows the order that :IOMsg keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	Logged
4	EventLogged
5	EventLoggingPriority
6	RetentiveValue
7	MaxLength
8	InitialMessage
9	AccessName
10	ItemUseTagname
11	ItemName
12	ReadOnly
13	AlarmComment
14	SymbolicName

:GroupVar keyword attributes

The DBLoad import file includes the :GroupVar keyword to define Group Variable tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :GroupVar keyword associated with the properties of a Group Variable tag.

Note: InTouch Group Var tags are obsolete. The :GroupVar keyword is included to support legacy applications only.

The table shows the order that :GroupVar keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	SymbolicName

:HistoryTrend keyword attributes

The DBLoad import file includes the :HistoryTrend keyword to define HistTrend tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :HistoryTrend keyword associated with the properties of a HistTrend tag.

The table shows the order that :HistoryTrend keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	SymbolicName

:TagID keyword attributes

The DBLoad import file includes the :TagID keyword to define Tag ID tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :TagID keyword associated with the properties of a Tag ID tag.

The table shows the order that :TagID keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment

:IndirectDisc keyword attributes

The DBLoad import file includes the :IndirectDisc keyword to define indirect discrete tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :IndirectDisc keyword associated with the properties of an indirect discrete tag.

The table shows the order that :IndirectDisc keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	EventLogging
4	EventLoggingPriority
5	RetentiveValue
6	SymbolicName

:IndirectAnalog keyword attributes

The DBLoad import file includes the :IndirectAnalog keyword to define indirect analog tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :IndirectAnalog keyword associated with the properties of an indirect analog tag.

The table shows the order that :IndirectAnalog keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	EventLogging
4	EventLoggingPriority
5	RetentiveValue
6	SymbolicName

:IndirectMsg keyword attributes

The DBLoad import file includes the :IndirectMsg keyword to define indirect message tags that will be imported to the Tagname Dictionary. The following table lists the attributes of the :IndirectMsg keyword associated with the properties of an indirect message tag.

The table shows the order that :IndirectMsg keyword attributes are specified when DBDump is used to create the import file. See Tag keyword attributes for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment

String Position	Attribute
3	EventLogging
4	EventLoggingPriority
5	RetentiveValue
6	SymbolicName

Use blank strings in an import file

For a dictionary import file, there is a difference between a field containing a blank string and a field without data. Keyword attributes that can be assigned a blank string are:

Comment	Eng Units	OffMsg
InitialMessage	OnMsg	Application
ItemName	Topic	

In the following example, a blank string is indicated by quotation marks (" "):

```
:Comment="HI"
:MemoryDisc,Comment,Group
Tagname1,,$System
Tagname2,"",$System
```

where:

The value of the Comment field for Tagname1 is `Hi`, and the value of the Comment field for Tagname2 is a blank comment.

Microsoft Excel ignores quotation marks that denote a blank string when it saves the file, resulting in the following:

```
:Comment="HI"
:MemoryDisc,Comment,Group
Tagname1,,$System
Tagname2,,$System
```

To ensure that a blank string is used with Excel, type a space in the cell as the attribute value.

Use default values for fields

You can use keywords to set the default values for specific fields of a record. The default values are the original InTouch values for the tag type. For example, a memory discrete tag uses the Group=$System, EventLogging=Off, InitialValue=Off, as default values.

For example:

```
:KEYWORD=value
```

This sets the default value of the referenced field for all subsequent data records. Use this feature to set the default value for fields that should remain unchanged for a number of records. If a field has a default value defined, the default value is used if there is no data in the record for the value.

For example, if you set `:GROUP=Reactor_Site`, then all tags that have a blank entry for the GROUP column are assigned to the Reactor_Site Alarm Group. If the tag has, for example, $System entered for the GROUP, the tag remains assigned to the Alarm Group $System.

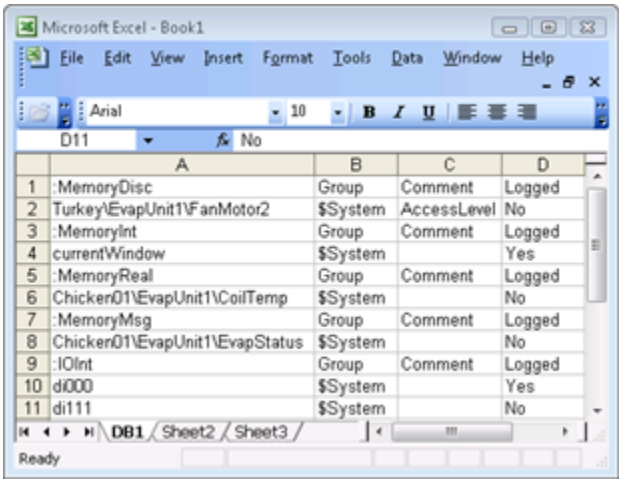
You can reset an individual keyword to its original default value by omitting the value in the equation. For example, :GROUP=.

To reset all keywords, use the :RESET command. This command does not have arguments and affects all entries in the file that occur after the command.

Create SuperTag instances

You can create SuperTags using the DBLoad utility within the Application Manager. However, the SuperTag instances you create are not reflected in the SuperTag template definition.

You must use the valid SuperTag format, and the SuperTag instance data records must begin with the valid keyword for the tag type. For example:



	A	B	C	D
1	:MemoryDisc	Group	Comment	Logged
2	Turkey\EvapUnit1\FanMotor2	$System	AccessLevel	No
3	:MemoryInt	Group	Comment	Logged
4	currentWindow	$System		Yes
5	:MemoryReal	Group	Comment	Logged
6	Chicken01\EvapUnit1\CoilTemp	$System		No
7	:MemoryMsg	Group	Comment	Logged
8	Chicken01\EvapUnit1\EvapStatus	$System		No
9	:IOInt	Group	Comment	Logged
10	di000	$System		Yes
11	di111	$System		No

The following syntax examples are valid:

```
ParentInstance\ChildMember
ParentInstance\ChildMember\Submember
```

The following syntax examples are invalid:

```
ParentInstance\
ParentInstance\ChildMember\
```

If you use an invalid format, an error message appears.

When you import the CSV file containing SuperTag instances, the instances are automatically added to the Tagname Dictionary and are immediately available for use in animation links and InTouch QuickScripts.

Import tag definitions with DBLoad

When you import the contents of a file with DBLoad, all tag definitions are imported into the Tagname Dictionary of the selected InTouch application.

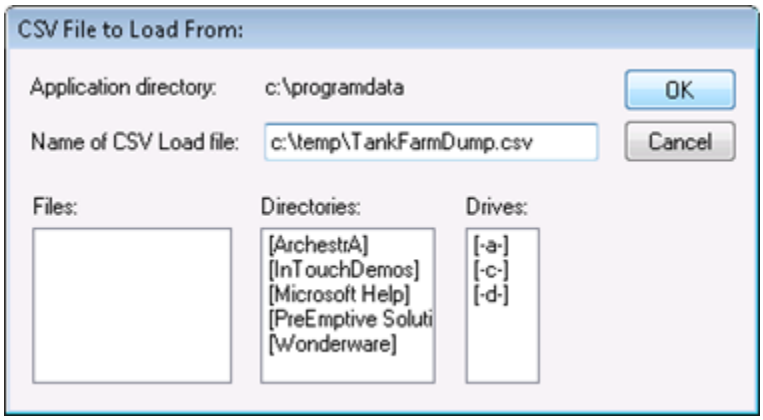
If the import fails, a message appears describing the reason for the failure. The error messages are written to the Logger.

Import tag definitions into an InTouch application

1. Close WindowMaker and WindowViewer.
2. Back up the application whose Tagname Dictionary will be loaded with tag definitions from an import file.
3. Start **Application Manager.**
4. Select the application from the list whose Tagname Dictionary will receive the imported tag definitions.
5. On the **File** menu, in the **Data** group, select **DBLoad**.

   A message appears requesting confirmation that you backed up the InTouch application.
6. Select **Yes** to confirm the application is backed up.

   The **CSV File to Load From** dialog box appears.



7. In the **Name of CSV Load file** box, locate and select the file you want to import.
8. Select **OK**.

   The next step varies based upon whether DBLoad imports new or existing tag definitions to the Tagname Dictionary.

   - If you are importing new tag definitions, the new tag data is loaded into the application's Tagname Dictionary. A message appears confirming the data was successfully loaded and merged.
   - If you are importing existing tag definitions, the import stops if the :mode keyword is set to :mode=ask and the import file contains duplicate tags. You are shown options to handle the duplicate tags or you can cancel the import. For more information about keyword options, see Set the operating mode for dictionary import files.

Known Limitation with Importing Tag Definitions using DBLoad Utility

The DBLoad utility may fail to import files or show performance issues when importing huge tag count newtag.tag files of file size exceeding 2 GB.

Limitations:

- DBLoad may fail to import huge tag count file, if newtags.tag file size exceeds 2GB limit.
- DBLoad will hit performance issues while importing huge tag count CSV file.

Workaround:

Use the following registry entry to avoid importing tags to newtags.tag file, and instead import to the Tagname.x file.

1. Open the Registry Editor.
2. Navigate to **HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Wonderware\InTouch\Installation**.
3. Right-click on the Installation folder in the navigation pane, or anywhere in the main area, select **New**, and select **DWORD value**.
4. Rename the newly created file as `DisableWriteToNewTag`.
5. Set “DisableWriteToNewTag” key to 1.
6. Now, import tags using DBLoad from CSV file, and the import will be successful.

The effect of different values of DisableWriteToNewTag registry key on the import functionality is listed below:

DWORD Value of DisableWriteToNewTag	Effect on Import
DisableWriteToNewTag registry key is present and the DWORD value is set to 1	DBLoad utility will skip importing to newtags.tag file
DisableWriteToNewTag registry key is present and the DWORD value is set to 0	DBLoad utility will import tags to newtags.tag file

Note: For InTouch HMI version 2023 onwards, the registry key DisableWriteToNewTag is not present by default, and the DBLoad Utility skips importing to the newtags.tag file.

Export and import InTouch windows between InTouch applications

You can export windows from all three InTouch application types from WindowMaker, but there are some restrictions on importing exported windows or windows directly from an InTouch application.

- For standalone InTouch applications, you cannot import any windows from published and managed InTouch applications that contain Industrial graphics. A warning message appears, and information on which windows were not imported is written to the Logger.

  If you import windows from managed or published InTouch application that contain Industrial graphics, the windows are imported, but the Industrial graphics are not functional and appear as "Not Found".
- For managed InTouch applications, you can import any windows from published, stand-alone, and other managed InTouch applications. Embedded Industrial graphics are not imported.
- For published InTouch applications, you can import any windows from standalone InTouch applications. Embedded Industrial graphics are not imported.

Import windows

Importing windows from an existing InTouch application into your current application allows you to reduce development time because you can reuse your previously created windows, objects, and window scripts.

You must convert an application to the current version of the InTouch HMI software before you can import windows.

By default, placeholders are created for the tags associated with an imported window. After importing, you can convert the placeholders to local tags or remote tag references. For more information, see Tag placeholders for imported windows and scripts. If the associated tags already exist in the target application, during the import

you can select to use these instead.

When you import windows containing SmartSymbols and select to use existing tags, the InTouch HMI still keeps placeholders for the recovered symbols, even though the tags are available in the target application.

When you import a window from an application that contains SuperTags, only the SuperTag instances actually used in the window are imported into the new application. The entire SuperTag template structure is not imported. For example, if the application has hundreds of SuperTag member tags defined in it, and only 50 of those are used in the imported window, only those 50 are imported.

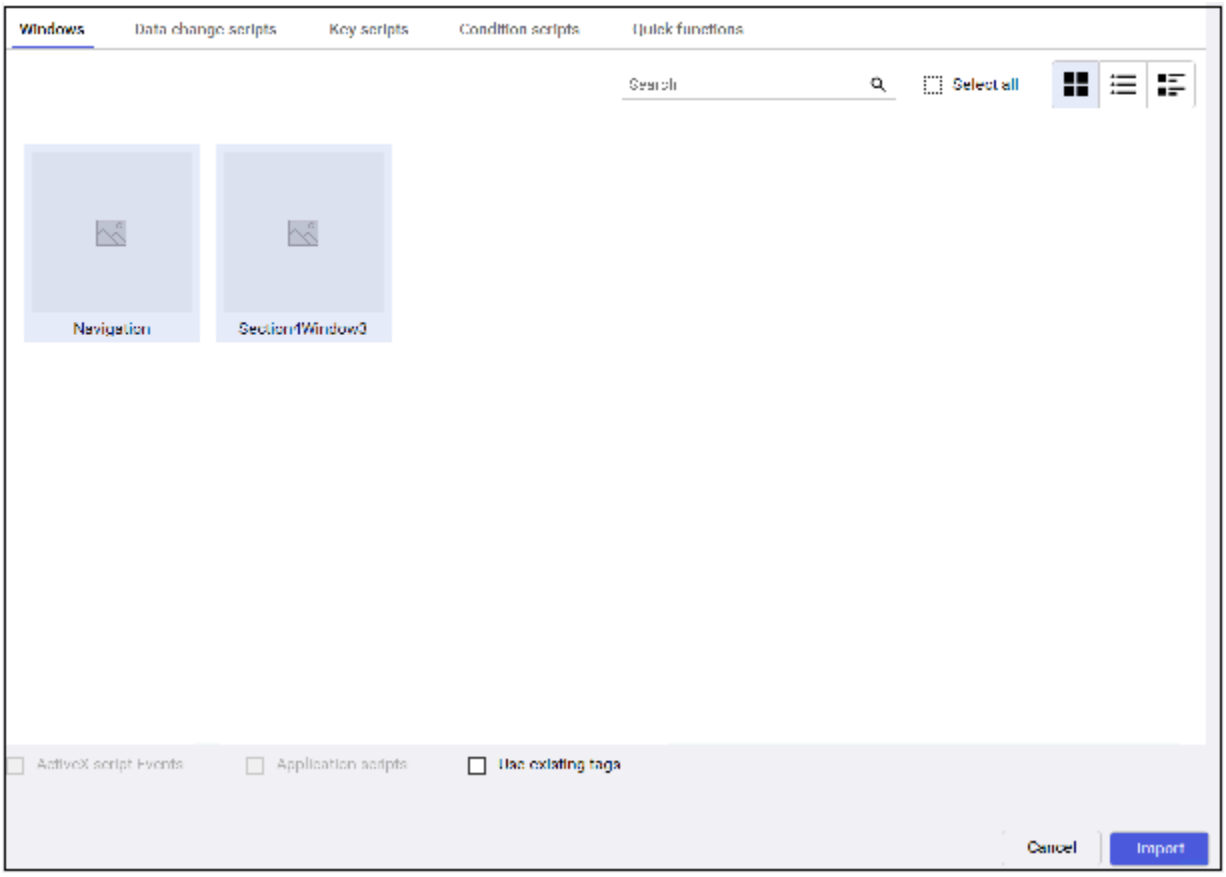
Important: If you move InTouch window files using any method other than importing or exporting them, the contents of the application Tagname Dictionary can become corrupt.

Import a window

1. Close all windows in your current application.
2. On the **File** menu, select **Import**, select **Visualization**, and then select **Windows and Scripts**.

   The **Open Folder** dialog box appears.
3. Select the folder for the application containing the windows to import.
4. Select **OK**.

   The **Application Data Import Options** dialog box appears.



5. In the **Windows** tab to select the individual windows to import.
6. Select the **Use Existing Tags** check box if the tags associated with the imported windows already exist in your application and you want to use them instead of placeholders.
7. Select **Import**.

8. Convert the placeholder tags to either local tags or remote tag references. For more information, see Convert placeholder tags for an imported window.
9. If an imported window contains one or more wizards, double-click on each wizard to open its properties panel. If an imported window contains one or more SmartSymbols, edit each SmartSymbol and create new instances.

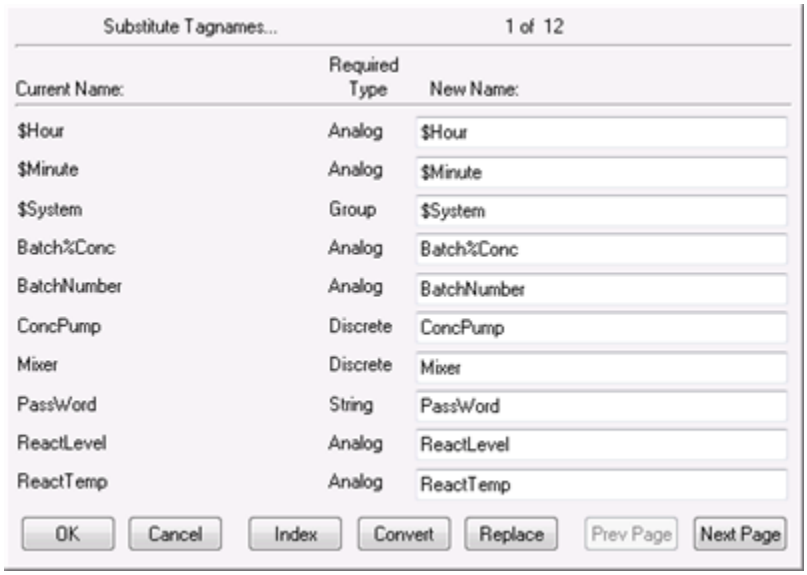
Convert placeholder tags for an imported window

When you import or export a window or QuickScript to or from your current application, all the tags associated with that window or QuickScript are transferred with the window. But, the tags are not added to your new application's Tagname Dictionary. Instead, the tags are automatically marked as placeholder tags unless the **Conserve Placeholders** options is selected on import. You must convert these placeholder tags and, if required, define them in your new application Tagname Dictionary.

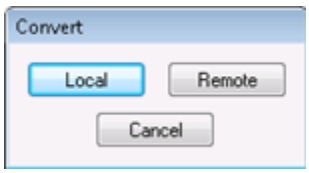
To convert tags for a window

1. Open the window in WindowMaker.
2. Press F2 to select all objects in the window.
3. On the **Animation** menu, in the **Substitute** group, select **Tags**.

   The **Substitute Tagnames** dialog box appears.



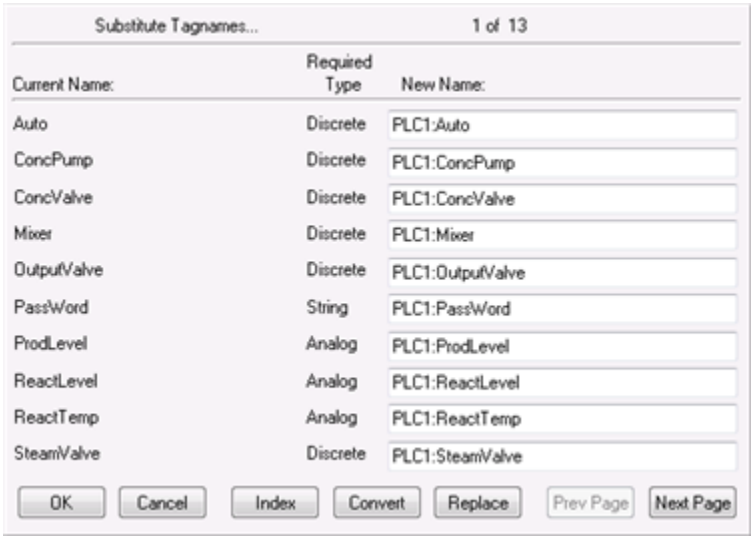
4. Select **Convert**. The **Convert** dialog box appears.



5. Convert the tags.
   - Select **Local** to convert the placeholder tags to local tags. You are prompted to define each tag in the Tagname Dictionary.
   - Select **Remote** to convert the placeholder tags to remote tag references. The **Access Names** dialog box

appears. Select the Access Name and select **Close**.

After the conversion, the **Substitute Tagnames** dialog box shows the new tags.



6. Select **OK**.

Export windows

You can export application windows to

- Create or maintain a library application of all windows.
- Create remote tag references in another application.

You must convert an application to the current version of the InTouch HMI software before you can export windows.

When you export a window, all objects and animation links associated with that window are exported. The tags associated with the objects in the window are converted to placeholder tags to prevent existing tags in the destination application from being overwritten. For more information on converting placeholder tags, see Convert placeholder tags for an imported window.

Important: If you move InTouch window files using any method other than importing or exporting them, the application's Tagname Dictionary can be corrupted.

Export a window

1. Close all windows in your current application.
2. On the **File** menu, select **Export**, select **Visualization**, and then select **Windows**.
3. Select the windows to export, and select **Export**.

   The **Open Folder** dialog box appears.
4. Select the folder of the application to which to export the windows.
5. Select **OK**.
6. If a problem occurs, the **Problem with Export Operation** dialog box appears. Select the option for the action you want to take and then select **OK**.

Import scripts

You can import existing QuickScripts from an InTouch application into your current application to save development time.

You must convert an application to the current version of the InTouch HMI software before you can import scripts.

By default, placeholders are created for the tags associated with an imported QuickScript. After importing, you can convert the placeholders to local tags or remote tag references. For more information, see Tag placeholders for imported windows and scripts. If the associated tags already exist in the target application, during the import you can choose to use these instead.

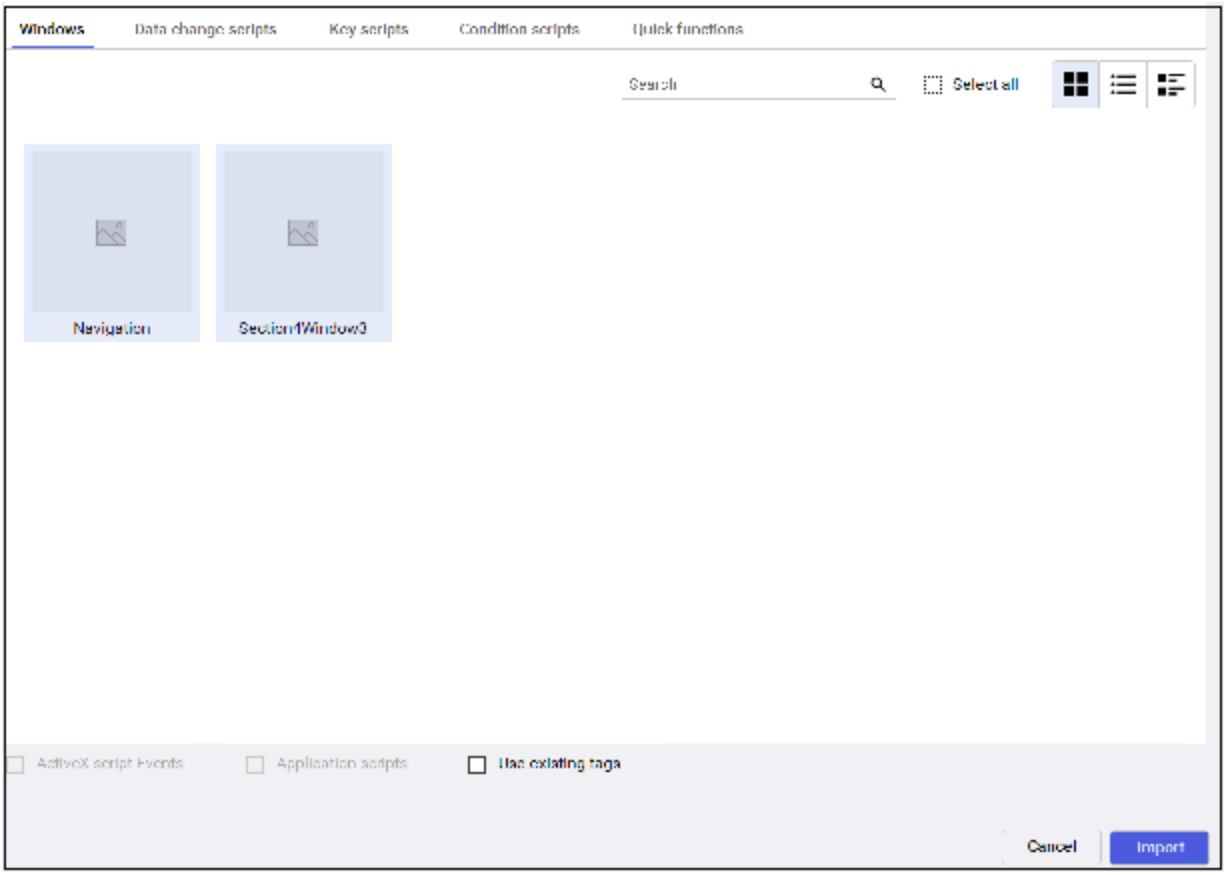
To import a window script, you must import the entire window.

For an imported ActiveX Event script to function properly in the target application, the same ActiveX control and the same event for which the script was originally created must also be used in the target application and it must be loaded into memory. If the window containing an ActiveX control is closed, any scripts associated with it (either ActiveX Event scripts or QuickScripts) do not run properly.

Import a QuickScript

1. Close all windows in your current application.
2. On the **File** menu, select **Import**, select **Visualization**, and then select **Windows and Scripts**.

   The **Open Folder** dialog box appears.
3. Select the folder for the application that contains the scripts to import.
4. Select **OK**. The **Application Data Import Options** dialog box appears.



5. Select the check box for the Qucik Function type(s) that you want to import and then choose **Select** to select the individual script(s) to import.

   Note: To import a window script, you must import the entire window. For more information, see Import windows.

6. Select the **Use Existing Tags** check box if the tags associated with the imported script(s) already exist in your application and you want to use them instead of placeholders.
7. Select **Import**. If your application has scripts with identical names, you are prompted to overwrite, skip, or rename.
8. Convert the placeholder tags to either local tags or remote tag references. For more information, see Convert placeholder tags in an imported script.

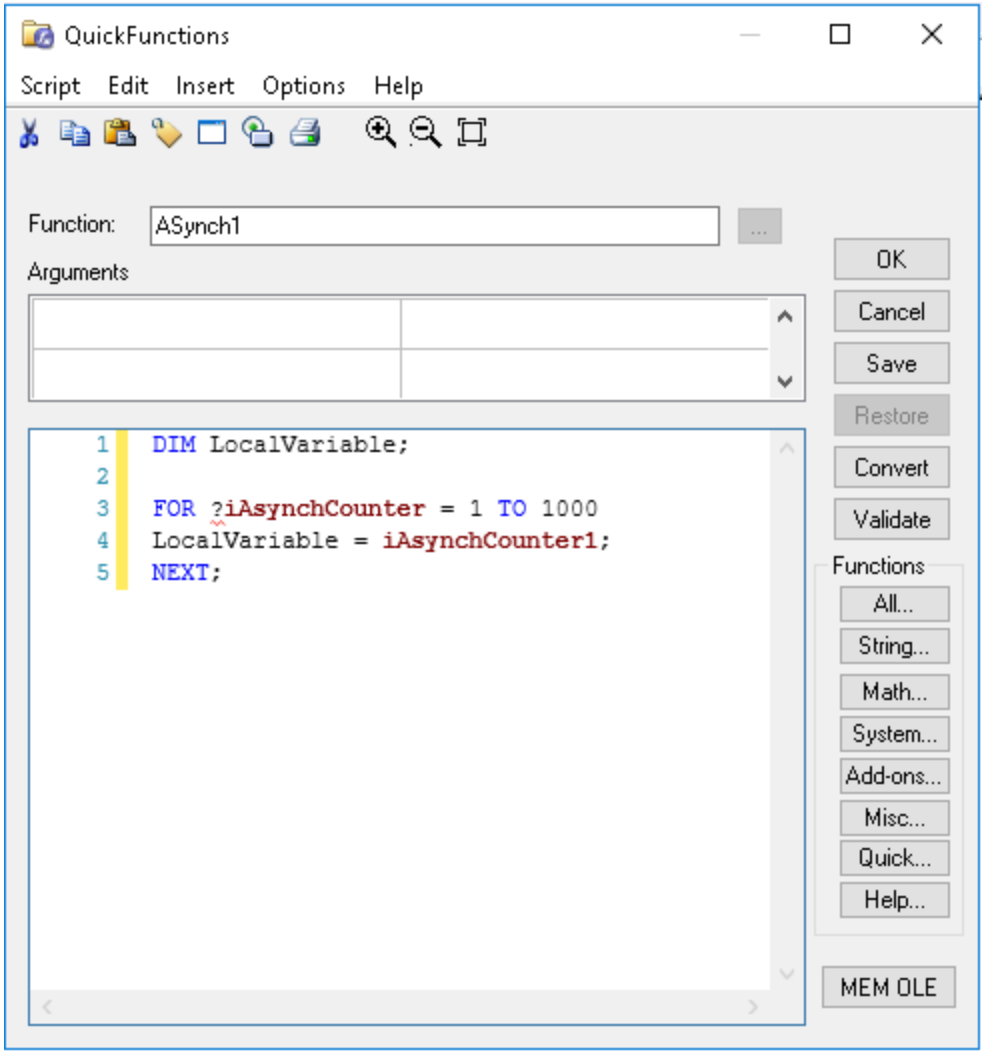
Convert placeholder tags in an imported script

When you import or export a QuickFunction to or from your current application, all the tags associated with that QuickFunction are transferred. But, the tags are not added to your new application's Tagname Dictionary. Instead, they are automatically marked as placeholder tags. You must convert these placeholder tags and, if required, define them in your new application Tagname Dictionary.

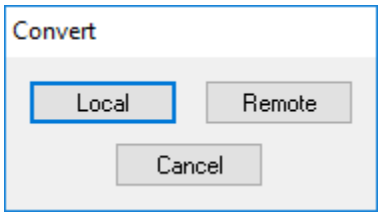
Convert placeholder tags in an imported script

1. On the **Scripts** pane, select the type of QuickFunction you imported.

   The QuickFunction script editor appears, showing the first QuickFunction on file for the selected type of script.



2. Select **Convert**. The **Convert** dialog box appears.



3. Convert the tags.
   - Select **Local** to convert the placeholder tags to local tags. You are prompted to define each tag in the Tagname Dictionary.
   - Select **Remote** to convert the placeholder tags to remote tag references. The **Access Names** dialog box appears. Highlight the Access Name and Select **Close**.
4. After the tags are converted, select **OK** in the QuickScript editor.

Tag placeholders for imported windows and scripts

When you import a window or QuickScript, you can configure how you want the associated tags to be handled.

- **Use placeholder tags.**

  By default, imported tags are converted to "placeholder" (or "index") tags. A maximum of 4096 placeholders is allowed.

  Placeholder tags include a three-character prefix. For example, if the original tag is WaterHeater, then the placeholder tag is ?d:WaterHeater.

  If you import a tag that contains 30, 31, or 32 characters, the placeholder prefix is still added to the beginning of the tag, and the length of the existing tag is not truncated. For example, for placeholder tags only, a 32 character tag is increased to 35 characters. This increase in tag length is not supported for standard tags.

  To use a placeholder tag in the application, you must either:

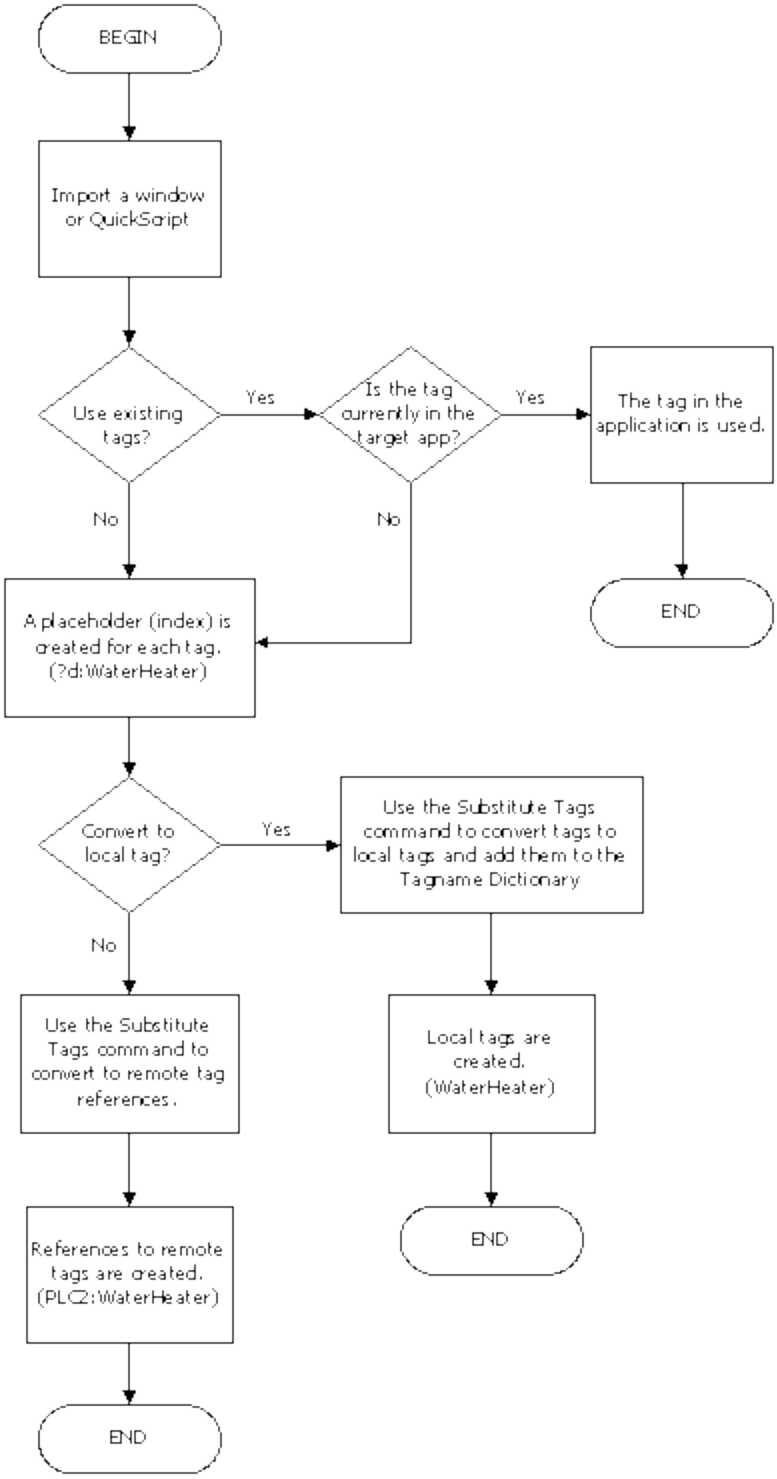
  Convert it to a regular (local) tag and define it in the Tagname Dictionary.

  Convert it to a remote tag reference. An example of a remote tag is PLC2:WaterHeater. Remote tag references allow your application to instantly receive data from a remote tag server and eliminates the need to define a single tag in the local Tagname Dictionary.

- **Using existing tags.**

  During an import, if you select to use existing tags, the InTouch HMI verifies that the imported tags already exist in the Tagname Dictionary. If a tag already exists, then the tag is imported as a fully qualified tag. Using this option reduces the total number of placeholders, allowing you to import applications with larger tag databases.

The following flowchart describes how tags are handled for imported windows and QuickScripts.



Export Industrial Graphics from an application

You can export all Industrial Graphics from an application to an aaPKG file. You can then import the graphics from the file to another application on the same or different computer.

You cannot select Industrial Graphics individually to export from an application. All Industrial Graphics are

exported from an application.

Export Industrial Graphics from an application

1. Open the application in WindowMaker containing the Industrial graphics that you want to export.
2. On the **File** menu, select **Export**, and from the **Visualization** group, select **All Industrial Graphics**.

   The **Export Industrial Graphics** dialog box appears to specify the destination folder and the name of the export file.
3. Select the destination folder to export the aaPKG file.
4. If you want, enter the name of the export file in the **File name** field.

   The default export file name is IndustrialGraphics.aaPKG.
5. Select **Save**.

   A horizontal bar shows the progress of the Industrial Graphics being loaded into the export file.
6. Once the export process is finished, navigate to the destination folder in Windows Explorer, and verify that the export file has been created.

Import Industrial Graphics to an application

You can import Industrial Graphics created in another application to the active application running in WindowMaker.

Only Industrial Graphics from the aaPKG file are imported. The imported graphics overwrite any graphics of the application open for editing in WindowMaker. If the aaPKG file contains non-supported components, the import fails and a dialog box with an error is shown.

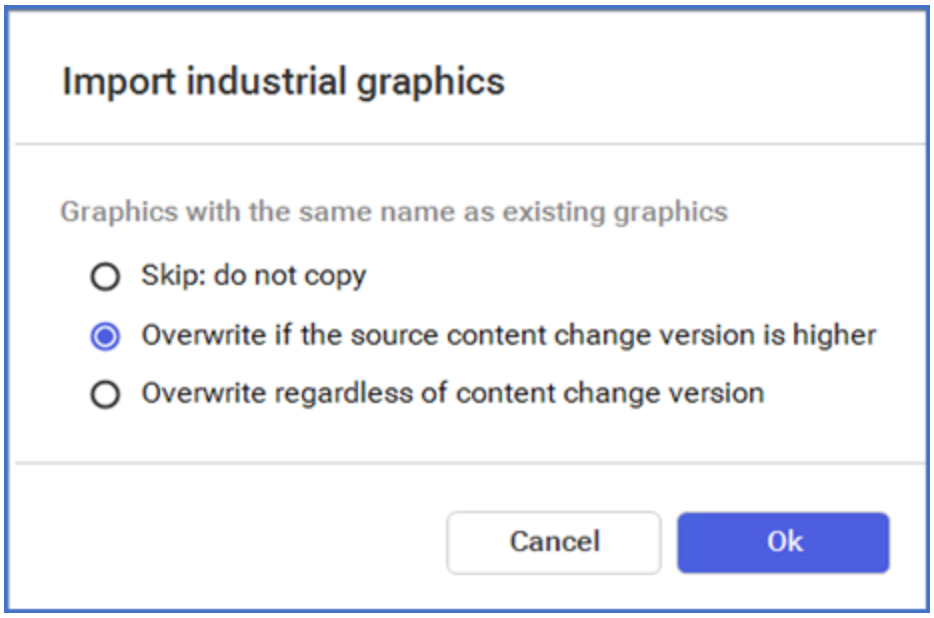
Import Industrial Graphics to an application

1. Open the application in WindowMaker that you want to import Industrial Graphics.
2. On the **File** menu, select **Import**, and from the **Visualization** group, select **Industrial Graphics**.

   The **Import Industrial Graphics** dialog box appears to specify the folder containing an export file of Industrial Graphics.
3. Using Windows Explorer, go to the folder containing an aaPKG file of exported Industrial Graphics.
4. Select the aaPKG file to import.

   The **File name** field shows the name of the file you selected.
5. Select **Open**.

   The Import Industrial Graphics dialog box appears with the following options for overwriting graphics.



- **Skip: Do not copy** - The graphics will not be imported.
- **Overwrite if the source content change version is higher** - Will import the graphics only if the version of the file imported is higher than the installed version.
- **Overwrite regardless of content change version** - The graphics will be imported.

6. Select **OK**.

   A horizontal bar shows the progress of the Industrial Graphics being imported into the active application. When finished, the progress indicator disappears.

Export selected symbols from the Industrial Graphic toolbox

You can export selected Industrial Graphics from the Industrial Graphic Toolbox of an application to an aaPKG file. You can then import these graphics from the file to another application on the same or different computer.

Note: This procedure explains how to export selected Industrial Graphics. See Export Industrial Graphics from an application for instructions to export all Industrial Graphics.

Export selected Industrial Graphics from an Application

1. Open the application in WindowMaker containing the Industrial Graphics that you want to select to export.
2. Select the symbols you want to export in the Industrial Graphic Toolbox.
3. Right-click on a selected symbol to show the shortcut menu.
4. Select **Export** and then **Symbol(s)...** from the shortcut menu.

   The **Export Industrial Graphics** dialog box appears to specify the destination folder and the name of the export file.
5. Select the destination folder to export the aaPKG file.
6. If you want, enter the name of the export file in the **File name** field.

   The default export file name is the name of the first selected symbol from the Industrial Graphic Toolbox.
7. Select **Save**.

A horizontal bar shows the progress of the Industrial Graphics being loaded into the export file.

Import and embed custom client controls

You can create a custom Windows client control and embed it in an Industrial Graphic in your application. First, you must import the client control to WindowMaker's Industrial Graphic Toolbox. This section describes the steps to import and then embed a custom client control in separate procedures.

Import a custom client control

1. Create a custom client control for your application.
2. Place the client control in a folder accessible to the computer where InTouch WindowMaker is installed.
3. On the **File** menu, select **Import**, and from the **Visualization** group, select **Client Control**.

   Important: Only standalone applications can import custom client controls. You cannot import custom client controls to legacy or published InTouch HMI applications.

   The **Import Client Control(s)** dialog box appears with a field to enter the name of a custom client control you created.
4. Using Windows Explorer, go to the folder where you placed the client control .dll file.
5. Select the client control .dll file and select **Open**.

   WindowMaker updates and shows the custom client control you imported in the Industrial Graphic Toolbox.

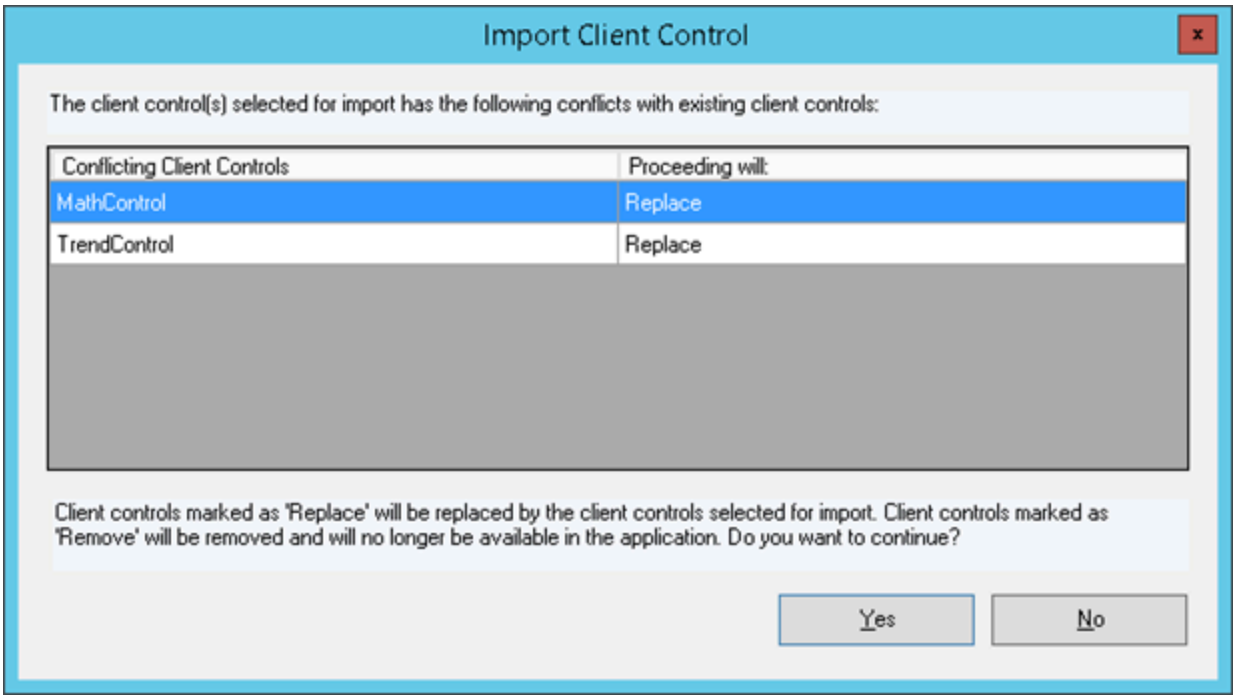
You can also remove an imported client control from the Industrial Graphic Toolbox. First, select the client control within the Industrial Graphic Toolbox. Then, right-click to show the shortcut menu and select **Delete**.

Resolve conflicts when importing duplicate client controls

You can import a different version of a client control and overwrite the existing control. The .dll hosting the existing control will be replaced by the importing library. Conflicting client controls will be detected upon import of the new client control .dll.

Note: Conflict detection is based solely on the name of the control. Library filenames or versions have no affect on conflict detection.

For example, if you import a client control .dll containing the two controls **MathControl** and **TrendControl** and the current library contains controls of the same name, the **Import Client Control** dialog box will display:

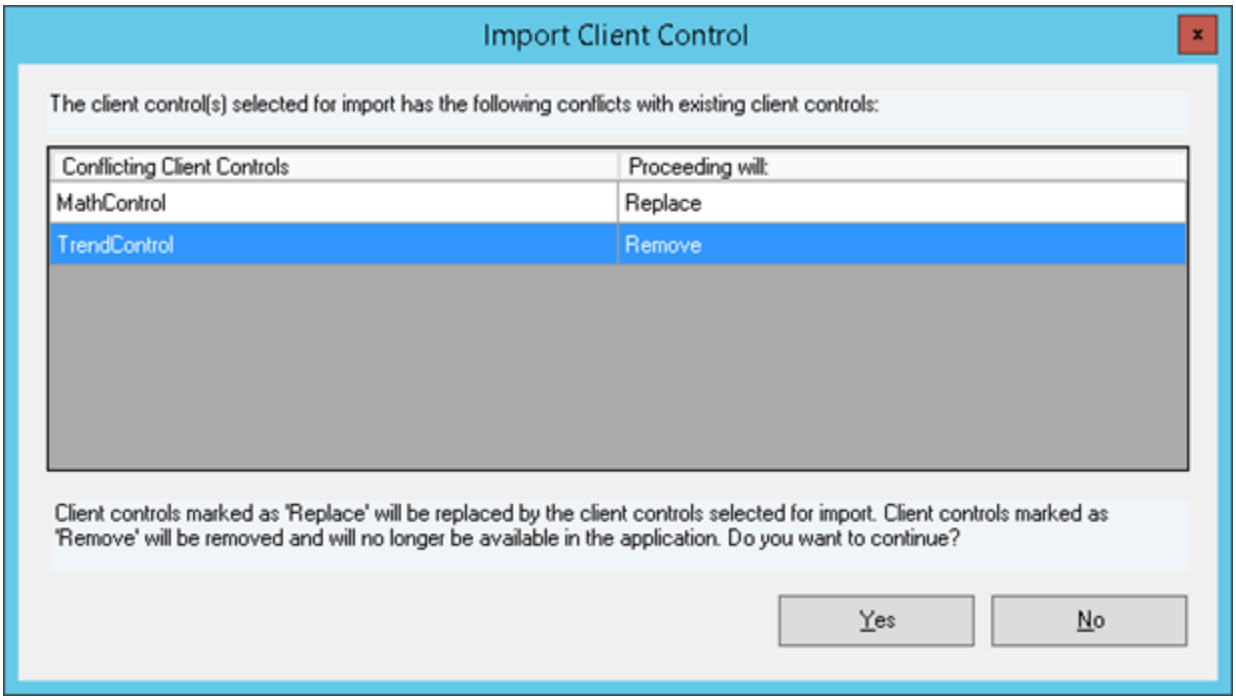


The existing client control .dll will be replaced, and the new control will now be available in the library.

If you see "Remove" in the **Proceeding will** column, it means there are controls in the current library that are not in the importing library. Because the hosting .dll must be replaced to resolve the conflicting controls, any controls that are in the current.dll but not in the importing .dll will be removed upon proceeing with the import.

For example, importing a client control .dll containing the controls MathControl and DatabaseControl and the current library contains MathControl and TrendControl, TrendControl will be removed from the library upon import.

The **Import Client Control** dialog box will prompt you to acknowledge the removal:



The library will be replaced and TrendControl will be removed upon completion of the import.

Restart WindowMaker to update the controls in the Graphic Toolbox.

Note: If you have imported a newer version of client control already embedded in a symbol, restarting WindowMaker and refreshing the graphic thumbnail will not update the contents of the control. You must edit and save the symbol for the new client control to be reflected in the thumbnail.

Embed client controls in Industrial Graphics

Client controls are embedded from the Industrial Graphic Toolbox. The Graphic Toolbox already contains several client controls. You can embed these existing controls into Industrial Graphics, or you can import custom controls and embed those.

Embed a client control into an Industrial Graphic

1. Open the application in WindowMaker that you intend to embed a custom client control.
2. Open the window containing the Industrial Graphic that you intend to embed a custom client control.
3. Select the Industrial Graphic.
4. From the menu bar, select the **Embed Industrial Graphic** icon.

   Important: You cannot drag and drop the custom client control from the Industrial Graphic Toolbox onto the Industrial Graphic. You must always embed the custom client control.
5. Configure your custom client control as needed for the application.

Import HTML5 widgets

Widgets are small web components that can extend the functionality of a webpage or website. Custom-built websites can also incorporate widgets, by using open-source code or frameworks to provide certain functionality in whole or in part. A widget is a self-contained code block that slots into a website without changing any of its features. Widgets are most frequently used to provide on-screen user interface elements that ingrate with other platforms and data sources. A widget can be run on any web page on a website, with consistent placement and user interface. For example, social media, weather, RSS or podcast widgets.

By default, the following widgets are available under the Widgets folder in the Graphic Toolbox:

- Carousel
- Web Browser
- QR Code Scanner
- Map_App

You can import a widget for a standalone and managed application. The file format is Custom Widget Package (.cwp), which includes HTML5, CSS, and Javascript files.

Importing HTML Widgets

1. Launch WindowMaker.
2. On the **File** menu, select **Import**, highlight **Visualization**, and then select **HTML5 widget**.

   The **Import HTML5 widget** dialog box appears.

3. Select the folder for the application containing the windows to import.
4. Select **OK**.

   The widget will appear in the toolbox.

After importing the widgets

1. Create a graphic.
2. Edit the graphic and embed the widget.
3. Set the properties under the Widget Properties section. Each widget will have its own set of properties.
4. Insert the widget on a window.

The widget can now be viewed on WindowViewer and any web browser. Depending on the properties set in the design time you can manipulate the widget in runtime. Scripts using Custom Properties under 'Widget Properties' to modify widgets are not supported.

Carousel widget

A carousel widget allows you to cycle through elements—images or slides of text—like a carousel, without any input. This widget can be used to display dashboards, alerts or alarm information on large monitors on the plant floor.

Note: Client Control (Alarm Client Control and Trend Client Control) display in Carousel Widget is not supported in Window Viewer and Web Client.

Properties

In addition to the standard graphics properties, you can also configure properties specific to the widget, under **Widget Properties**.

Name	Description	Default
Autoplay	If the Autoplay property is set to true, the carousel widget will automatically start on load. If it is set to false, the user must select the next item to start the carousel.	True
BackgroundColor	Sets a background color for the widget. Specify the color value in RGB, HTML Code (#FF0000) or valid HTML color name.	White
GraphicNames	A comma separated list of graphics the carousel will display in runtime.	Empty
Interval	The amount of time delay (in milliseconds) between	5000

Name	Description	Default
	automatically cycling an item.	
Keyboard	If the Keyboard property is set to true, the carousel will respond to keyboard inputs.	True
Loop	If the Loop property is set to true, the carousel will cycle through the graphics continuously, else it will stop after a single cycle.	True
Pause	If the Pause property is set to true, the carousel will pause the cycling of the graphics, when it detects the mouse hovering or a touch down event. The graphics will resume cycling when the mouse is moved away.	True

The carousel widget is based on the Bootstrap 4.0 Carousel component, for more information on bootstrap, go here: https://getbootstrap.com/docs/4.0/components/carousel/

Web browser widget

Using the web browser widget, users can display a web site in WindowViewer and the Web Client.

If Web Client is running in HTTPS, then only HTTPS URL page can be loaded. If Web Client is running in HTTP, then HTTP and HTTPS can both be loaded. If the policy of the web site blocks cross domain access, then this widget will not work. The URL need not be in double quotes, but must be a valid URL.

Properties

URL: The address of the website.

Limitations

- If no protocol is specified, by default the https protocol will be used.
- If the Web Client is configured to use the HTTPS protocol, then only the HTTPS URL page will be loaded. If the HTTP URL is used, the web browser widget will display a message "`Mixed Content: The page at 'https://localhost/intouchweb' was loaded over HTTPS, but requested an insecure frame 'http://**********'. this request has been blocked: the content must be served over HTTPS.`"
- The web browser widget will not function, if the web site policy blocks cross domain (cross origin) access. A link will be provided to open the web page in a separate tab.

QR code scanner

The QRCode_Scanner widget connects to a camera to scan for a QR code and returns the resulting string.

Properties

Property Name	Description	Default Value
QRCode	The resulting string of the scanned QR code. The default value is empty.	Empty
AutoStart	If set to true, the camera will start automatically.	True
AutoStop	If set to true, the camera will stop after scanning a QR Code.	True
Start	If set to true, the camera will start.	False
Stop	If set to true, the camera will stop.	False
BackgroundColor	Sets the background color of the widget. Specify the color value in RGB, HTML Code (#FF0000) or valid HTML color name.	Black

Limitation

- The device must have a camera.
- Using the QR Code on a physical machine instead of a virtual machine is recommended.
- Access the web client using the secure URL (https://) when using the web client remotely.

Usage

You can configure a script to read the QR code and display a graphic based on the scanned value.

In RunTime, the QR Code Scanner widget will appear with a floating toolbar with the following buttons - AutoStart, AutoStop and StartStop.

When the widget is loaded, the camera will start automatically if AutoStart is set to True. To leave the camera on, select **AutoStop**.

To manually start the camera, select **StartStop** and scan the QR Code.

The camera will stay on after you scan the QR code, allowing the user to scan additional QR codes. To stop the

camera, select **StartStop**.

The floating toolbar will display the QRCode derived from the QR Code scanned by the camera.

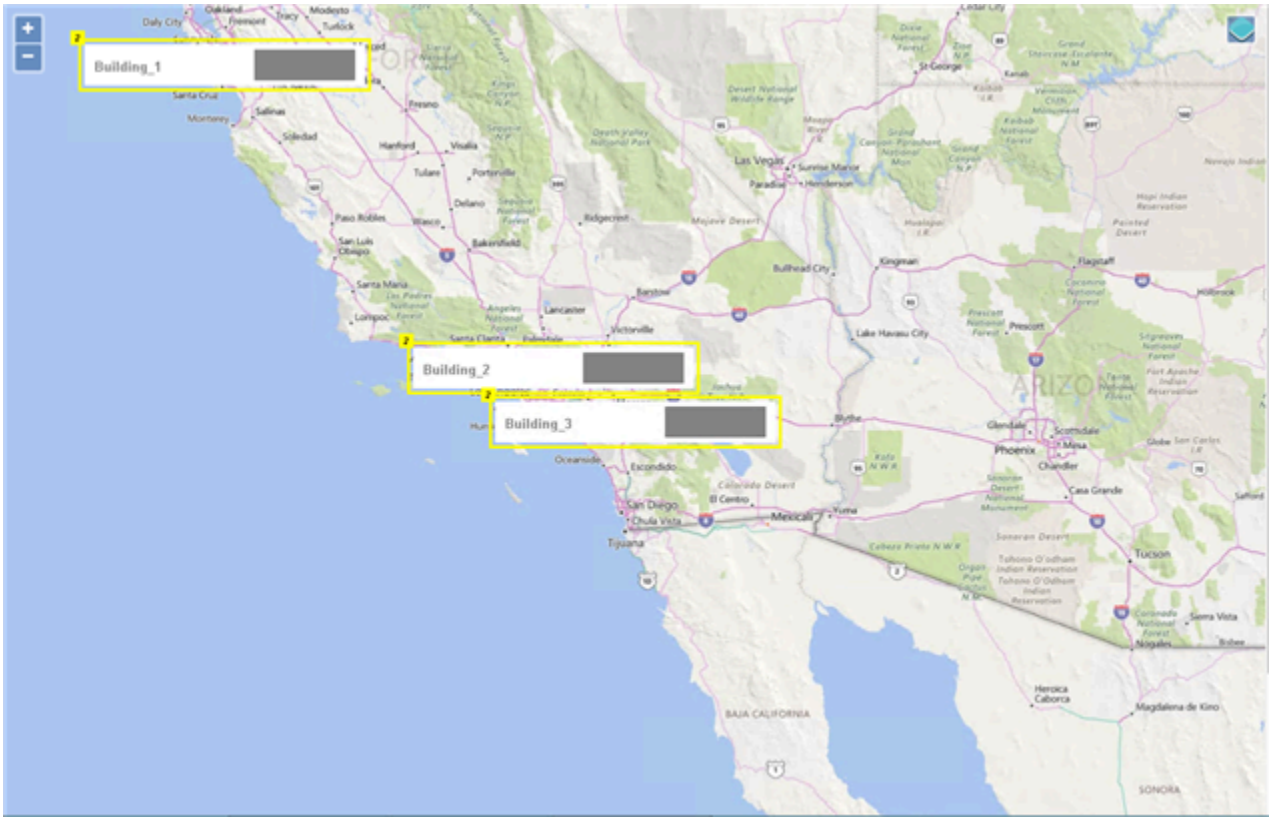
The user can script an action based on the QRCode returned.

Map_App widget

The Map_App widget shows a map containing symbols within a running application. During run time, the map provides controls and touch support to enable users to pan to different areas of the map and zoom in or out to show more or less map detail. Graphics placed in a map typically represent business assets located within an area shown by the map. These graphics can include alarming to show the current state of processes at each business location.

The following example map shows alarm indicators indicating the location of buildings and the current alarm state of each building. During runtime, users can click zoom buttons at the upper left corner of the map to zoom the view in or out. Different map data appears based on how you assigned your company's asset data to the zoom layers. Selecting the icon at the upper right corner of a map lists the maps the user can select to show different data within the current map view.

The MapApp supports touch gestures when a ViewApp runs on a touch-enabled device. Users can pan, zoom, or select items shown on a map using standard touch gestures.



Import script function libraries to an InTouch application

You can import script function libraries to an InTouch Application. Different types of script function libraries can be imported, including .NET (*.dll and other .NET file extensions), script library files (*.aaSLIB), and InTouch script

extension files (*.wdf).

The script function library you imported to one application is automatically included when exporting the application to create another Application. The script function library also is available when publishing the application to which it was imported.

Import a script function library to an application

1. Open the InTouch application to which you want to import a script function library.
2. Select **File** on the WindowMaker main menu, then select **Import**, then select the **Scripts**.

   The **Import Script Function Library** dialog opens.
3. Browse to the function library you want to import.
4. Select the file to import and select **Open** to start importing the script function library.

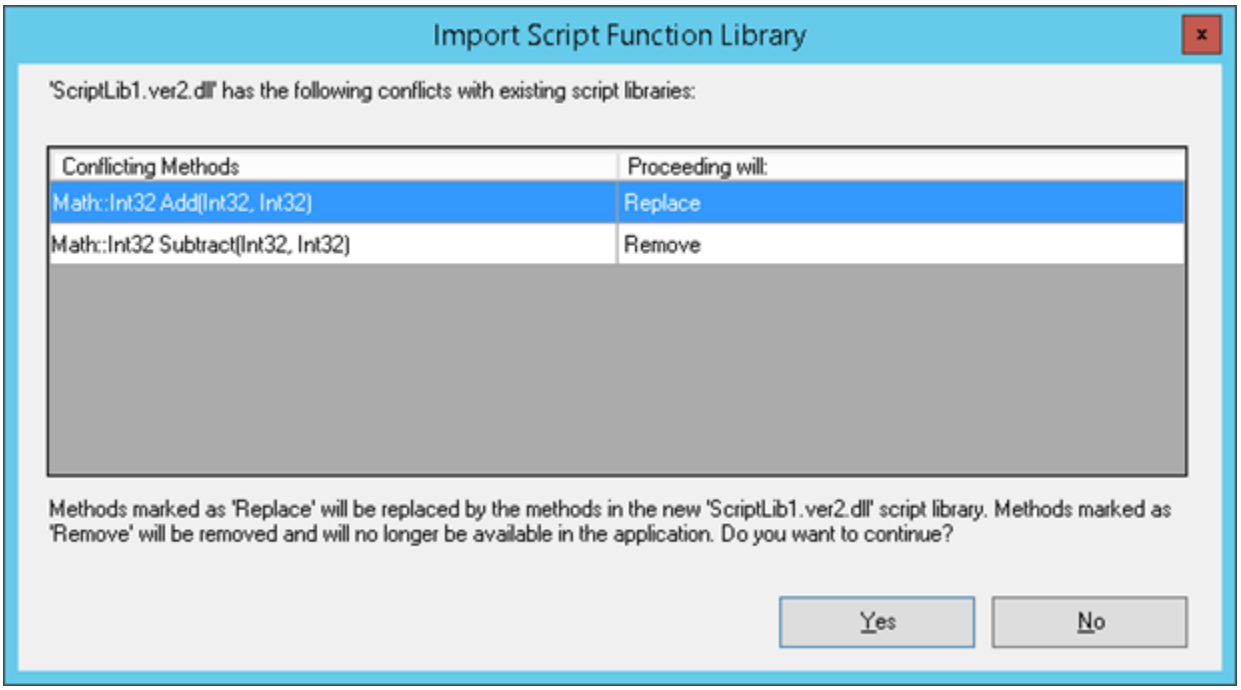
   Note: No progress bar or progress information window appears during the import. An information window opens when the import successfully completes.

Resolve imports of conflicting methods in .NET script libraries

When importing a .NET class script library into an application, the existing script library will be replaced by the importing library. Conflicting script methods will be detected at this time. Conflict detection is based on name space, class name, method name and parameter declaration.

Note: The version or filename or either .dll have no affect on method conflict detection.

Upon import, conflicting methods will be displayed in the **Import Script Function Library** dialog box:



In this example, the `Math::Int32 Add(Int32, Int32)` exists in the current library and contains the same class, method name and parameters as a method in the importing library. It is marked "Replace" in the "Proceeding will" column. Proceeding with the import will replace the entire script library in the application with the importing library.

The `Math::132 Subtract(Int32, Int32)` is marked "Remove" because the importing library does not contain

the `subtract` method. Script method conflict resolution requires replacing the entire script library, which will also result in the removal of this method if it is not in the importing library.

You cannot cancel the import of an individual method that would remove an existing method from the library, as in the example above. You must proceed with all the conflicting methods or cancel the entire import.

Important: Only .NET class library files can be detected as duplicates at time of import. .aaSLIB library and .wdf script extension files will not import if they conflict with methods in the existing library. In this case, no notification of the conflict will be given.

Configure the Application Style Library for applications

You can configure style libraries for graphics in a InTouch application. You can configure application styles for Quality and Status, Element Styles, and numeric Format Styles. Your configuration changes are saved to the application's repository.

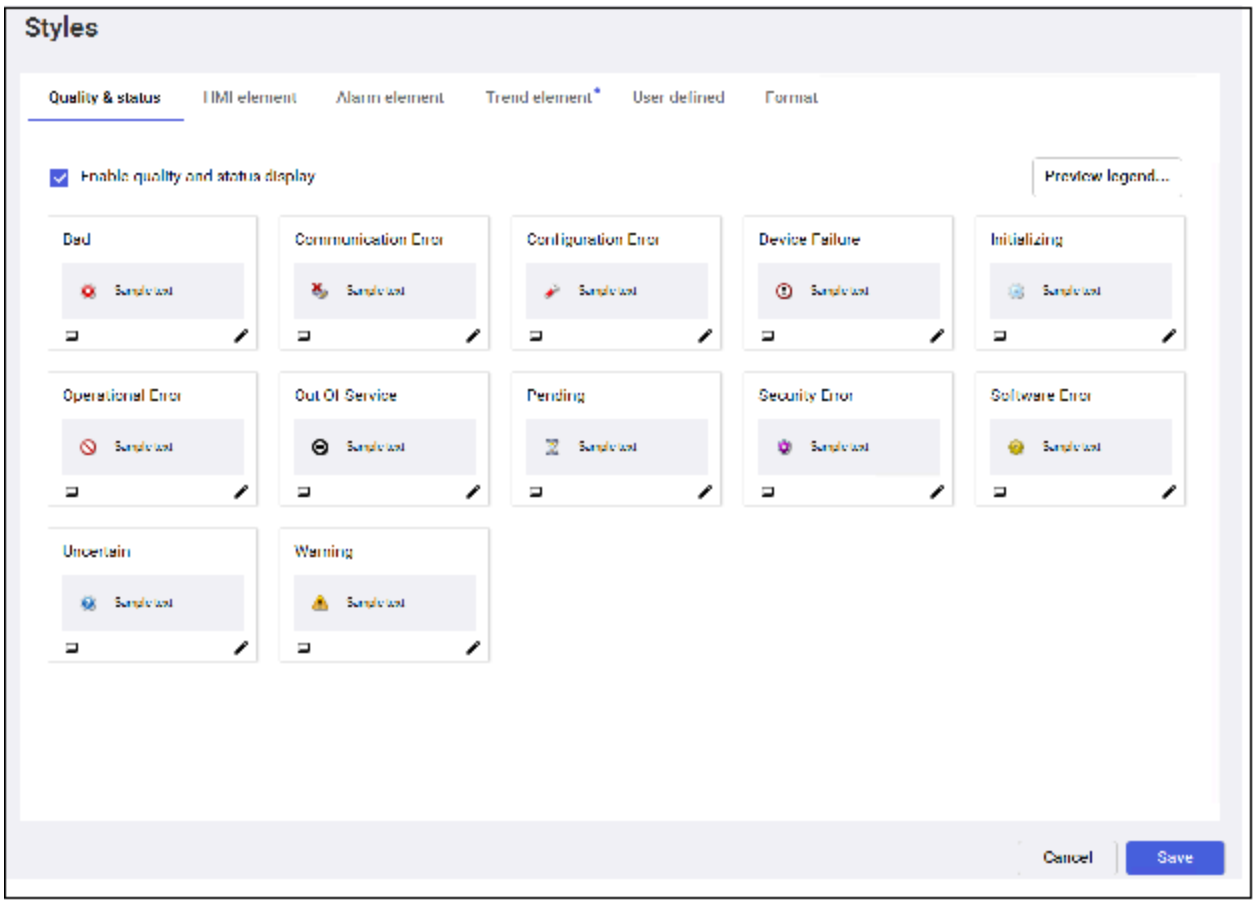
- Quality and Status indicators are graphic icons that represent the current quality of application data and the state of equipment shown by application symbols.
- Element Styles define a set of visual properties that determine the appearance of text, lines, graphic outlines, and interior fill shown in Industrial graphics.
- Format Styles provide options to individually configure application-wide styles for common types of numbers used in industrial graphics.

Important: This section describes the workflow within WindowMaker to access a application's style libraries. For more information about editing application styles, see WindowMaker online help or *Industrial Graphic Editor User* documentation.

Configure the Application Style Library

1. Open an application in WindowMaker.
2. On the **File** menu, select **Configure**, and then select **Styles**.

   The **Styles** configuration screen appears with tabs to configure quality and status indicators, graphic Element styles, and number format styles.



3. Select a tab for the application style you want to edit.

Note: WindowViewer can run only one application at a time. If a platform is deployed on a local node, the configured styles of the Galaxy will take precedence over any configured styles in any other standalone or managed applications.

Export and import the Application Style Library

You can export an Application Style Library from an application and then import it to another application. The settings for quality, Element Styles, and numeric formats are exported to an XML file.

Export an Application Style Library from an application

1. Open WindowMaker.
2. From the **File** menu, select **Export,** and then select **Styles**.

   The **Export Application Style Library** file browser screen appears with fields to specify a file name.
3. Select the folder to place the exported XML file and the name for the file.
4. Select **Save**.

   A dialog box confirms that the Application Style Library was exported successfully.

Import an Application Style Library into an application

1. Open WindowMaker.
2. From the **File** menu, select **Import**, and then select **Styles**.

   The **Import Application Style Library** file browser screen appears with fields to specify a file name.
3. Select the folder where the exported XML file is located and select it to show the name of the export file in the **File Name** field.
4. Select **Open**.

   A dialog box confirms that the Application Style Library was imported successfully.

Configure alarm priority mapping for applications

You can configure the alarm priority mapping of an InTouch application to set a priority range for each alarm severity level.

Important: This section describes the workflow within WindowMaker to map alarm priority ranges to alarm severities. While InTouch does not have built-in Alarm Severity management as does Application Server, users can make use of InTouch tags to implement Alarm Border animation. In this case, the priority to severity mapping in the dialog box is used only as a visual aid to associate priorities to alarm border colors and alarm indicator icons. For more information about configuring alarm priority mapping and alarm shelving, see WindowMaker online help or *Industrial Graphic Editor User documentation*.

Configure alarm priority mappings for applications

1. Open an application in WindowMaker.
2. On the **File** menu, select **Configure**, and then select **Alarms**.

   The **Alarm Priority** section appears with fields to map a priority range to each alarm severity. This screen also contains fields to enable alarm shelving based on alarm severity.

   WM-Configure-AlarmsFinal
3. In the **From Priority** and **To Priority Range** fields, select and enter numbers from 1 to 999 to set the lower and upper boundaries of an alarm priority range for each alarm severity.

   Each priority range should be contiguous without overlap between priority ranges. Alarm severity 1 starts at priority 1 by default.
4. In the **Shelve** column, select or clear the check box to enable alarm shelving for each alarm severity.
5. Select **OK** to save your changes.

   Your changes are saved to the application's application folder.

Export Industrial Graphic text strings from an application

If your application is intended to support run time language switching, you can export the text strings of its Industrial graphics to a dictionary file. You can then translate the strings within the dictionary file to other languages using a text editor, an XML editor, or a spreadsheet program like Microsoft Excel or the Language Assistant.

When you export the graphic text strings, you must specify an output folder for the dictionary file. A best practice is to create a separate folder for each dictionary file whose strings will be translated into another

language.

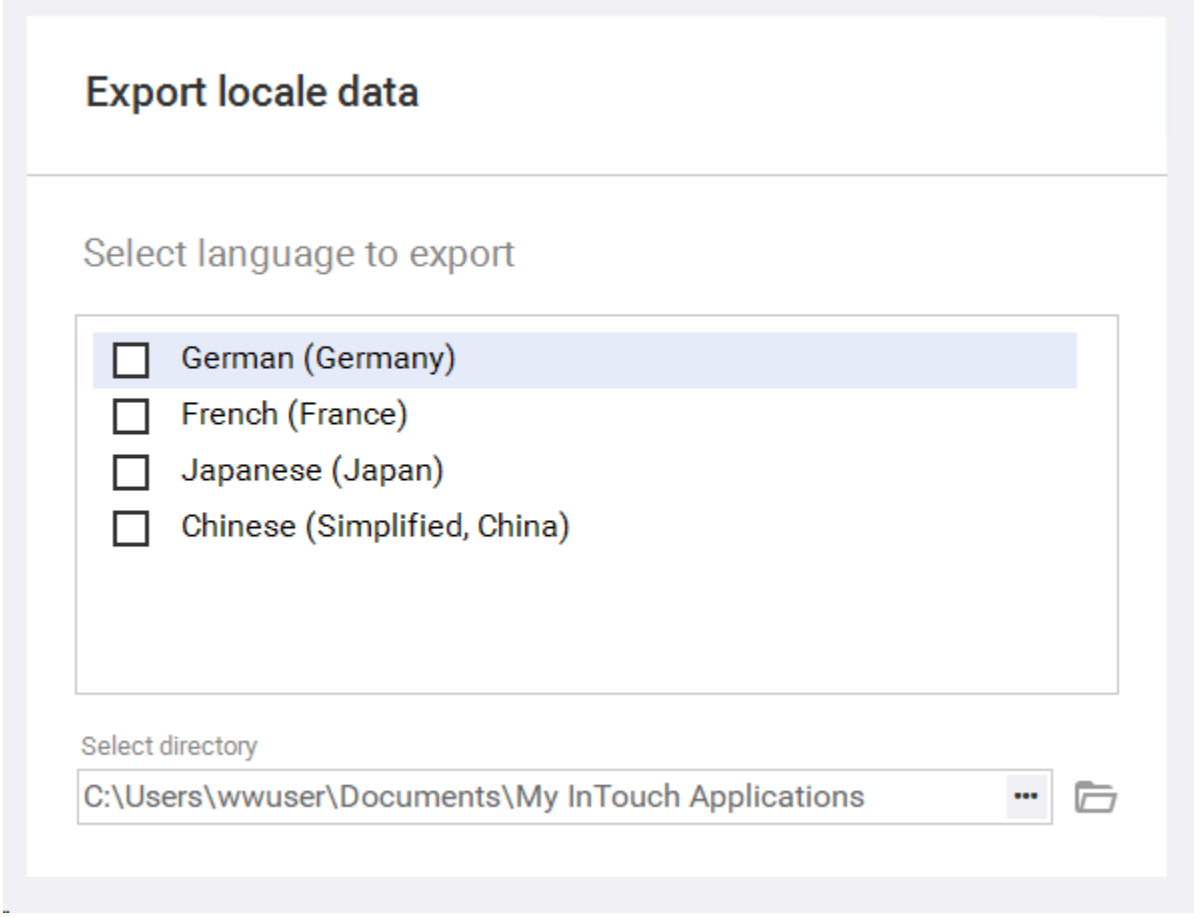
All exported dictionary files follow a naming convention: <AppFolderName>AA_<LanguageID>.xml. For example, if an application folder name is PumpStation and the language being exported is French (Language ID = 1036), then the file name is PumpStationAA_1036.xml.

If you will be exporting language strings for different objects at different times, use separate target folders to prevent subsequent exports from overwriting the first export.

Export Industrial Graphic text strings

1. Open the application in WindowMaker.
2. On the **File** menu, select **Export**, select **Localization**, and then select **Industrial graphic translations**.

   The **Export Locale Data** screen appears.



3. Configure the symbol text strings to export.
   - In the **Languages to export** list, select the check box for the language dictionary to export. The default language is not listed.
   - In the **Select directory** field, type the folder to which you want to export the dictionary file.

     You can also browse to select an existing folder or create a new folder.
4. Select **Export**.

Import text strings of Industrial Graphics to an application

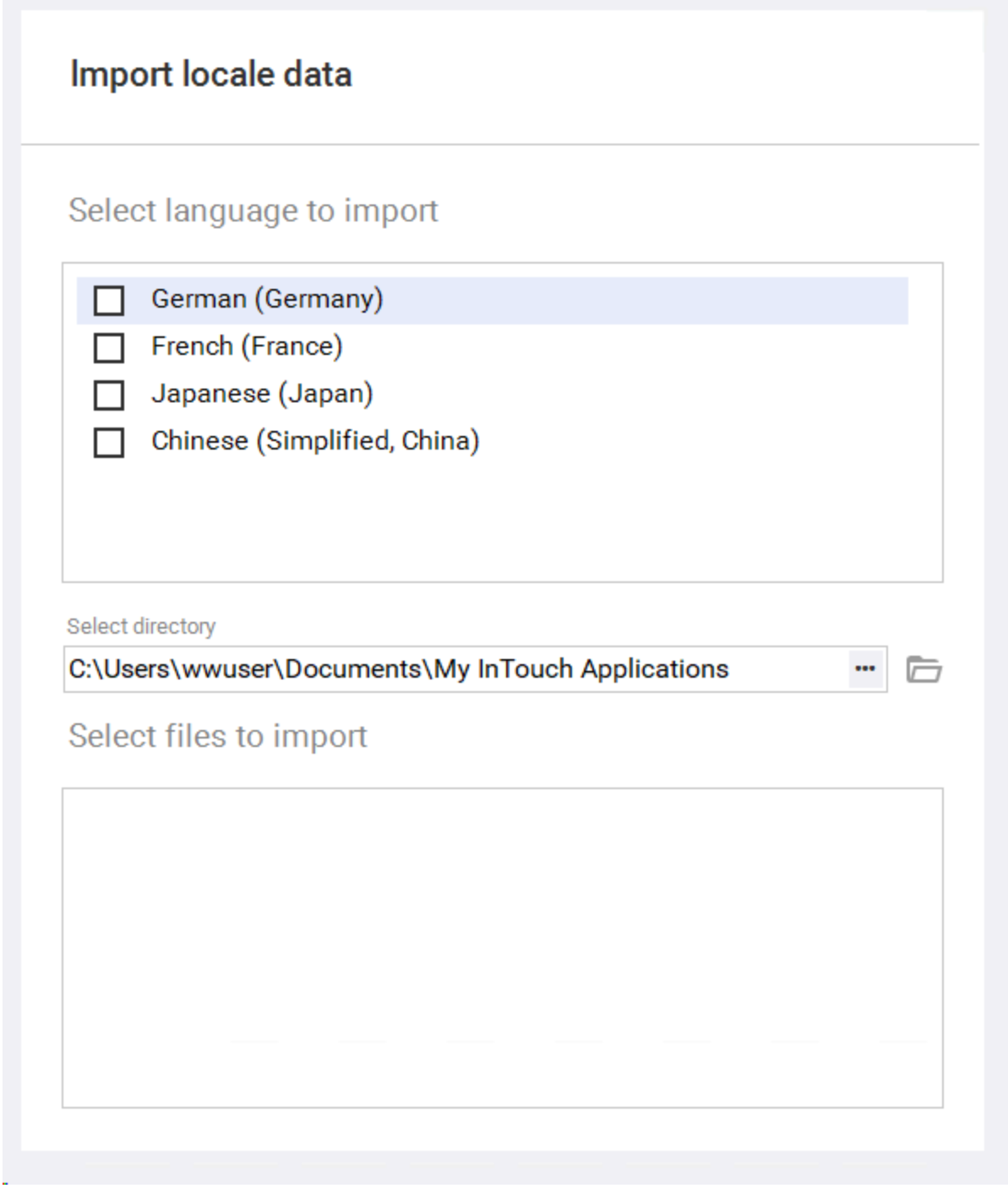
For symbol text, you must import the translated dictionary files for each language to enable run-time language switching for those languages. All dictionary files for a given language should be placed in the same folder.

You can import files for only one language at a time. When you import, you select the desired language and specify the dictionary files to import.

Import a translated dictionary file

1. Open the application in which you want to import Industrial graphic text.
2. On the **File** menu, select **Import**, select **Localization**, and then select **Industrial graphic translations**.

   The **Import Locale Data** screen appears.



3. Configure the import settings.
   - In the **Language to import** list, select the check box for the language dictionary to import.
   - In the **Select directory** box, specify the folder that includes the dictionary file to import.
   - In the **Select files to Import** box, select the .xml files to import. Only files that include the current application folder name and the locale ID for the selected language are shown.
4. Select **Import**.

Export localization strings from a symbol

If your application is intended to support run time language switching, you can export the text strings of one or more symbols selected from the Industrial Graphic Toolbox. You can then translate the exported strings within the file to other languages using a text editor, an XML editor, or a spreadsheet program like Microsoft Excel.

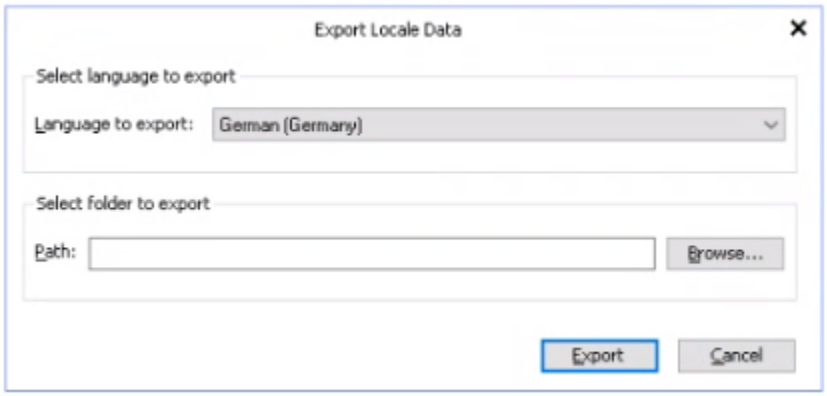
When you export the text strings from a symbol, you must specify an output folder. A best practice is to create a separate folder for each file whose strings will be translated into another language.

All exported localization files follow a naming convention: <AppFolderName>AA_<LanguageID>.xml. For example, if an application folder name is PumpStation1 and the language of the localization strings being exported is Mexican Spanish (Language ID = 2058), then the file name is PumpStation1AA_2058.xml.

Export localization strings from a symbol

1. Open the application in WindowMaker.
2. Select the symbols from the Industrial Graphic Toolbox whose localization strings you want to export.
   - Select a symbol name to select a single symbol.
   - Press the Ctrl key and left-click on symbol names to select two or more symbols.
   - Left-click on a symbol name and then press the Shift key and left-click on another symbol name to select all symbols between the two selected symbols.
3. Right-click on a selected symbol to show the shortcut menu.
4. Select **Export**, then **Localization**, and finally **Selected Symbols(s)**.

   The **Export Locale Data** dialog box appears.



5. Configure the symbol text strings to export.
   - In the **Languages to export** list, select the localization strings to export from the symbols. The default language is not listed.
   - In the **Path** field, type the folder to which you want to export the localization strings. Select **Browse** to select an existing folder or create a new folder.
6. Select **Export**. A bar shows the progress of the export operation.
7. Select **View Details** and verify the localization strings within each selected symbol were exported successfully.

Import the Industrial Graphic Library

During application development you can import the Industrial Graphic Library and Situational Awareness Library into a standalone application, if

- The application was created with a blank template and does not contain the Industrial Graphic Library or Situational Awareness Library
- An older standalone application or modern application was migrated, but the libraries were not imported

To import the Industrial Graphic Library to an application:

- In the Industrial Graphic Toolbox, right-click the application name and select **Import Industrial Graphic Library**.

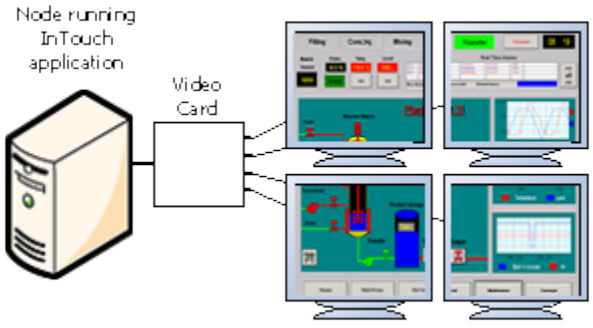
  The Import Industrial Graphics dialog appears. The Industrial Graphics Library is imported first followed by the Situational Awareness Library.

  On completion, the Industrial Graphic Library and the Situational Awareness Library appear in the Industrial Graphic Toolbox.

Set up a multi-monitor system

A multi-monitor system shows an InTouch application on several monitors simultaneously. Together, a multi-monitor configuration creates a composite screen composed of all monitors connected to the computer running an InTouch application. Each monitor can show a portion of the screen or only a single window component like a keypad.

While running an InTouch application, you can move the mouse between monitors and drag windows from one monitor to another. Also, in some multi-monitor configurations you can show an entire InTouch application window across all monitors, as shown in the following figure.



Multi-monitor configurations

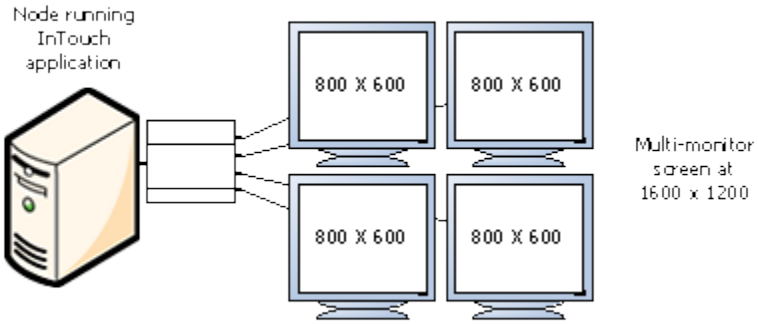
You can use two basic multi-monitor configurations.

- Single video card
- Multiple video card

Each configuration has unique hardware, software, and configuration requirements. Also, each configuration supports a different set of multi-monitor features.

Single video card configuration

In the single video card configuration, the computer has a single video card installed with multiple output ports connected to monitors.



The composite screen resolution is the sum of the individual horizontal and vertical resolution of each monitor. For example, a popular video card connects four 17 inch monitors stacked as a cube: two on the bottom and two on the top. In the previous figure, each monitor runs at a screen resolution of 800 x 600 pixels. The composite virtual screen resolution is 1600 x 1200 pixels.

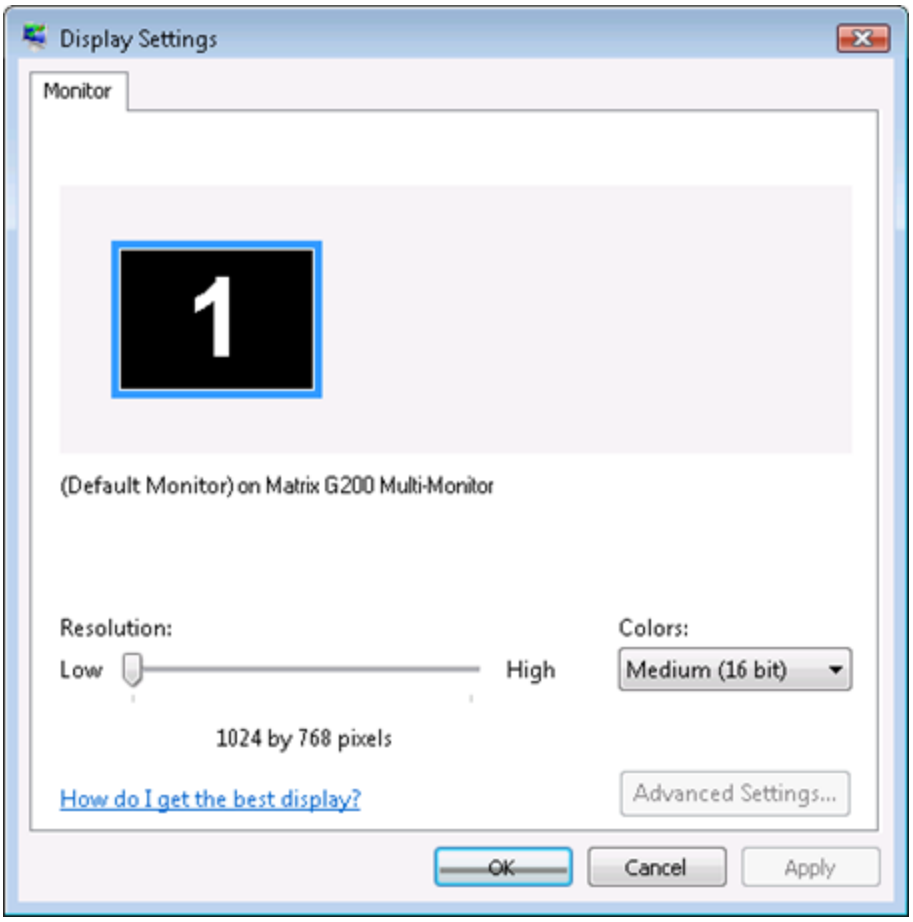
Characteristics of a single card configuration

Single video card drivers have the following characteristics:

- The single video card drives all monitors simultaneously to create a single, large screen.
- The properties of all attached monitors can be configured using a single set of screen values.
- The composite screen shows the Windows taskbar across all of the monitors in the bottom row of the configuration.
- Windows applications can be maximized to fit all monitors.

Characteristics of single card drivers

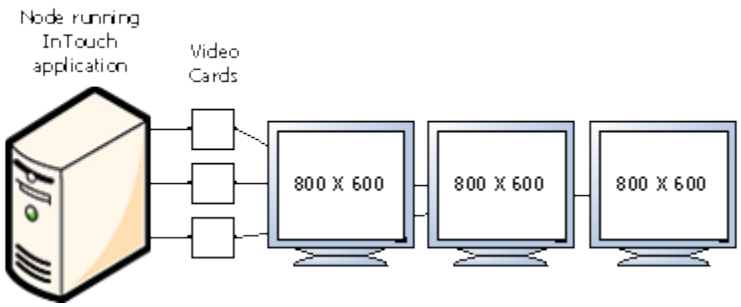
The following figure shows the Windows **Display Properties** dialog box to configure the driver for all monitors connected to a single video card with multiple output ports.



In this figure, the resolution setting is for four monitors arranged side by side in a single row. The resolution for each monitor is 1024 x 768. Added together, the composite screen resolution is 4096 x 768. You only need to configure a single monitor's resolution, color depth, and refresh rate. The resolution setting applies to all monitors connected to the single video card.

Multiple video card configuration

In the multiple video card configuration, the computer has multiple video cards installed. Each video card connects a single monitor to the computer running an InTouch application.



Characteristics of a multiple card configuration

Dynamic Resolution Conversion (DRC) works with other distributed features to provide independence from

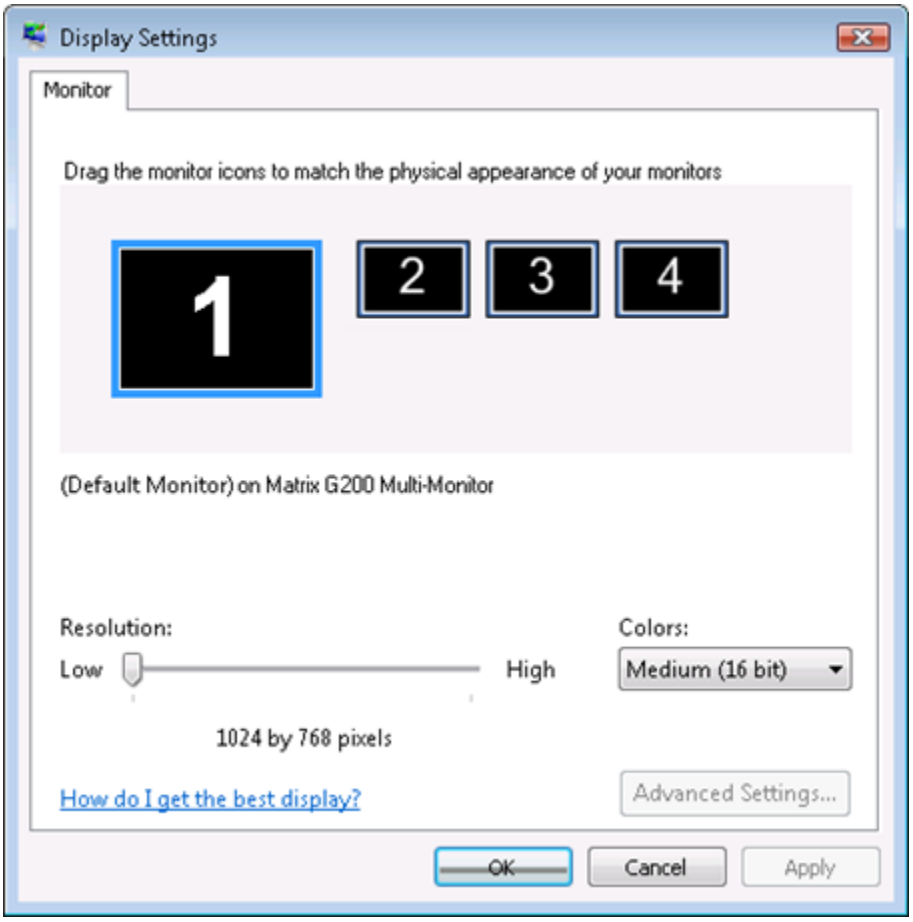
screen resolution restrictions. In a NAD architecture, you create and maintain an InTouch application on a development node and then copy it to several View nodes. DRC allows all view nodes to show the application, even if the nodes are running at different screen resolutions.

DRC enables each View node to scale the application to a number of user-defined options, including a custom resolution. This scaling takes place while WindowViewer compiles the application and does not require WindowMaker. Because each View node can use a different DRC setting, you must configure each individual View node.

DRC makes it easy to support multi-monitor systems. Simply select from the DRC resolution conversion options to show an InTouch application over the entire composite screen or just a portion of it.

Characteristics of multiple card drivers

The following figure shows the Windows **Display Properties** dialog box to configure the drivers for all monitors connected to individual video cards installed on the computer running an InTouch application.



Select a numbered rectangle in the **Display Properties** dialog box to select the monitor you want to configure. You arrange the numbered rectangles to match the physical placement of the monitors. Screen resolution, color depth and refresh rate apply only to the monitor you select.

Plan a multi-monitor application

To set up multiple monitors for your application

- Choose a multi-monitor video card
- Determine the application screen resolution
- Determine the number of monitors to display the application
- Determine the placement of application windows

Choose a multi-monitor video card

Technical Support can provide you with a list of recommended video cards that support multi-monitor InTouch applications.

Before you select a video card, get more information from Technical Support to answer the following questions:

- What versions of InTouch does the video card support?
- Does the card support a single or a multiple card configuration?
- What are the recommended drivers for the video card?
- What are the recommended configuration settings for the video card?

Determine the application screen resolution

Determining your overall screen resolution and knowing the exact size of your viewing area simplifies the process of creating an application for a multi-monitor environment.

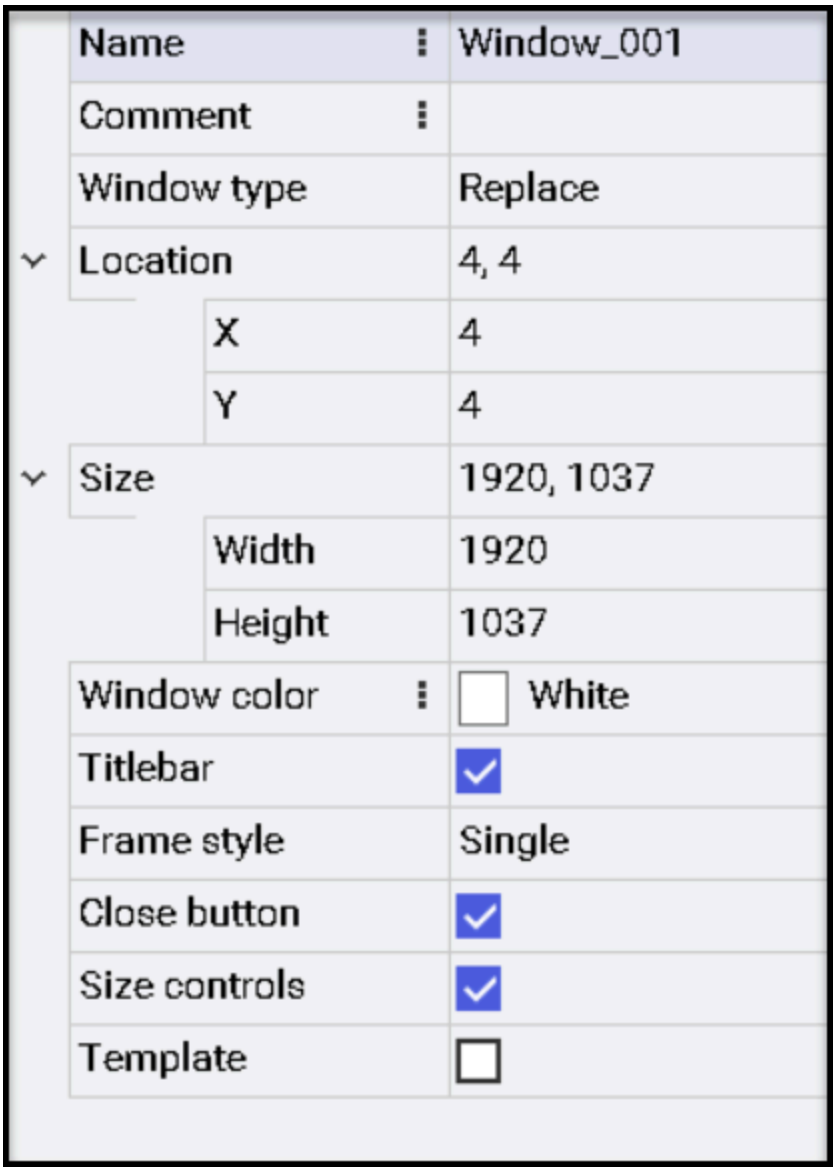
Create a drawing that shows the overall monitor configuration. The drawing should show the resolution of each monitor and the combined resolution of all the monitors together. This drawing helps you visualize the horizontal and vertical pixel range for each monitor.

For example, if you have a composite screen composed of two horizontal monitors with a screen resolution of 800 x 600, then the top left pixel location of the second monitor would be at pixel 800 x 0. The screen pixel count goes from 0 to 799 for the first monitor and 800 to 1599 for the second. Using the drawing as a guide, you can determine the placement of application windows on the composite multi-monitor screen.

Determine the number of monitors to display the application

You can simplify the effort to create a multi-monitor InTouch application by using a development environment similar to the production environment. Using a multi-monitor development environment may not be possible in all cases. When you only have a single monitor attached to the computer used to develop your InTouch application, you can still build a multi-monitor application by developing the windows and configuring the windows dimensions and locations to your estimated display needs.

Use the WindowMaker **Properties** pane to modify the characteristics of a window. In the **Windows** pane, select the window you want to modify. The Window **Properties** pane appears on the right navigation pane.



The **X Location** and **Y Location** values determine the horizontal and vertical pixel placement of a window's top left corner on a screen. The origin of horizontal and vertical pixel scales is at the top left corner of a screen.

The **Window Width** and **Window Height** settings determine the overall size of the window. For example, you can configure a window with the following settings:

- X location = 1024
- Y location = 0
- Window Width = 1024
- Window Height = 768

The multi-monitor configuration consists of four monitors arranged in a single horizontal row. Each monitor has a resolution of 1024 X 768. The overall composite screen resolution is 4096 X 768.

By setting the window's horizontal origin to 1024 and vertical origin to 0, you force this window to appear on the second monitor during run time. The window covers the entire screen surface of the second monitor.

Determine the placement of application windows

You can use several different configurations when developing InTouch windows for a multi-monitor environment.

Show windows in a forced location

One method is to simply develop and force windows to show where specified. Make sure that WindowViewer is maximized across the total viewing area of all monitors. This allows the InTouch application windows to show on specified monitors.

You can use InTouch security features to deny access to the Windows desktop.

Windows are moved manually

Another option is to develop an application where windows are manually moved to the monitors of choice, allowing a single application to run on different monitor configurations. This involves the following:

- All windows in the application must be of type **Popup**.
- The main WindowViewer parent window can be small and not covering all monitors. However, you cannot use InTouch security for denying access to the Windows desktop in this configuration because InTouch is not maximized.

In this configuration popup windows are used which can be easily moved to any monitors, regardless of the main WindowViewer parent window location. Popup windows do not have to remain within the parent WindowViewer window. You can shrink the size of the main window and move it to a corner of a monitor, allowing all the popup windows to be moved freely to the monitors of choice.

Windows are placed automatically based on environment

The final method includes an additional step added to the above method. The step allows an application to automatically place windows based on the environment used. This is the most complicated of configurations and requires extensive scripting and planning.

In this configuration, the ShowAt() and ShowTopLeftAt() script functions dynamically place windows based on a default set of coordinates and calculations. This can be configured many different ways depending on your application requirements.

Develop a multi-monitor InTouch application

You must assign values to selected parameters in the InTouch.ini and Win.ini files to support multi-monitors. These parameters enable you to place InTouch system dialogs and keypads in the proper locations on the composite screen.

Configure multi-monitor parameters

To enable multi-monitor support, add a set of InTouch parameters to the Windows Win.ini file. These parameters enable multi-monitor support for the node running the InTouch application and the resolution of each monitor.

Configure the multi-monitor settings on a node

1. Edit the Win.ini file located in the Windows folder of the computer running the InTouch HMI software.
2. Locate the [InTouch] section within the Win.ini file and add the following parameters:

Parameter	Description
MultiScreen=1	A value of 1 enables multi-monitor mode. A value of 0 disables multi-monitor mode.
MultiScreenWidth=*nnnn*	Width of a single screen in pixels.
MultiScreenHeight=*nnnn*	Height of a single screen in pixels.

For example, if you want to show your InTouch application with a screen resolution of 2560 x 1024 on two horizontal monitors, enter the following:

```
[InTouch]
MultiScreen=1
MultiScreenWidth=1280
MultiScreenHeight=1024
```

Configure screen resolution conversion

You can specify a parameter value to maintain the current resolution of InTouch application windows when migrating between nodes running different screen resolutions.

The ScaleForResolution parameter value determines whether application windows (*.win) are automatically scaled by WindowMaker after the display resolution changes on the computer running WindowViewer. The ScaleForResolution parameter does not affect the resolution of WindowViewer dialog boxes.

Configure screen resolution conversion on a node

1. Edit the InTouch.ini file of the computer running InTouch.
2. Add the ScaleForResolution parameter to the file.

   ```
   ScaleForResolution=1
   ```

   When set to 0, resolution conversion is disabled.

   When set to 1, resolution conversion is enabled.

Note: If the ScaleForResolution parameter is not added to the InTouch.ini file, the default value is enabled (ScaleForResolution=1). When you disable the parameter (ScaleForResolution=0), you are still prompted to convert the resolution. But, the resolution conversion does not occur.

Deploy the application and verify multi-monitor settings

The ScaleForResolution parameter becomes particularly important when you develop an application on a single monitor system that is intended to run on a multi-monitor system. The value assigned to the ScaleForResolution parameter determines whether the application can be scaled when moved from one environment to the other.

Important: It is recommended that you make a backup copy of the application before moving it to an different environment.

For example, if an application is developed on a computer with a single monitor with a resolution of 1024 x 768 and is intended to run on a system with four monitors in a side-by-side configuration with a total resolution of 4096 x 768, this requires an application conversion.

When you deploy the application on the multi-monitor system, a message appears prompting you to convert the application.

If the ScaleForResolution .ini setting is configured, you still see this message but the application is not converted and can then be run as designed. Select **Yes** to continue startup.

If the .ini setting is not configured, the InTouch HMI converts and scale all of the graphics and windows in the application to the new resolution. Doing so stretches and enlarge all windows and graphic displays, thus creating some unwanted results.

Important: Make sure that the multi-monitor Win.ini parameter settings are also configured on the destination computer before running your application. Win.ini settings do not automatically transfer with an InTouch application.

Verify multi-monitor support during run time

You can download an optional script function from the Technical Support script library that verifies if the local node running the InTouch application provides multi-monitor support.

The WWMultiMonitorNode() function determines if the node supports multi-monitors and the number of monitors attached to the node.

Typically, you run the WWMultiMonitorNode() function from a QuickScript to determine the number of monitors assigned to the node running the InTouch application.

The following example shows an example of a QuickScript statement with the value of the WWMultiMonitorNode() function assigned to an InTouch integer tag. The QuickScript can be set to run when the application starts in WindowViewer.

```
{MultiMonitors defined as an integer tag}
MultiMonitors = WWMultiMonitorNode();
{After executing this function Result = 4}
```

WWMultiMonitorNode() reads the MultiScreen parameter specified in the node's Win.ini file. The WWMultiMonitorNode() function returns either a 0 or a positive integer.

- 0 return value

  WWMultiMonitorNode() returns a 0 if MultiScreen=0 or if the MultiScreenWidth or MultiScreenHeight parameters are set incorrectly to 0 in the [InTouch] section of the Win.ini file.
- Positive integer return value

  WWMultiMonitorNode() returns the number of monitors in the multi-monitor configuration if MultiScreen=1 and the MultiScreenWidth and MultiScreenHeight parameters have been assigned correct screen resolution values.

Use InTouch on a tablet PC

Windows XP Tablet PC Edition and InTouch comes pre-installed with a line of portable Tablet PCs. These rugged Tablet PCs are waterproof and vibration resistant, making them suitable for most industrial environments. Tablet PCs are also available from other computer manufacturers that can run InTouch applications.

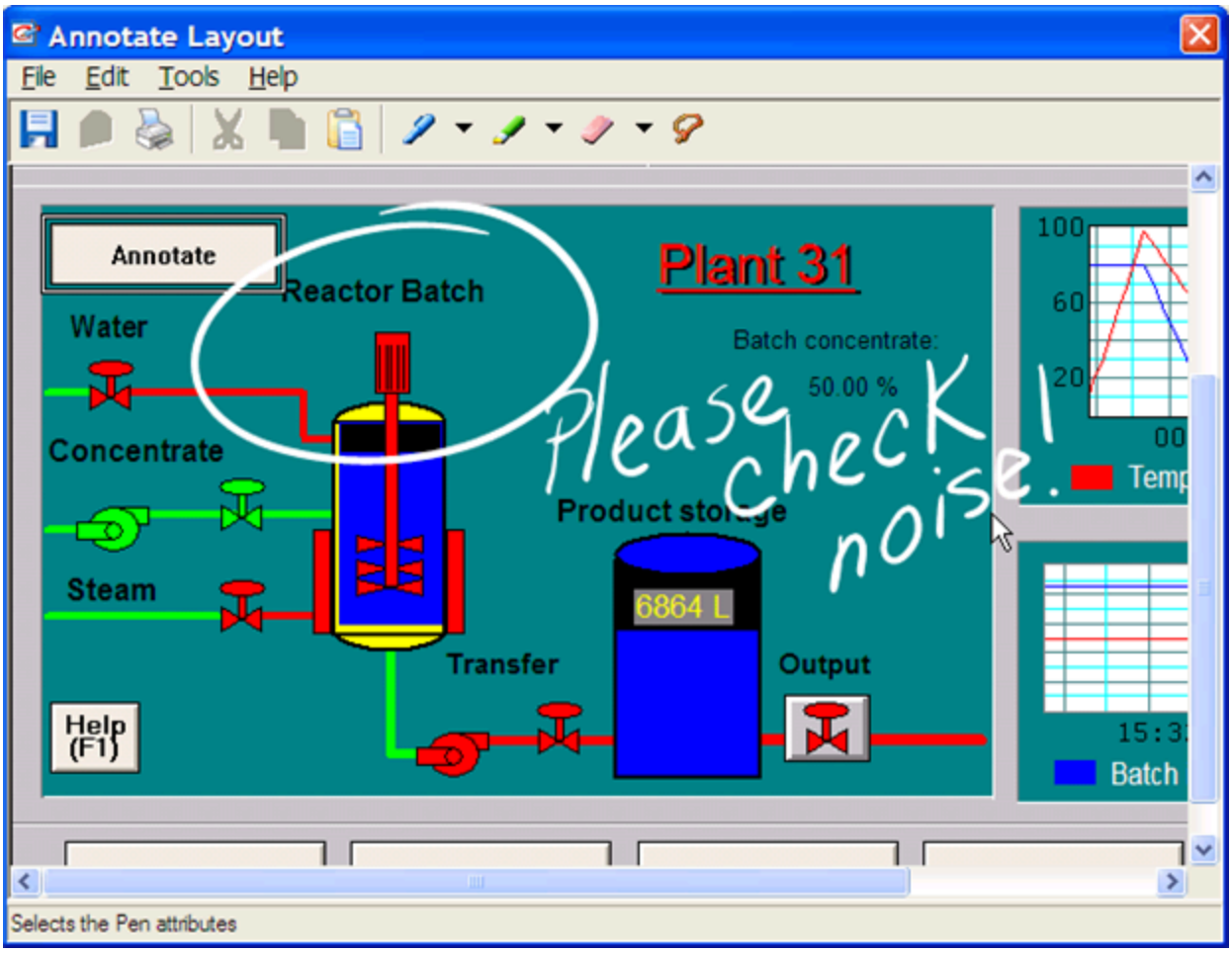
Operators carry a Tablet PC with them as they move around their plant. The Tablet PC runs an InTouch application that represents their actual plant processes. Using a pen that acts as a screen pointer or an input device, operators select InTouch objects on the screen or as a keyboard substitute to write notes directly on the screen.

Operators can write notes and annotate a running InTouch application with direct observations about their actual plant processes.

Annotate and send visualization screens as e-mail messages

Use the AnnotateLayout() script function to capture screens shown on a Tablet PC. The AnnotateLayout() function is available only when InTouch runs on a Tablet PC using the Windows XP Tablet PC operating system.

The AnnotateLayout() function takes a screen capture of the visible portion of the active InTouch window. The captured screen appears in the **Annotate Layout** dialog box.



The **Annotate Layout** dialog box contains a toolbar and menu options. The dialog box shows the screen capture in its client area. You can annotate the image using various drawing tools, and save, print, or send the screen capture in an e-mail message.

Make window annotations

Make annotations to the window with the following tools

- **Pen:** To draw and write comments.
  
- **Highlighter:** To highlight areas of the window using a semi-transparent color.
  
- **Eraser:** To delete parts of an annotation.
  

Each of these tools has certain options such as size, color, or transparency.

- To set these options, select the downward arrow next to each tool's icon and then select the command for the option.

- To restore these options to their default settings, on the **Tools** menu, select **Restore Defaults**.

Select, copy, and delete window annotations

You can select, copy, and delete annotations that you make in the window.

Select annotations

1. Select the **Lasso** icon in the toolbar.
2. While holding down the stylus button, draw an area around the annotations that you want to select.

   You can now cut, copy or delete the selected annotations.

Cut, copy, and paste annotations

- Use the standard Windows Cut, Copy, and Paste commands.

Delete annotations

- Do any of the following:
  - To delete all annotations on a window, on the **Edit** menu, point to **Clear** and then select **All**.
  - To delete annotations that you selected using the lasso, on the **Edit** menu, point to **Clear** and then tap **Selection**.

Save, print, and e-mail an annotated window

After you make annotations to a window, you can save it as an image file, print it, or send it as an e-mail attachment.

You only need to configure the e-mail server one time.

Save an annotated window

1. On the **File** menu, select **Save**. A standard Windows **Save As** dialog box appears.
2. Enter a name and format for the file and select **OK**.

Print an annotated window

1. On the **File** menu, select **Print**. A standard Windows Print dialog box appears.
2. Specify any printing options and select **OK**.

Send an annotated window as an e-mail attachment

1. On the **Edit** menu, select **E-Mail Configuration**. The **E-Mail Configuration** dialog box appears.
2. Enter the host name of the SMTP e-mail server to use for sending e-mail. If you are unsure, ask your administrator for assistance. Select **OK**.
3. On the **File** menu, select **E-Mail**. The **E-mail** dialog box appears.

4. Enter sender and recipient addresses and write a message. An image file of the annotated window is automatically added as an attachment.
5. Select **Send** to send the e-mail message.

AnnotateLayout() function

Shows the **Annotate Layout** dialog box, where you can annotate the current view screen from where this script function is called. This function is only supported on the Windows XP Tablet PC Edition operating system.

Category

System

Syntax

```
AnnotateLayout()
```

Remarks

When **Annotate Layout** dialog box appears, the screen image of WindowViewer is captured. Use the dialog box to:

- Annotate the screen capture using the pen in conjunction with tool bar and menu item settings.
- Save the image and the annotation as a .gif or .jpeg file.
- Print the image and the annotation (if a printer is configured).
- Send the image and the annotation as an attachment of an e-mail message (if SMTP is configured).

Change screen orientation

If the Tablet PC is running in tablet configuration, and WindowViewer is configured to dynamically change the application resolution to the screen resolution, an InTouch application developed in landscape mode is scaled to fit portrait mode.

If WindowViewer is not configured to dynamically change the application resolution, the landscape application is not scaled. In this case, some InTouch windows can be truncated on the Tablet PC.

When switching from one configuration to another, the screen resolution is switched by default. For example, if the tablet PC running in laptop configuration is switched to tablet configuration, the screen orientation switches from landscape (1024 x 768) to portrait (768 x 1024) mode.

Manage InTouch services

A service is a Windows process that performs a specific unattended background system function without a user interface or a required user logon.

The following startup options are available for Windows services:

- **Automatic**. When Windows restarts, the service automatically starts without any user intervention.
- **Manual**. A user or an application process must explicitly start the service.
- **Disable**. The service is prevented from starting. This is useful for troubleshooting.

Note: The parameters option in the InTouch WindowViewer service is not supported.

Services are started without compromising the Windows security system.

The InTouch HMI includes the following Windows services:

- Alarm DB Logger
- Alarm DB Purge/Archive
- NetDDE Helper
- SuiteLink
- WindowViewer

About managing InTouch services

A service is a Windows process that performs a specific unattended background system function without a user interface or a required user logon.

The following startup options are available for Windows services:

- **Automatic**. When Windows restarts, the service automatically starts without any user intervention.
- **Manual**. A user or an application process must explicitly start the service.
- **Disable**. The service is prevented from starting. This is useful for troubleshooting.

Note: The parameters option in the InTouch WindowViewer service is not supported.

Services are started without compromising the Windows security system.

The InTouch HMI includes the following Windows services:

- Alarm DB Logger
- Alarm DB Purge/Archive
- NetDDE Helper
- SuiteLink
- WindowViewer

Run WindowViewer as a service

If you configure WindowViewer to run as a Windows service, WindowViewer automatically starts when the computer on which the application is installed starts. The WindowViewer service runs in the background. If the WindowViewer service is running you cannot start another instance of WindowViewer.

Running WindowViewer as a service provides the following benefits:

- Most disaster recovery plans require that essential computer systems start immediately after electrical

power is restored. Microsoft Windows Servers can restart automatically after power is restored. When WindowViewer runs as a service, your plant automation system can begin running immediately. The last InTouch application that was opened in WindowViewer automatically starts when the computer restarts.

- WindowViewer continues to log historical data, gather alarm information, process scripts, act as an I/O Server, and write values as an I/O client, even as different operators log on and off.

Note: A logged on user must have proper access to the network location if a network application is used to run as a service or a network path is used as a historical logging folder.

If WindowViewer is already running as a service and you attempt to start it again from a shortcut icon or by selecting WindowViewer on the Windows **Start** menu, a message is logged in the Operations Control Log viewer. The message describes the restrictions to starting WindowViewer when it has been configured to run as a service.

If WindowViewer is already running as a service and you attempt to launch Application Manager or WindowMaker, a warning message will be logged in the Operations Control Log viewer.The message explains that Application Manager and WindowMaker cannot open when WindowViewer is running as a service.

Important: When running WindowViewer as a service, the user account privileges have been set to non-interactive to reduce the potential security exposure of running a service with administrator privileges.

Configure WindowViewer to start as a service

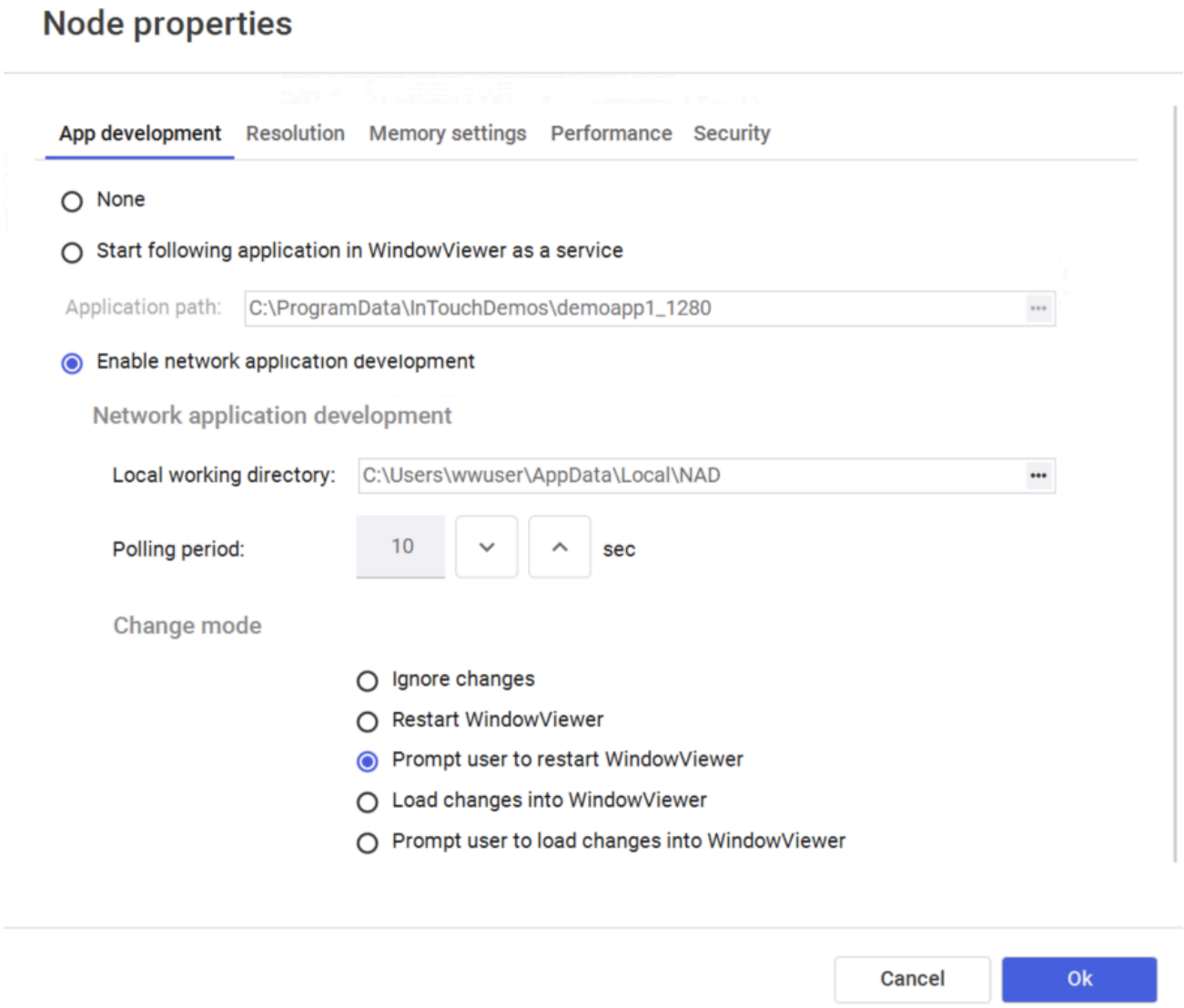
Running WindowViewer as a Windows service provides continuous operation after the operator logs off and automatic start up at system boot time without operator intervention. This allows unmanned station start up of WindowViewer without compromising operating system security.

When WindowViewer is configured to start as a service, an InTouch application must also be specified to run in WindowViewer as a service. You can specify the application directory in the **Node Properties** dialog box or manually enter it into the WIN.INI file.

Configure WindowViewer to start as a service

1. Launch InTouch Application Manager.
2. On the **Tools** menu, select **Node Properties**.

   The **Node Properties** dialog box appears.



3. Select **Start following application in WindowViewer as a service** to configure WindowViewer to automatically run as a service.

   The **Application path** field will become enabled.
4. Select the ellipsis button to prompt a file explorer and navigate to your InTouch application.

   The application directory will populate in the group box.
5. Select **OK**.
6. Select the WindowViewer icon in the Application Manager toolbar.

   WindowViewer will now run as a service for the specified InTouch application.

Note: You can also fast switch from WindowMaker to WindowViewer to start the WindowViewer service for the InTouch application if you have configured the Node Properties as described in the above steps. You can do this in place of starting WindowViewer from the Application Manager.

Edit WIN.INI to run application as service in WindowViewer

If the option **Start following application in WindowViewer as a service** is enabled in the **Node Properties**, you can manually enter the application directory into the WIN.INI file. If you update the WIN.INI file before selecting the application in Application Manager, WindowViewer runs as a service for the application once selected.

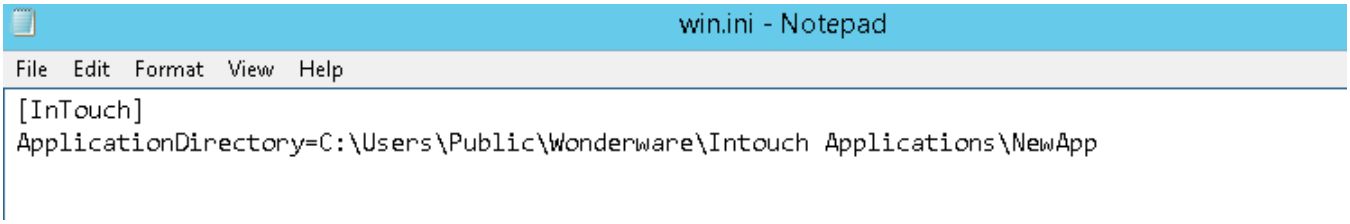
You can also update the WIN.INI with the application open in WindowMaker. If you then fast-switch to run time, WindowViewer runs as a service for the application.

Note: The above functionality is not supported for Managed InTouch applications. If you attempt to fast-switch a Managed application from WindowMaker to run as a service in WindowViewer, a warning message will be logged in the Logger.

The WIN.INI is located here:

```
C:\ProgramData\Wonderware\InTouch\Service\win.ini
```

Enter the directory of the application you want to run as a service, as in the example below:



Start a service manually

You can manually start the InTouch WindowViewer service using the Windows Control Panel.

WindowViewer does not appear in the Services Control Panel unless you configured it to start as a service. For more information, see Configure WindowViewer to start as a service.

Start the WindowViewer service using Control Panel

1. Start the Control Panel.
2. Double-click **Administrative Tools** and then double-click **Services**. The **Services** dialog box appears.
3. In the details pane, right-click **Wonderware WindowViewer** service and then select **Start**.

   Important: The command prompt cannot be used to start WindowViewer as a service.

Stop a service

You can manually stop the WindowViewer service using the Control Panel.

Stop the WindowViewer service using the Control Panel

1. Start the Control Panel.
2. Double-click **Administrative Tools** and then double-click **Services**. The **Services** dialog box appears.
3. In the details pane, right-click WindowViewer and then select **Stop**.

Configure the user account for InTouch services

By default, Windows services run using the local system account. InTouch services require a user account with administrative privileges, which may not be provided by the local system account.

When you install the InTouch HMI, you specify an administrative account that all AVEVA services run under, if the account was not created already. This account is considered the master account. The InTouch services use the master account to automatically start up.

Note: The master account is also called the impersonation account. An impersonation account is the user or group account that provides access to the restricted resource "area" of your site or server.

If you want to change the master account, use the Change Network Account Utility.

Caution: Changing the master account affects all AVEVA services, not just InTouch services.

Change the master account

1. On the **Start** menu, point to **Programs**, point to **AVEVA**, and then select **Change Network Account**. The **Change Network Account** dialog box appears.
2. Change the user account. For more information, see the Change Network Account documentation.
3. Select **OK**.

Troubleshoot InTouch services

If a service depends on other services starting before it can start, Windows verifies if the prerequisite services are running before starting the service.

Depending on your requirements for running WindowViewer, be aware of the following dependencies:

- The NetDDE Helper service must be running if you plan to use Distributed Alarming or Distributed History or if you intend to access network DDE data.

  The NetDDE Helper service also depends on both the Network DDE and Network DDE DSDM services being installed and configured for either Manual or Automatic startup. During installation, the NetDDE Helper service is configured for Manual startup. WindowViewer automatically starts this service when the computer starts.
- If you need WindowViewer to act as a SuiteLink server or client, then the SuiteLink service must be running.

  The SuiteLink service also requires that Microsoft TCP/IP be installed.
- If you want to store any messages or errors while WindowViewer is running, you must make sure that the Operations Control Log Viewer service is installed.

  Both the SuiteLink and Operations Control Log Viewer services should be installed and configured to run in automatic startup.

View error messages for services

Use the Windows Event Viewer to troubleshoot error messages related to services. For example, you may see the error "One or more services failed to start …" The Windows Event Viewer lists informational messages, warnings, or errors that occurred while starting Windows services. For more information about the Event Viewer, see your Microsoft documentation.

You can see any warning or error messages that resulted from an InTouch service failing to start. If the Event Viewer indicates that the WindowViewer service failed to start, the most likely cause is a dependency on a prerequisite service that is not running.

Troubleshoot problems with the services user account

If InTouch services fail to install or start after you install the InTouch HMI, you could have a problem with the user account that they run under.

Troubleshoot services user account problems

1. Open the Windows User Manager window and create a new master user account.

   This user account must have administrative privileges on the local computer to start an InTouch component as a service. If you do not see your computer's node name in the domain list, then manually type in the node name.

   For more information, see Configure the user account for InTouch services.
2. Verify that your computer's node name is no longer than 14 characters. If the node name contains underscore characters (_) or dashes (-), remove them.
3. During installation when you are prompted to enter the domain name, type in the node name of your computer, not the domain name. Then, type the user name that was created in step 1 and your password.
4. If you already installed the InTouch HMI, you can still specify the domain name, user name, and password by running the ArchestrA Change Network Account Utility.
5. Reboot your computer.
6. Log onto your network domain with any valid user account. Even if your domain goes down, it does not affect your InTouch application that is running on the local computer.

Deactivate advised I/O items

When you start up the Windows operating system, the services that are configured to automatically start will start in the "background" with no visible user interface appearing on the desktop. The services in this situation are running in the system context. When an operator logs on the system, any services that are running in the system context that have an associated user interface automatically appear on the desktop. In this situation, the services are now running in the desktop context.

If you configure the WindowViewer service to automatically start, the service runs in the system context when the operating system starts. Then, when a user logs on, the WindowViewer service continues to run but in the desktop context, and the WindowViewer user interface automatically appears.

If you have InTouch Access Names defined with the **Advise only active items** option turned on, and have I/O tags that are active only in certain InTouch application windows (the tags are not used anywhere else in the application), it is possible to "deactivate" those tags. For example, if WindowViewer is running as a service, and you close an application window using a script, the window automatically is unloaded from memory, thus terminating the link to those tags.

Registry keys for the InTouch services

The InTouch services are listed as keys in the Windows registry:

SuiteLink:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SLS
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\slssvc
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SuiteLink

NetDDE Helper:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WWNetDDE

WindowViewer:

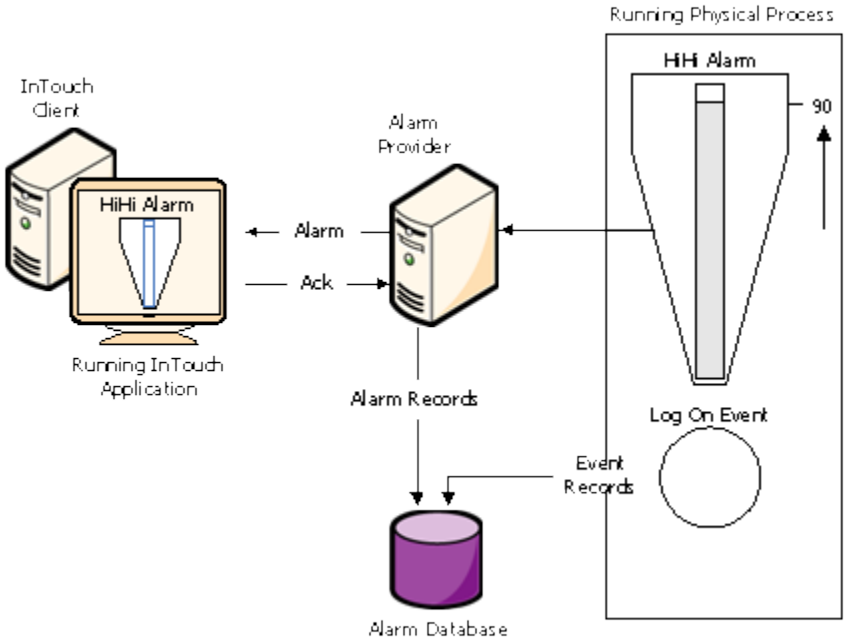
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VIEW

Alarms

You can create InTouch applications that generate alarms and events to notify operators about the status of process activity.

- Alarms warn run-time operators about process conditions that could potentially cause problems. Typically, you set up an alarm to trigger when a process value exceeds a defined limit. An operator must usually acknowledge the alarm.
- Events represent normal system status messages. A typical event is when a system condition occurs, such as an operator logging on to an InTouch application. Operators do not have to acknowledge events.

The following figure shows how the InTouch HMI handles alarms and events while an application is running. Alarm and event data is saved to the alarm database.



You can configure any tag for event monitoring. An event message is logged to the alarm system each time the tag value changes. The event message includes how the value changed and whether the operator, I/O, scripts, or

the system initiated the change.

Migrate from legacy alarm systems

You can migrate your applications built using the Standard Alarm System or AlarmSuite.

About migrating from legacy alarm systems

You can migrate your applications built using the Standard Alarm System or AlarmSuite.

Migrate from the standard alarm system to the distributed alarm system

When you migrate a Standard Alarm system to the Distributed Alarm system, all of the Standard Alarm Displays in the master/slave application are migrated to Distributed Alarm Display objects.

Colors, fonts, the expressions, and the alarm query settings are not migrated. The new Distributed Alarm Display object has the following default query, where node name is the name of the master node:

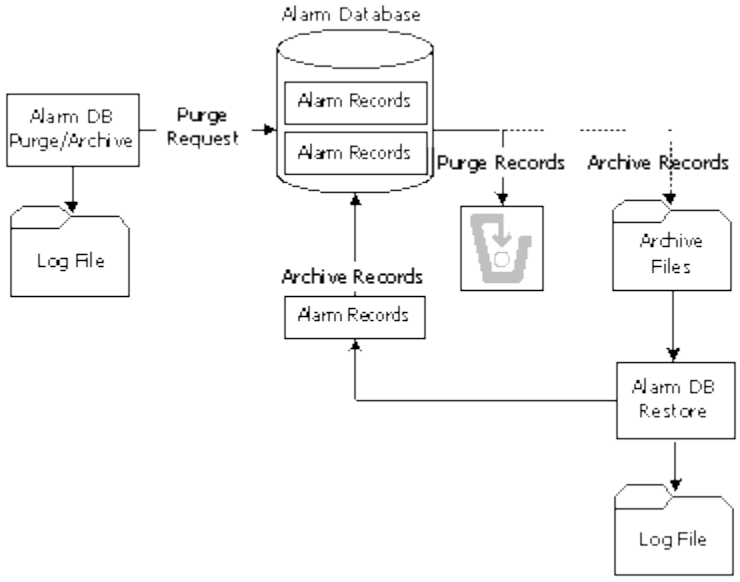
\\nodename\InTouch!$system

The acknowledgement and alarm status dotfields continue to work as before. Depending if the I/O tag was configured for NetDDE or SuiteLink, you may need to enable NetDDE. However, you may decide you no longer need separate controls for issuing acknowledgements, as the alarms can now be acknowledged using the Distributed Alarm Display object.

Maintaining the alarm database

You manage the alarm database using two InTouch utilities. Use the Alarm DB Purge-Archive utility to remove records from the database permanently or archive them to files. If the database becomes corrupt, use the Alarm DB Restore utility to restore archived records.

The following figure shows how both utilities purge/archive records and then restore them back to the database.



You must be logged on to the computer as an administrator to use the Alarm DB Purge-Archive utility.

Configure purge or archive settings

Use the Alarm DB Purge-Archive utility to:

- Select the type of records to purge from the alarm database.
- Purge records automatically on a daily, weekly, or monthly schedule.
- Optionally archive purged database records to files.
- Save the status of archive or purge operations to a log file to troubleshoot problems.
- Show the status of purge or archive operations.

Important: The Alarm DB Purge-Archive utility runs as a set of Windows services. To reduce the security exposure of running the Alarm DB Purge-Archive utility with administrator privileges, the user account permissions are set to non-interactive.

Connect to the alarm database

Before you can use the Alarm DB Purge-Archive utility, you must connect to the alarm database.

Configure the database connection

1. Open the **Alarm DB Purge-Archive** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Purge-Archive**.
2. Select the **Database** tab.

3. Configure the database connection. Do the following:
   a. In the **Authentication** list, select the authentication method: **SQL Server Authentication** or **Windows Authentication** (default).

Note: Windows authentication can provide better application security than SQL Authentication. If you switch from Windows Authentication to SQL Authentication, a pop up dialog will appear recommending that you use Windows Authent**ic**ation for this reason. If you choose to ignore this warning and proceed with SQL Authentication, select OK. A similar message will be logged in the Log Viewer.

   a. In the **Server Name** list, select the node name of the server.
   b. In the **Database** box, type the name of the alarm database.
   c. In the **Credential Information** area, from the **Credentials** drop-down, select the credentials for authentication.

Note: The Credentials field is enabled only when you select SQL Server authentication type. The Windows

Authentication method uses the credentials of the user currently logged in, and disables the Credentials field. For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credential Manager in AVEVA™ InTouch HMI Application Development.

4. Select **Test Connection** to test the connection to the database. A message indicates if the connection to the alarm database is successful.
5. Select **OK**.
6. Select **Apply**.

Configure how much data to purge from the server

You can:

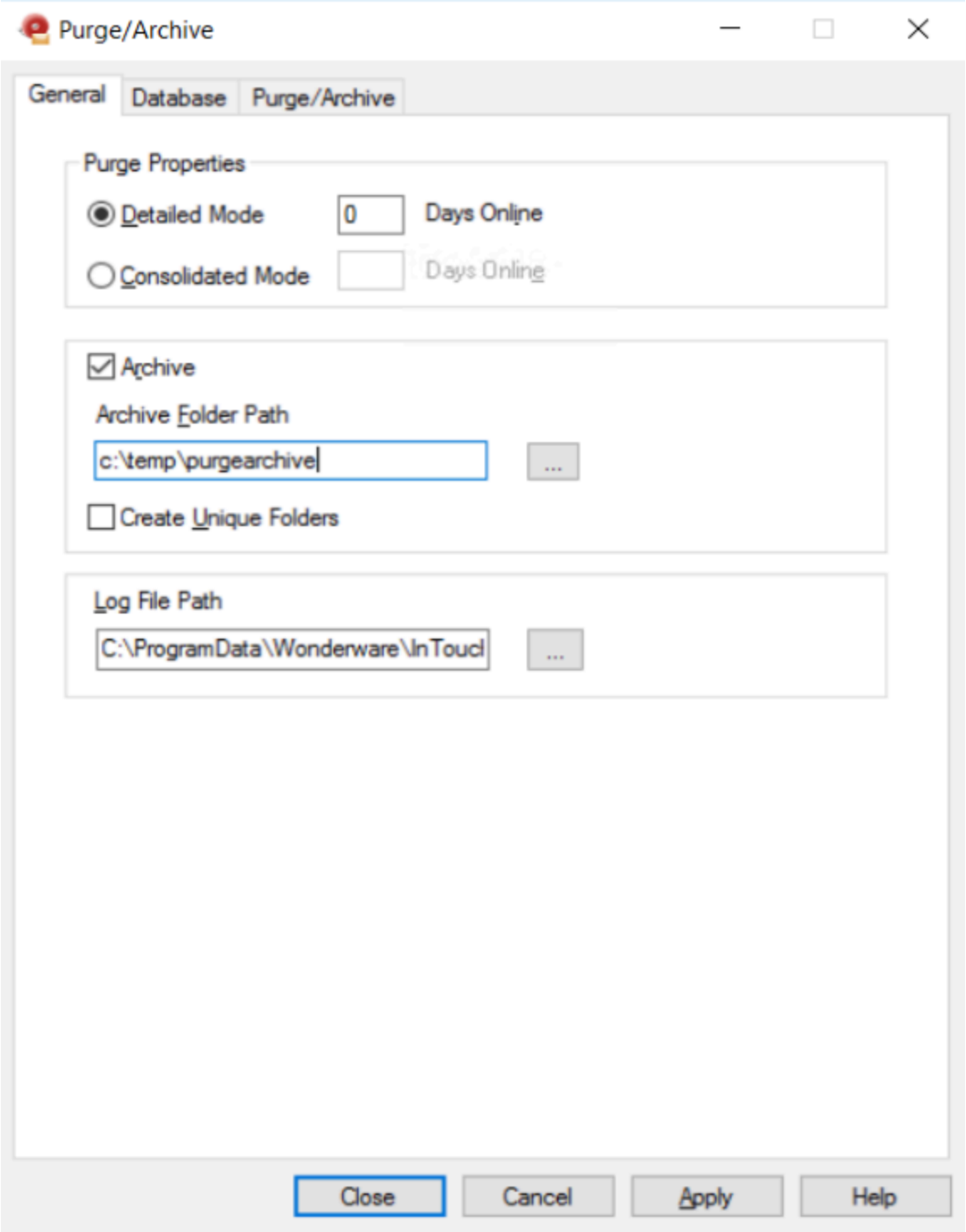
- Select the type of alarm records to be purged from the alarm database.
- Optionally archive purged records from the alarm database to files.
- Select the folder location to store the purge log file.

You can select the type of table that needs to be purged, either the AlarmDetail or AlarmConsolidated table.

All data from the day previous to the number specified is purged. Valid entries are 0-9999. If you select 0, all records are purged from the alarm database except the current day's records.

Select records to purge

1. Open the **Alarm DB Purge-Archive** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Purge-Archive**.
2. Select the **General** tab.



3. In the **Purge Properties** area, configure the type of records to purge. Do either of the following:
   - Select **Detailed Mode** to purge alarm records that are saved in the database in Detailed mode.
   - Select **Consolidated Mode** to purge alarm records that are saved in the database in Consolidated mode.
4. In the **Days Online** box, type the number of days worth of records to retain in the alarm database.
5. Select **Apply**.

Configure the archive of purged data

Archive the records purged from the alarm database and then restore them using the Alarm DB Restore utility.

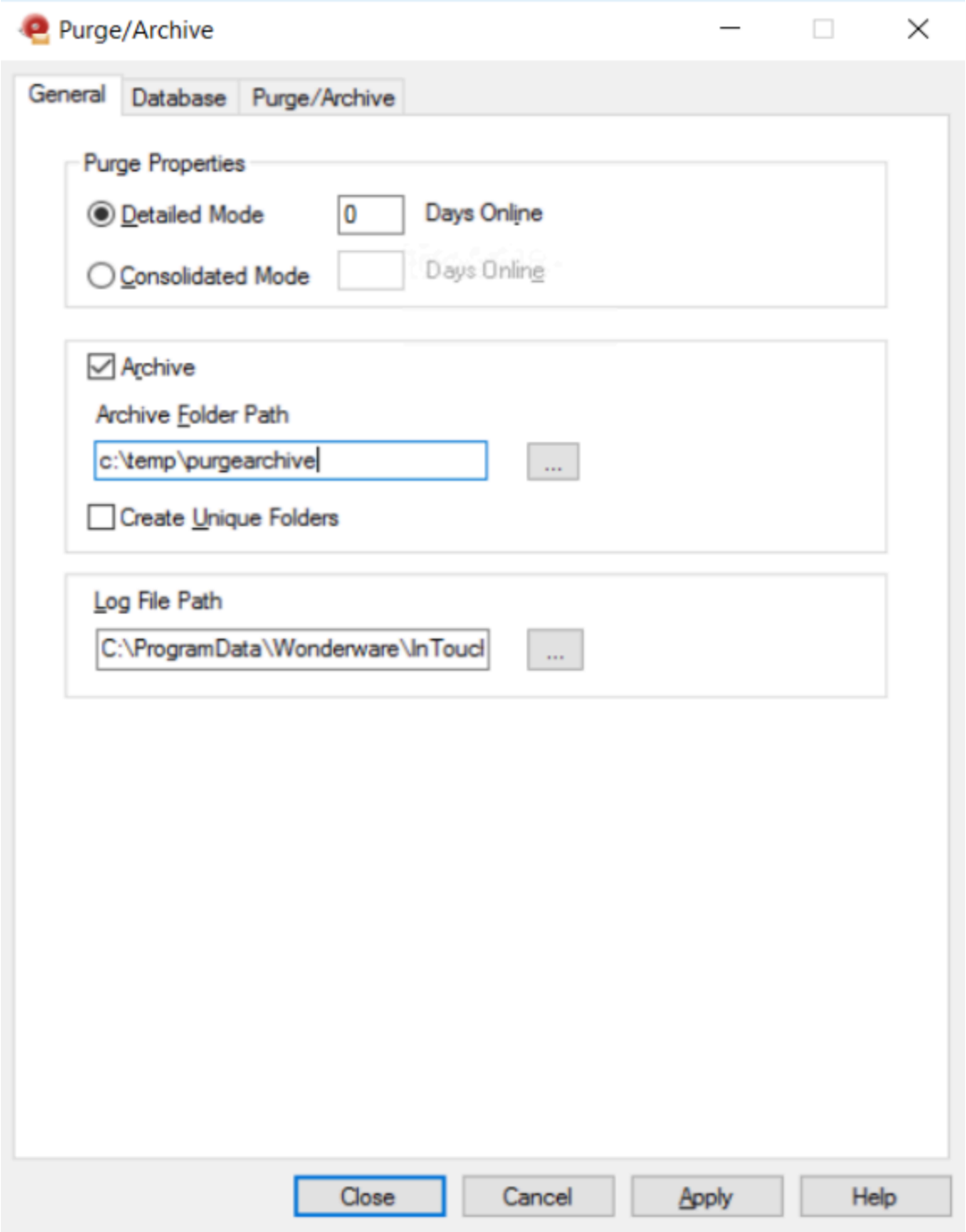
When you purge the alarm database, the Alarm DB Purge-Archive utility automatically creates a set of nine archive files that correspond to the purged alarm database tables. Each file contains the purged records of a single table.

The Alarm DB Purge-Archive utility assigns names to the archive files based upon the table name, date, and time when the purge operation occurred. For example, the name of the archive file for the AlarmMaster table that was purged on June 22, 2007 at 5:30 p.m. is formatted like the following:

AlarmMaster_06222007_1730.txt

Configure the archive

1. Open the **Alarm DB Purge-Archive** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Purge-Archive**.
2. Select the **General** tab.



3. Select the **Archive** check box.
4. In the **Archive Folder Path** box, type the folder location where archive files should be saved or select the ellipsis button to browse for the location.
5. Select the **Create Unique Folders** check box if you want the archive files to be placed in an individual sub-folder beneath the archive file folder.
6. Select **Apply**.

Configure log file settings

The Alarm DB Purge-Archive utility generates status messages during a purge operation. You can view these messages online from the utility's **Status** window. The Alarm DB Purge-Archive utility also writes purge messages to the purge log file named WWAlmPurge.log.

The example below shows the messages stored in the log file after a successful purge operation.

```
Purge Started on 12:16:48 PM 6/22/2007
Starting transaction....
Archiving Table ProviderSession...
Archiving Table Query...
Archiving Table Cause...
Archiving Table Alarm Master...
Archiving Table OperatorDetails...
Archiving Table Alarm Detail...
Archiving Table Comment...
Archiving Table Events...
Archiving Table TagStatus...
Purging records in the database...
Committing....
Purge Completed On 12:16:52 PM 6/22/2007
144 records from AlarmMaster were purged along with the related records from other tables.
```

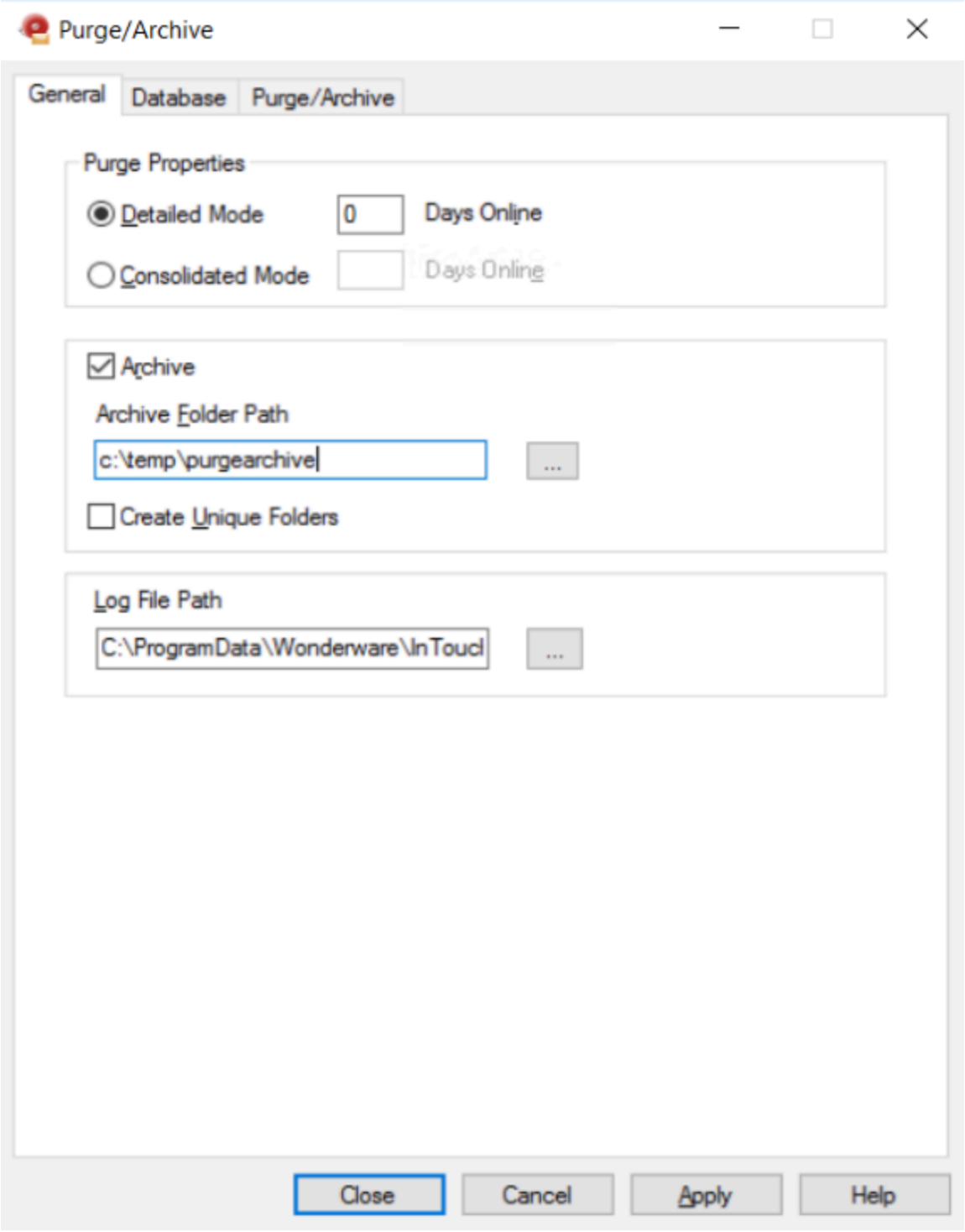
By default, the purge log file is stored in this folder: C:\Documents and Settings\All Users\Application Data\Wonderware\InTouch. For computers running the Microsoft Windows Vista operating system, the default application folder is C:\Users\UserName\Documents\My InTouch Applications.

You can change the storage location of the purge log file.

The Alarm DB Purge-Archive utility appends new messages to the log file each time a purge occurs.

Set archive logging

1. Open the **Alarm DB Purge-Archive** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Purge-Archive**.
2. Select the **General** tab.



3. In the **Log File Path** box, type the folder location where the purge log file should be placed or select the ellipsis button to browse for the location.
4. Select **Apply**.

Manually purge and archive the database

You can purge and archive your alarm database manually. This overrides the activation time and starts the

purging and archiving immediately.

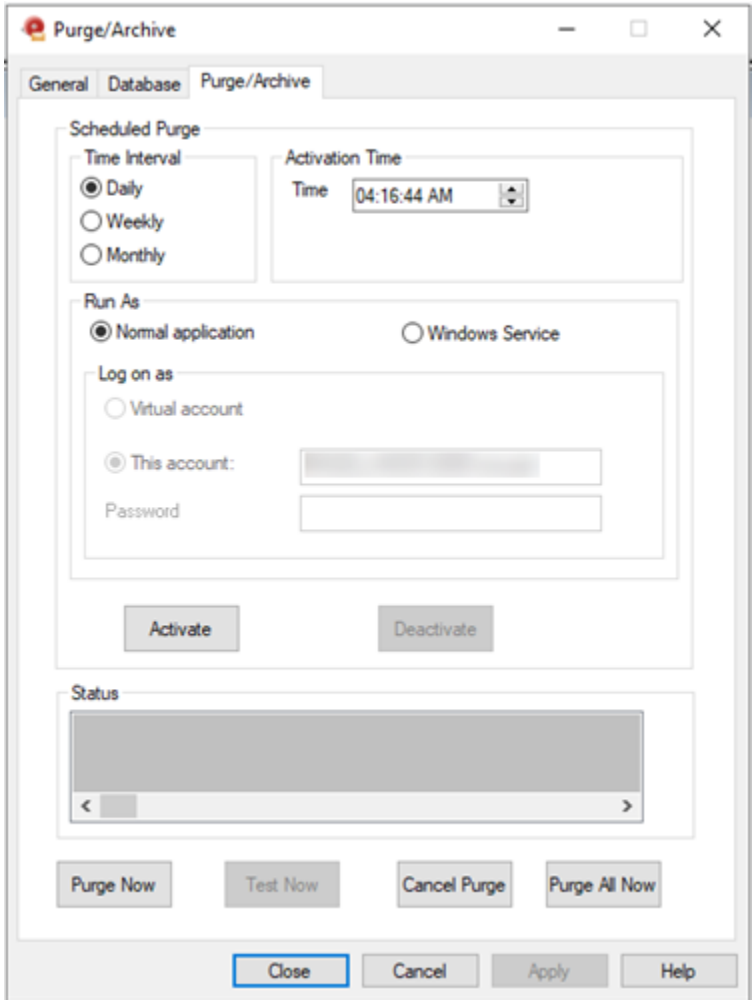
The purge operation checks for the presence of an archive file and appends to the same. If the archive file is not present, the file is created as per the naming convention and then used for archiving.

The purge operation does not delete entries in tables such as ProviderSession, Query, and Cause that are linked to the main tables such as AlarmMaster through foreign key constraints. The related records in these tables are written to the files to maintain the data consistency and also retained in the database.

Caution: Manually purge all records (the Purge All Now option) only when the Alarm DB Logger service is stopped. If the purge operation is committed successfully while the Alarm DB Logger service is running, the Alarm DB Logger service stops logging and starts caching records.

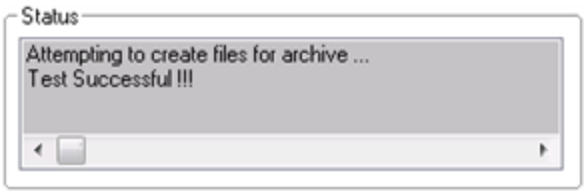
Manually purge and archive records from the alarm database

1. Open the **Alarm DB Purge-Archive** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Purge-Archive**.
2. Select the **Purge/Archive** tab.



3. Select **Test Now** to perform a test purge to verify your connection to the database and archive locations.

   The test purge creates empty archive files in the specified archive folder. The **Status** area shows a message that the test was successful.



The **Test Now** button is available only if you have chosen to archive your purged records. The Archive option is located on the **General** tab.

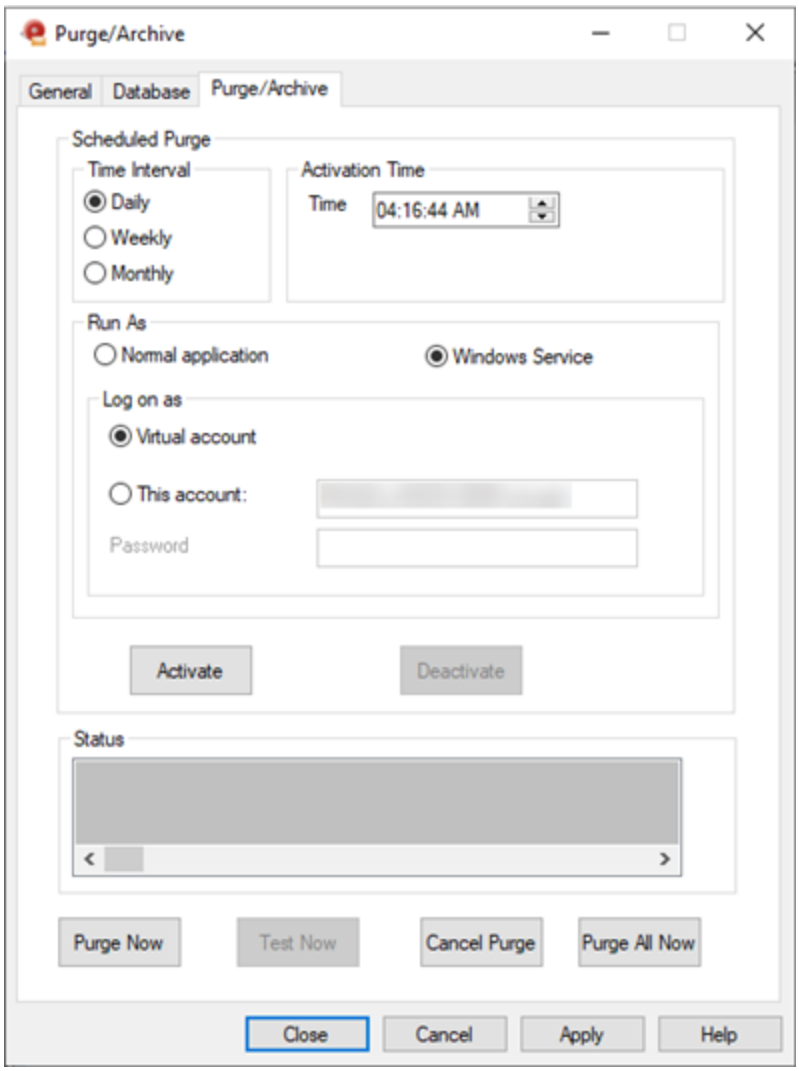
4. Purge the records from the database. Do either of the following:
   - Select **Purge Now** to purge the selected records.
   - Select **Purge All** Now to purge all records.
5. To stop a purge, select **Cancel Purge**. If you cancel the purge, the alarm database is rolled back to its original state.

Set a schedule for automatic purging

The Alarm DB Purge-Archive utility can automatically purge or archive records from the alarm database at scheduled intervals. You can perform a test purge to verify your connection to the database and target locations and to start and stop purging.

Set a schedule for automatic purging

1. Open the **Alarm DB Purge-Archive** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Purge-Archive**.
2. Select the **Purge/Archive** tab.



3. In the Time Interval area, select a purge interval, either daily, weekly, or monthly.

   If you select **Weekly** or **Monthly**, a **Day** box appears in the **Activation Time** area for you to specify the day of the week or day of the month.

   If you select **Daily**, in the **Time** box, configure the time of day that you want the purge/archive operation to start.

4. In the **Run As** area, select **Normal application** to run the purge-archive utility as an application or select **Windows Service** to run it as a service.

   For **Windows Service**, either select **Virtual account** or specify the username and password for another account under the **This account:** area.

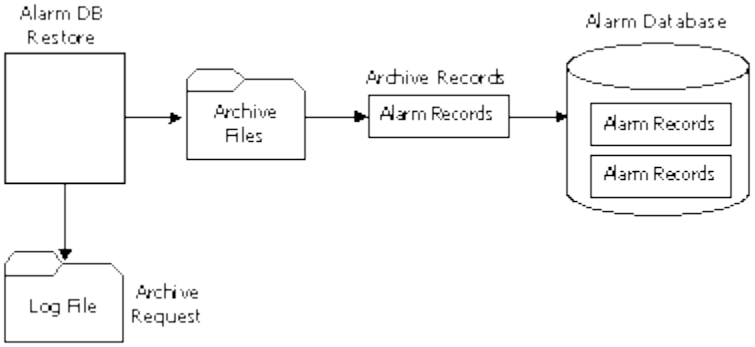
   Note: For more information on Virtual account, see Use virtual accounts.

5. Select **Apply** to save your purge and archive settings.
6. Select **Activate** to place the Alarm DB Purge-Archive utility on an automatic purge schedule.
7. Select **Close**.

   To stop the automatic purge schedule and remove the Window Service, select **Deactivate**. After a brief delay the service will be removed from Window Services.

Restore the alarm database

The Alarm DB Restore utility restores the archived alarm records in the archive files back to your alarm database. The following figure summarizes the steps to restore alarm records to the database.



To restore a database, you must

- Configure the connection to the alarm database.
- Select which records to restore to the alarm database.
- Restore archived records to the alarm database.

When minimized, the Alarm DB Restore utility appears as an icon in the system tray. When you right-click the icon, a menu shows the following commands:

Command	Description
Restore	Begins the restoring process.
Cancel Restore	Cancels the restoring process.
Clear Status	Clears the status window.
Hide Window	Minimizes the Alarm DB Restore utility to an icon in the system tray.
Show Window	Opens and maximizes the Alarm DB Restore utility.
Exit	Closes the Alarm DB Restore utility.

If you right-click in the Alarm DB Restore utility, the same menu appears.

Configure the database connection

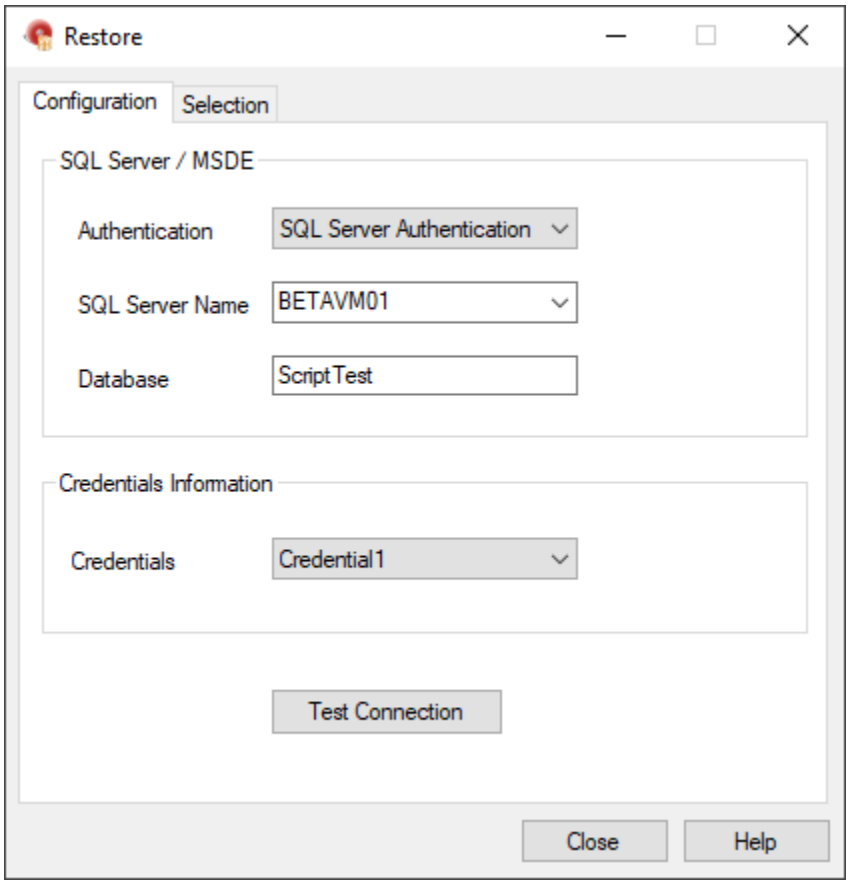
You must select a database to restore the archived data to. If the specified database is not present on the server, you are prompted to create a new database with default server parameters.

Configure a database for restoring archived data

1. Open the **Alarm DB Restore** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.

b. Double-click **Alarm DB Restore**.

2. Select the **Configuration** tab.



3. Configure the connection to the alarm database. Do the following:

a. In the **Authentication** list, select the authentication method: **SQL Server Authentication** or **Windows Authentication** (default).

Note: Windows authentication can provide better application security than SQL Authentication. If you switch from Windows Authentication to SQL Authentication, a pop up dialog will appear recommending that you use Windows Authent**ic**ation for this reason. If you choose to ignore this warning and proceed with SQL Authentication, select **OK**. A similar message will be logged in the Log Viewer.

b. In the **SQL Server Name** list, select the node name of the server that hosts the alarm database.

c. In the **Database** box, type the name of the alarm database.

d. In the **Credentials Information** area, from the **Credentials** drop-down, select the credentials of alarm database for authentication.

The Credentials field is enabled only when you select SQL Server authentication type. The Windows Authentication method uses the credentials of the user currently logged in, and disables the Credentials field.

Note: For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credential Manager in AVEVA™ InTouch HMI Application Development.

a. Select **Test Connection** to test your connection to the database. A message indicates whether the

connection to the alarm database is successful or not.

b. Select **OK**.

4. Select **Close**.

Configure which files to restore

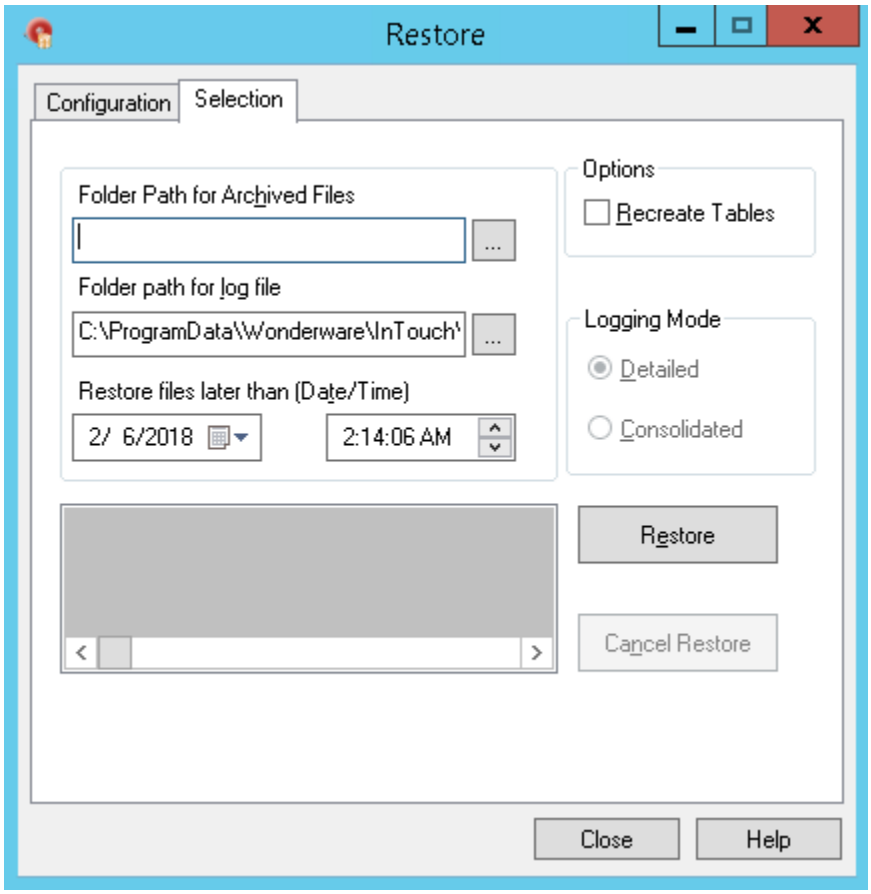
You can select a time period for the records to restore and whether you want the database tables to be recreated.

If you cancel the restore, the database is rolled back to its original state.

Caution: If you try to restore archived alarms that are already present in the database, the archived records are not restored. This avoids duplicate alarm/event entries in the database. The Alarm GUID or Event GUID associated with records determines whether an alarm or event is already present in the database.

Select database records to restore

1. Open the **Alarm DB Restore** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Restore**.
2. Select the **Selection** tab.



3. In the **Folder Path for Archived Files** box, type the full path (up to 255 alphanumeric characters) to the location of the archived files or select the button to locate and select the folder where archived files are stored.

4. In the **Restore files later than (Date/Time)** area, select the date and time to start restoring records to the database.

   The starting date and time are set by default to the current date and time.
5. In the Folder path for log file box, type the full path (up to 255 alphanumeric characters) where the log files are created and stored or select the button to locate and select a folder.
6. If you select the **Recreate Tables** check box, the tables of the specified alarm database are recreated.
7. Depending on the type of **Logging Mode** you selected for the alarm records contained in the archived files, select:
   - **Detailed** - Recreate the alarm database tables in detailed mode.
   - **Consolidated** - Recreate the alarm database tables in consolidated mode.

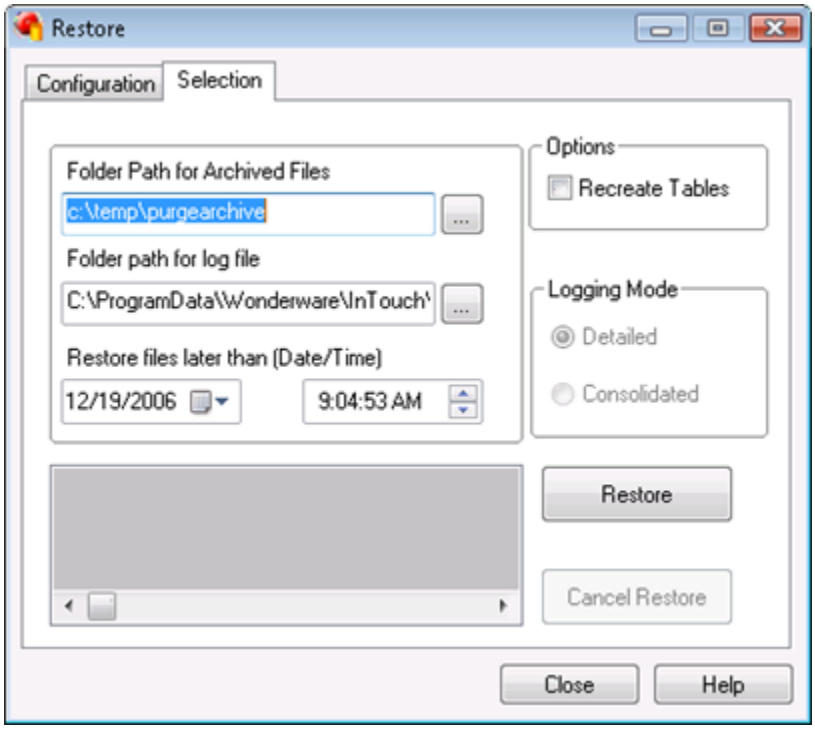
   Important: Recreating tables overwrites all records currently stored in the alarm database.
8. Select **Restore**.

Start a database restore operation

Restore archived database records after you have established the database connection, specified the archived files folder and a time filter.

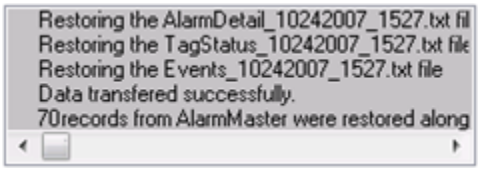
Restore database records from an archive

1. Open the **Alarm DB Restore** utility. Do the following:
   a. In the **Tools** view, expand **Applications**.
   b. Double-click **Alarm DB Restore**.
2. Select the **Selection** tab.



3. Select **Restore**. A message shows whether the restoration is successful and the number of records restored

to the database.



Customize applications settings from the INTOUCH.ini file

The first time you run an InTouch application, the INTOUCH.ini file is created in the application folder. When the INTOUCH.ini file is created, values are assigned to a set of parameters that determine the operating characteristics of an individual InTouch application.

As you continue configuring your application from WindowMaker or WindowViewer, new INTOUCH.ini parameters are created or existing parameters are modified. For example, when you configure logging from the WindowMaker **Historical Logging Properties** dialog box, logging parameters are added to the INTOUCH.ini file.

Other configuration parameters must be manually added to the INTOUCH.ini file.

After you customize your application, you can copy the INTOUCH.ini file to a different application's folder. This way, you can create consistent operating characteristics for your applications without having to repeat all customization steps.

Custom INTOUCH.ini parameters

The following table lists a set of parameters that you can manually enter in the INTOUCH.ini file to provide additional custom properties to your InTouch applications.

INTOUCH.ini Parameter	Purpose
16PenTrendDrawMode	Determines whether a 16-Pen Trend shows data values in average mode or min-max mode.
ApplicationThumbnail	Sets the name of the application thumbnail file.
AllowPubAppEdit	Sets the application flag, so that it can edit a published application. If the value is 1, you can edit a published InTouch file.
CommentRetentive	Determines whether run-time changes to the Alarm Comment field are saved.
ForceLogCurrentValue	Determines whether the current value of logged tags are written to the Historical Log file at an interval set by the ForceLogging parameter.
ForceLogging	Sets the length of the interval when tag values are periodically written to the Historical Log file regardless of their current values.

INTOUCH.ini Parameter	Purpose
LoopTimeOut	Sets the time out period of FOR-NEXT loop processing in an InTouch script.
MarkAppReadOnlyNonRDS	On a non-RDS node, if this parameter is set to 1, it will consider this a read-only node and consume a read-only license for an InTouchView application.
MouseMustBeOnObjectForOnKeyUp	Determines if the mouse must be over the object for the On Key up action to be triggered. The value is '1' by default.
NoKeyboardResize	Determines whether the numeric keyboard is resized to the resolution of the WindowViewer screen.
OldRightMouseBehavior	Determines whether the right mouse button is enabled in WindowMaker.
PrintScreenWait	Sets the wait period before printing a screen from WindowViewer.
PrintWindowWait	Sets the wait period before printing an InTouch window from WindowViewer.
RemoteTagsLogEvents	Determines whether an InTouch application logs remote referenced tag alarms and events.
RemoteTagsNoIOEvents	Determines whether an InTouch application logs remote referenced tag alarms.
ScaleForResolution	Determines whether InTouch application windows are automatically resized when changing nodes that have different screen resolutions.
ViewLicenseRetryCount	Determines the times WindowViewer will attempt to acquire the license in the background, during Startup and when no license is available.
WindowNameWithSpecialCharacters	If the parameter is set to 1, then new windows can be created with special characters in the window name.

Set custom logging properties

You can add a set of parameters to the INTOUCH.ini file that specify how tag values are saved to the InTouch historical log file. The values assigned to these parameters determine logging frequency and if the values of remote referenced tags are logged.

Set logging frequency

The InTouch HMI writes entries to the historical log file based upon two conditions:

- The InTouch HMI writes an immediate log entry whenever a tag value changes by an engineering unit value greater than its log deadband value.
- The InTouch HMI writes the current values of all logged tags at a fixed interval. The default fixed interval is 60 minutes.

Add two parameters to the INTOUCH.ini file to change the interval.

- ForceLogging

  ForceLogging specifies the length of the fixed logging interval in minutes. ForceLogging can be set to a value from 5 to 120. The default is ForceLogging=60.
- ForceLogCurrentValue

  ForceLogCurrentValue forces the InTouch HMI to write log entries for all logged tags even if the current values are less than or equal to their log deadband ranges. The default is ForceLogCurrentValue=0.

In the following example, current tag values are written to the Historical Log file at 15 minute intervals or when the value of the tag changes:

```
ForceLogging=15
ForceLogCurrentValue=1
```

Log remote referenced tags

By default, remote referenced tags are not logged to the events log file. To log remote referenced tags, you must enable event logging and then add the RemoteTagsLogEvents parameter to the INTOUCH.ini file.

```
RemoteTagsLogEvents=1
```

To exclude I/O tags from being logged, add the RemoteTagsNoIOEvents parameter to the INTOUCH.ini file. The RemoteTagsNoIOEvents parameter applies only if the RemoteTagsLogEvents parameter is set to 1.

```
RemoteTagsNoIOEvents=1
```

Disable WindowMaker shortcut menus

By default, WindowMaker shows an shortcut menu when you right-click with your mouse over the selected object. If you prefer to develop your application using the same mouse behavior as earlier versions of the InTouch HMI, you can turn off WindowMaker's right-click behavior by setting the oldrightmousebehavior parameter to 1 in the INTOUCH.ini file.

```
oldrightmousebehavior=1
```

Set custom WindowViewer properties

You can add a set of INTOUCH.ini file parameters that set the behavior of WindowViewer to:

- Handle script looping.

- Scale InTouch windows for different screen resolutions.
- Set a waiting period to print windows or screens.
- Log run time changes to an alarm comment.
- Set the drawing mode of a 16-Pen Trend.
- Resize the numeric keypad.
- Resizing input fields of analog and string user input links.

Add a script loop timer

By default, a FOR-NEXT loop within an InTouch script must complete within five seconds. WindowViewer stops the script automatically if the FOR-NEXT loop processing has not finished by the time-out limit. This time-out limit prevents an infinite loop caused by a scripting error.

Occasionally, you may need to write a script in which the FOR-NEXT loop code processing exceeds the five second time-out limit. You can change the length of the time-out limit by adding the LoopTimeout parameter to your INTOUCH.ini file.

In this example, loop processing continues for a maximum of 20 seconds:

```
LoopTimeout=20
```

Scale InTouch windows to different screen resolutions

You can add a parameter to the INTOUCH.ini file to maintain the current resolution of InTouch windows when you migrate the application to other nodes running different screen resolutions.

The ScaleForResolution parameter value determines if application windows are automatically scaled by WindowMaker after the display resolution changes on the computer running WindowViewer. The ScaleForResolution parameter does not affect the resolution of WindowViewer dialog boxes. Resolution conversion is enabled when the ScaleForResolution parameter is set to 1.

```
ScaleForResolution=1
```

Set the length of the print waiting period

When you select a window or screen to print, WindowViewer loads the selected window or screen into memory. WindowViewer then waits 10 seconds to allow all DDE variables shown in the window or screen to be updated. After the waiting period ends, WindowViewer sends the window or screen to the printer.

The WindowViewer print waiting period can be changed by adding the PrintWindowWait or PrintScreenWait parameters to the INTOUCH.ini file. The wait period for either parameter is expressed in milliseconds.

```
PrintWindowWait=15000
PrintScreenWait=20000
```

Logging alarm comments

Operators can add a comment when acknowledging an alarm. To write run time changes to the Alarm Comment field in the tag database, add the following line to the INTOUCH.ini file for the current application.

```
CommentRetentive=1
```

Set the drawing mode of a 16-Pen Trend

You can select the line drawing mode of a 16-Pen Trend based on the value of the 16PenTrendDrawMode parameter.

- Averaging mode: `16PenTrendDrawMode=0`

  Because of the time range and the buffer size of the 16-Pen Trend, each pixel on the trend can represent several seconds' worth of data. Each interval can contain several samples with different values. As a result, the trend's data point can appear as a vertical line between the maximum and the minimum values observed within the interval.

  After the minimum to maximum vertical line is drawn, the trend pen moves to the calculated average value for the interval. The next interval begins by drawing the line from the average value to the next interval on the trend. The vertical minimum to maximum line is drawn and the pen rests at the average value calculated for the interval. This process repeats for each sampling interval.

  Averaging is the default drawing mode of a 16-Pen Trend if the 16PenTrendDrawMode is not specified in the INTOUCH.ini file.
- Min-Max mode: `16PenTrendDrawMode=1`

  In the Min-Max drawing mode the trend line is drawn by directly connecting the endpoints of each data collection interval.

Resize a numeric keypad

You can add a parameter to the INTOUCH.ini file that determines whether an InTouch application's numeric keypad can be resized or not. Increasing the size of the keypad at higher screen resolutions (1280 x 1024) keeps the text appearing on the keypad legible. But, you may have applications with limited screen space that set practical limits on the size of the keypad.

You can add the NoKeyboardResize parameter to the INTOUCH.ini file. By default the parameter is not included. Its default value is:

```
NoKeyboardResize=0
```

The default value permits the numeric keypad to be resized according to the screen resolution.

The alternative value you can assign to the parameter is:

```
NoKeyboardResize=1
```

In this case, the keypad does not resize based on screen resolution and the numeric keypad size remains fixed.

Resize the input fields of analog and string user input links

You can add the Resizable InputLink parameter to the INTOUCH.ini file to resize the input box of the Analog or String user input links with your mouse. The Resizable InputLink parameter must be set to a non-zero value.

After the Input field is resized the first time, WindowViewer adds the Resizable InputLink Width and Resizable InputLink Height parameters to the INTOUCH.ini file. These parameters specify the width and height of Input boxes in pixels.

Example:

```
Resizable InputLink = 1
```

```
Resizable InputLink Width=300
Resizable InputLink Height=50
```

Also, you can edit the INTOUCH.ini file to manually modify the values assigned to these parameters.

Resolve stuck application button or displayed value problems

A parameter can be added to the InTouch.ini file to resolve problems in which an InTouch application button is stuck in the down position or a displayed value does not change. The button and the value do not respond to repeated mouse clicks.

Possible causes of this problem can be an OnKeyUp script that does not run because a graphic element with an OnKeyDown script hides the window. Also, the stuck button problem can be caused when there are two scripts, OnKeyDown to set a bit and OnKeyUp to clear the bit. The operator clicks the button, but the window containing the button closes before the mouse is released.

Solve these problems with the following actions

- Insert the `UseLegacyOnKeyUp=1` parameter in the Intouch.ini file.
- Select the **Use In-Memory Window Cache** check box in the WindowViewer **Properties** dialog box.

Diagnose

Troubleshoot QuickScripts

You can troubleshoot QuickScripts by using the Log Viewer to display run time values of tagnames.

Log messages to the Log Viewer

Use the Log Viewer to help you debug QuickScripts. The Operations Control Log viewer is located in the System Management Console and is installed when you install the InTouch HMI.

One way to debug QuickScripts

1. Set check points in the QuickScript to log values to the Log Viewer.
2. Open Operations Control Log viewer to view the values.

Another way to debug QuickScripts is to create a Key Script that logs tag values to the Log Viewer.

Set check points in a QuickScript

1. Open the QuickScript that you suspect is causing errors.
2. Locate the line where you want to set a check point.
3. Insert one of the following snippets of code after that line:
   - LogMessage(messagetag);
     In this script, messagetag is the name of a message tagname whose value you want to log.
   - LogMessage(StringFromIntg(inttag,10));
     In this script, inttag is the name of an integer tagname whose value you want to log.
   - LogMessage(Text(realtag,"#.#######"));
     In this script, realtag is the name of a real tagname whose value you want to log.
   - LogMessage(DText(disctag,"TRUE","FALSE"));
     In this script, disctag is the name of a discrete tagname whose value you want to log.
   - Log more information to the LogViewer at a checkpoint, such as an identifier and/or tagname. For example,

     ```
     LogMessage("DEBUG tag:"+ind.name+" value:"+Text(ind,"#.####"));
     ```

     In this script, `ind` could be an analog indirect tag.

LogMessage() function

Writes a user-defined message to the Log Viewer.

Category

misc

Syntax

```
LogMessage("Message_Tag");
```

Parameter

Message_Tag

String to log to the Log Viewer. Actual string or message tagname.

Remarks

This is a very powerful function for troubleshooting InTouch scripting. By strategically placing LogMessage() functions in your scripts, you can determine the order of QuickScript execution, the performance of scripts, and identify the value of tags both before they are changed and after they have been affected by the QuickScript. Each message posted to the Log Viewer is stamped with the exact date and time.

Important: The percent (%) character formats diagnostic messages that appear in the Log Viewer while debugging scripts. WindowViewer can stop responding if the % character appears in a log string or a function parameter. To eliminate errors caused by %, use two %% characters.

Example(s)

```
LogMessage("Report Script is Running");
```

The above statement would print the following to the Log Viewer:

94/01/14 15:21:14 WWSCRIPT Message:Report Script is Running.

```
LogMessage("The Value of MyTag is " + Text(MyTag, "#"));
MyTag = MyTag + 10;
LogMessage("The Value of MyTag is " + Text(MyTag, "#"));
```

View Log Viewer messages

The Log Viewer is located in the Operations Control Management Console and is installed when you install the InTouch HMI.

View the logged values in Log Viewer

1. Select **Start**, point to **Programs**, point to **AVEVA**, and then select **System Platform Management Console**. The System Management Console appears.
2. In the left pane expand **Log Viewer**, expand **Default Group**, and then select **Local**. The Log Viewer messages appear in the details pane.
3. Locate the logged values from the LogMessage() function.

   Note: If you are debugging the script on a remote InTouch HMI node, you must add the Node name to the

Node Group in Log Viewer and view the Log Viewer messages of that node.

Understand error messages

Tag Viewer displays error messages when it fails to execute a request. The message is displayed in a dialog box, which contains the description of the error.

Main window

Feature	Description	Reason
Quick Search	The searched tagname/alarm group was not found.	System could not find a matching tagname/alarm group.
Watch window	The maximum number of references that can be added to watch window is 2000.	You have tried to add more than 2000 items to a watch window.
	The maximum number of watch windows is 50.	You have tried to add more than 50 watch windows.
	<File1> is not a valid watch list file.	You have specified an invalid file name while opening a watch window.
	<File1> already exists. Do you want to replace it?	You have specified an existing file name while saving a watch window.
	The following references does not exist in the current InTouch application and they were not added to the watch window: <List of references>	Some of the references that you loaded in the watch list are invalid or do not exist in the current InTouch application.

Add tag reference

Feature	Description	Reason
Add Tag Reference	Cannot find <f> in <abcd>.	You have entered a string <f> in the **Substitute box** and this string is not part of the string <abcd> in Tag Reference.
	The <From> value cannot be greater than <To> value.	You have entered a number in the **To** box that is less than the number in the **From** box.
	Please enter a number for <From>. Or, Please enter a number for <To>.	You have entered a string in the **Substitute box**, but the From box or To box is empty.
	The following references does not exist in the current InTouch application and they were not added to the watch window: <List of references>	You have entered one invalid reference or multiple references and some of the references are invalid.
	The maximum number of references that can be added to watch window is 2000.	You have tried to add more than 2000 items to a watch window.

Modify tag value

Feature	Description	Reason
Modify Integer Value	<1a> is not a valid numeric value.	You have entered an invalid entry <1a> in the **Value** box and selected **Apply**.
	The input value is out of valid range for integer data type.	You have entered a number that is outside the permitted range (-2147483648 and 2147483647).
Modify Real Value	The input value is out of valid range for real data type.	You have entered a value that is outside the permitted range (-3.402823466e38 and 3.402823466e38).
Tag Reference dialog box	The tag reference <InvalidReference> does not exist	You have entered a value <InvalidReference> in the **Tag**

Feature	Description	Reason
	in this application.	**Reference** dialog box and selected **OK**.
	The data to be set was of the wrong data type.	The reference was a read-only tag, and you tried to modify the value. Or, You have entered a value that is not valid for the selected type of the tag reference. For example, you have entered a string type value while modifying an integer type tag.

Rename watch window

Feature	Description	Reason
Watch Window	Watch window <Watch List 1> already exists.	You have entered an existing name while renaming a watch window.
	"ReadonlyReference" is read only and cannot be modified.	You have added a read-only tag in the watch window and double-clicked the tag to modify the value.

Manage the InTouch Web Client

This section includes:

InTouch Web Client error handling

InTouch web client provides a uniform error page for displaying expected errors. The error page will be displayed for the following scenarios:

- **No Application** - The web client could not detect the necessary supporting files for the last application run on WindowViewer. Open the required application in WindowMaker.

- **Permission Denied** - The user currently logged in does not have adequate permission to view the web page. Add the user to the aaInTouchUsers or aaInTouchRWUsers user group.
- **No License Available** - A valid license status is not obtained for the current web session. Confirm that a valid web client license is available.
- **Fast Switch Mode** - The web client application is in fast switch mode, but the user tries to access the web client using an URL other than http://localhost/InTouch.

Browser and mobile support

Browser Support

Note: For InTouch Web Client to function correctly, it is recommended to always install the latest version of browser or upgrade your browser to any latest version available.

InTouch Web Client has been tested with the following web browsers:

- Google Chrome
  - Chrome on desktop version 64.0.3282.186 or later
  - Chrome on Android Phone & Pad version 64.0.3282.137 or later
- Microsoft Internet Explorer 11.0.9600.18861 or later
- Microsoft Edge
  - Microsoft Edge on Windows 10 version 41.16299.15.0 or later
  - Microsoft Edge on Surface version 41.16299.248.0 or later
- Apple Safari for iOS version 11.2.6

You will receive an error message if the web browser is not supported.

Mobile Device Support

InTouch Web Client supports the following mobile devices and their default browsers:

- Apple iPhone
- Apple iPad
- Android Phone & Tablet
- Microsoft Surface

Troubleshoot viewing graphics in a web browser

If the InTouch Web Client is not displayed, verify the following settings

- Ensure that the *InTouch Web Service* is running on the node.
- Ensure the user account accessing the InTouch Web page is part of one of the user groups.
- If IIS is running on the same machine, it may conflict with the InTouch Web Server.
  - If IIS is running and it is not being used, stop and disable the service. If IIS must run, configure to use a port other than Port 80 to avoid conflict.

If you get a *HTTP Error 404.0 - Not Found* error when you attempt to launch the InTouch Web Client

- Ensure that Web Client is enabled on the selected node. Contact your administrator.

If any graphic primitives or animations are not visible or working as expected

- Check the list of unsupported features to see if it is not yet supported by the Web Client.
- Run the Web Client locally via http://localhost/InTouch, browse to the symbol, and look for the compatibility icon to view a list of the unsupported items in the graphic.
- If you use a primitive or animation in a graphic that is not supported, it will be removed and a message recorded in the Logger. The logged message contains the path to a generated CSV report file. You can view the Logger message, and navigate to the report for more information.

If the selected graphic is displayed but no data is shown or animations are not updated

- Ensure InTouch WindowViewer is running, if the graphics point to InTouch tags.
- Ensure Application Server is running, if the graphics point to Application Object references.
- Ensure the InTouch iData service is running.

Invalid certificate error

Error: Certificate Error: Your connection to this site is not secure.

When accessing the InTouch Web Client, the browser presents this error when a valid certificate is not available on the client.

Solution

1. Connect to the Server machine and launch the System Platform **Configurator**.
2. From the System Management Server page, select **Details**.

   The **Certificate** Dialog Box appears.
3. On the **Details** tab, select **Copy to File**.
4. Follow the instructions of the **Certificate Export Wizard** and save the file.
5. Copy the certificate file to the client machine.
6. In the client machine, launch **Microsoft Management Console** (mmc).
7. Import the certificate.
8. After importing the certificate, launch the browser.

Manage

AVEVA Operations Control connected experience

About Operations Control connected experience

Operations Control connected experience option enables Single Sign-On (SSO) experience across all Operations Control products for that node with CONNECT cloud capabilities. The Operations Control connected experience requires a CONNECT account with a valid Operations Control subscription and user management.

- Unified User Management: Enabling Operations Control connected experience on a node modifies the behavior of all participating Operations Control products on this node to require log in authentication with CONNECT when launching the first product on the node. Products on the node that are launched after that will authenticate using CONNECT and will enable Single Sign-On (SSO) for authenticated users. CONNECT-based authorization is the only security mode available under the Operations Control connected experience.
- Compatibility across nodes: The Operations Control connected experience must be enabled on all nodes in your system. Applications previously built on nodes not enabled for the Operations Control connected experience must be reconfigured to function in the Operations Control connected experience environment.

You can disable the Operations Control connected experience at any time, but the Operations Control connected experience must be disabled on all nodes in your system. Applications built under the Operations Control connected experience must be reconfigured to function under a non-Operations Control connected experience environment including both authentication methods and product licensing.

Configure Operations Control connected experience

To operate InTouch and other System Platform products under AVEVA Operations Control, either as a product subscription only, or with Operations Control connected experience, you must configure your installed and licensed products. It is still required to configure a license server for the headless services to work properly.

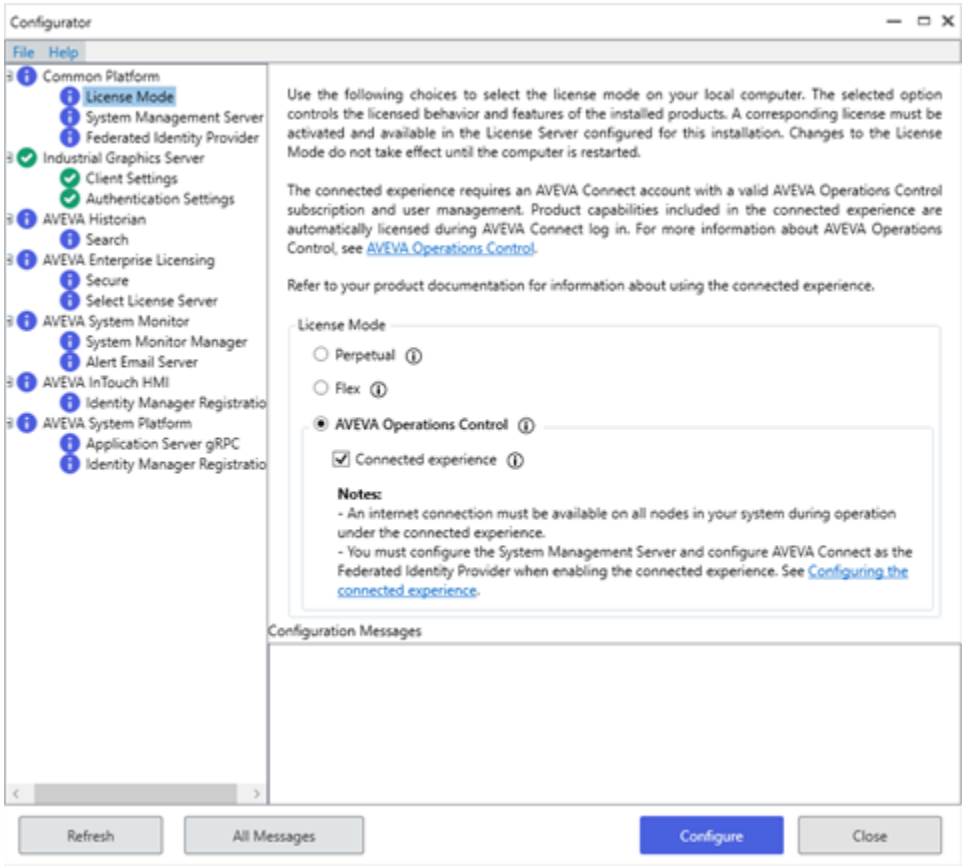
For more information refer to:

- Configurator Help
- AVEVA Identity Manager Help

Configure Operations Control connected experience

1. Open the Configurator from **Start** > **AVEVA** > **Configurator**.
2. Select **License Mode** in the left pane to configure license mode.
   a. Select **AVEVA Operations Control** radio button to subscribe for at least one of two AVEVA Operations Control packages (Edge and Supervisory); that includes unlimited use of all products in the product package for your defined set of users.
   b. Select **connected experience** check box to enable a Single Sign-On (SSO) experience across all

Operations Control products on this node with CONNECT cloud capabilities, available in the products by default.



3. To configure the System Management Server, in the left pane, under **Common Platform**, select **System Management Server**.

Note: If you are prompted for user credentials for the System Management Server, use the following format to enter the user name: DomainName\UserName. The prompt for user credentials may be displayed if you have domain admin privileges but are not an admin on the local machine. You must be a member of the Administrators or aaAdministrators OS group to configure the System Management Server.

   a. To connect to an already existing System Management Server, select the **Connect to an existing System Management Server**. From the list of System Management Server available, select the required System Management Server. Note that the list displays all machines on the network that have been configured to function as System Management Server. In most cases, there is only one System Management Server.

   b. To configure a machine as a Redundant SSO (RSSO) server, under **Connect to an existing System Management Server**, select **Configure this machine as a Redundant SSO Server**.

      OR

   c. To configure the current machine as a System Management Server, select **This machine is the System Management Server** option.

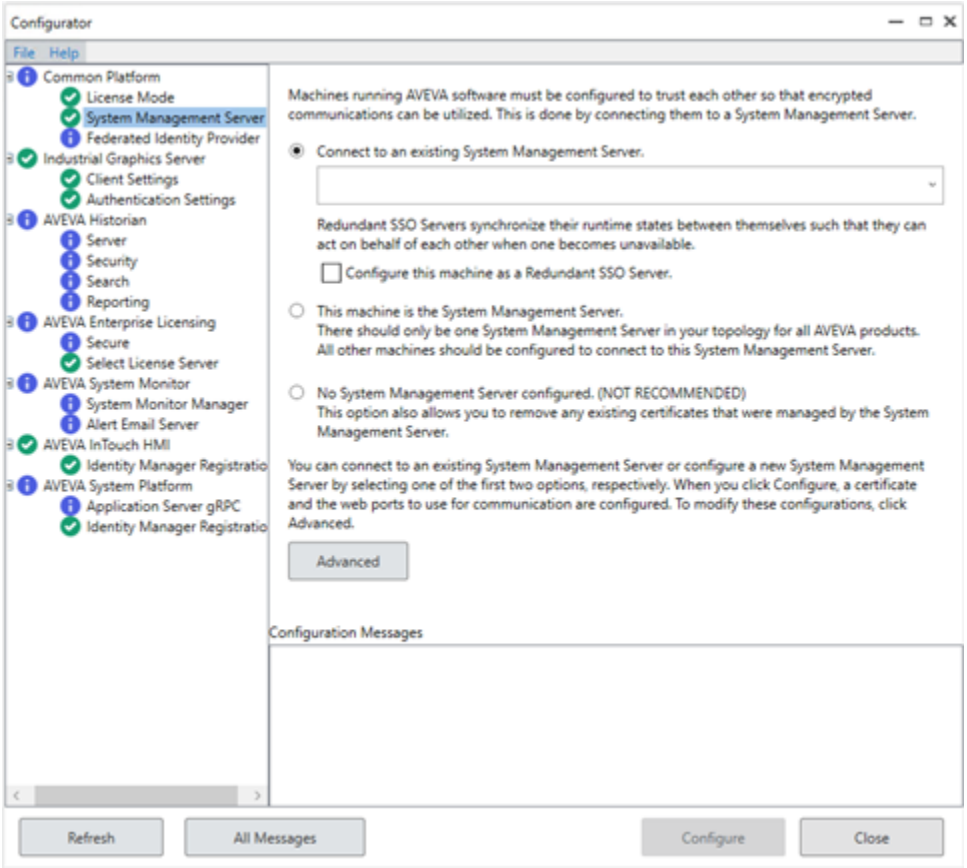
   d. Select **Configure**.

Note:

- **No System Management Server configured** option is not supported for Operations Control connected experience. You must configure a System Management Server to use Operations Control connected experience.

- Only one of the machines in the network is identified and configured as a System Management Server. Machines running AVEVA products can then connect to that single System Management Server to establish trust and configure encrypted communications.

- Configuring Redundant SSO does not configure a separate System Management Server.



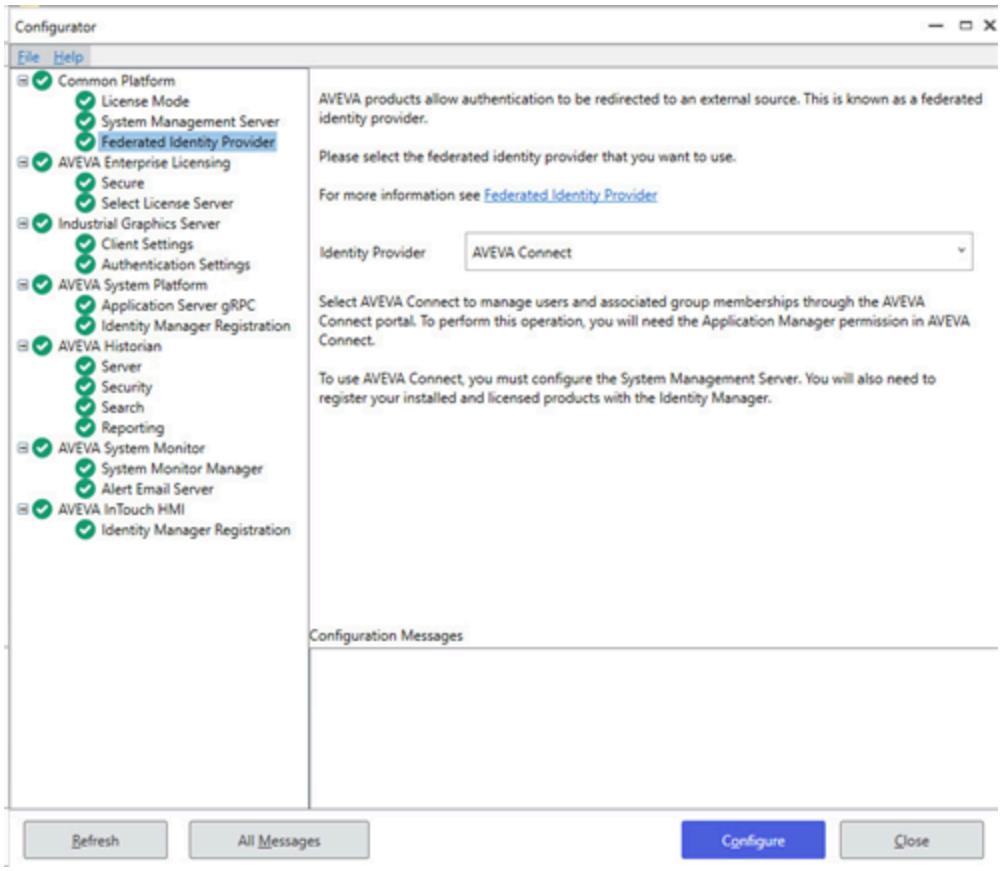
4. To configure Federated Identity Provider, in the left pane, under **Common Platform**, select **Federated Identity Provider**.

   The Identity Manager component in the Platform Common Services (PCS) Framework available on the System Management Server (SMS) and Redundant SSO (RSSO) machines can be configured for "federated" login with CONNECT. This means that a user can enter their email address as registered with a CONNECT account into the Identity Manager login form, at which point they are redirected to CONNECT so they can log in. At present, for Operations Control connected experience, only federation to CONNECT is supported and Azure AD is not supported.

   Note:

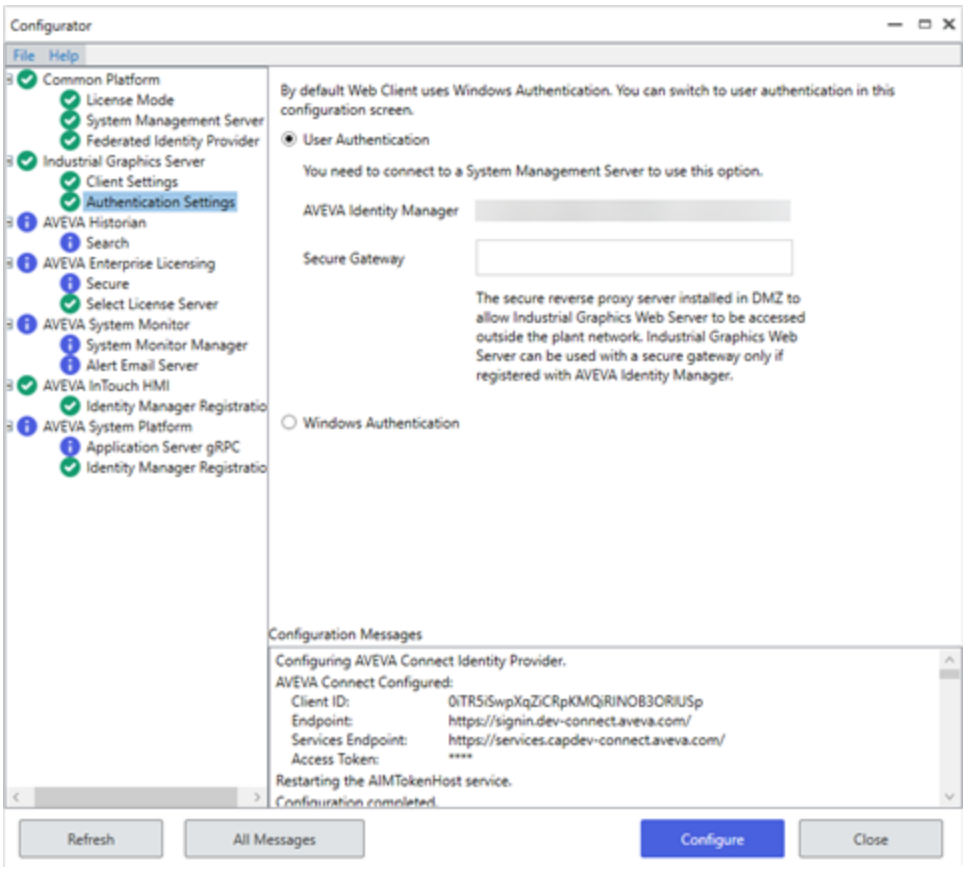
   - For a federated identity connection, you need a valid CONNECT account, and you must be an administrator on that account.

   - This step only applies to the node where This machine is the System Management Server is configured. If a node is connecting to an existing System Management Server, configuring the Federated Identity Provider is not required.



If you have multiple CONNECT accounts, then after the authentication, an Account selection dialog listing multiple CONNECT account name that you are part of will be displayed. Select the account that you want to be federated with and from that point onwards the machine and all the Operations Control products will by authenticated or validated for subscription for that selected account.If you are part of only one CONNECT account, then the account selection dialog will not be displayed.

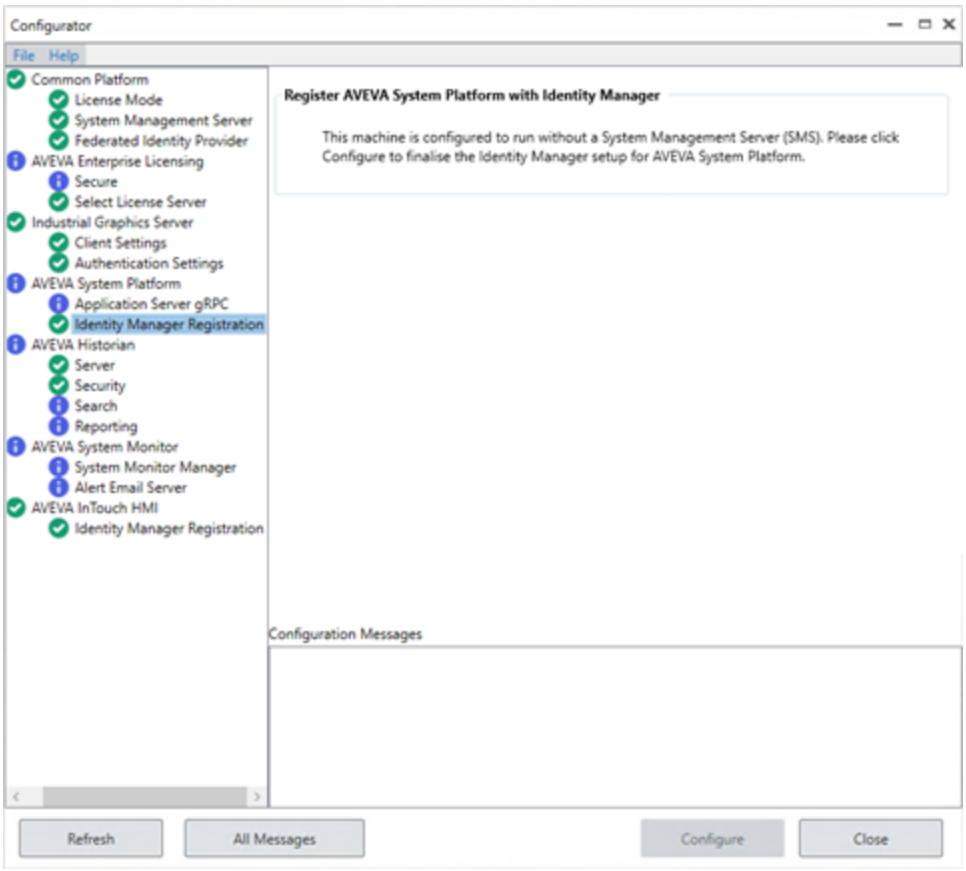
5. To Configure the InTouch Web Client to use User Authentication, in the left pane, under **Industrial Graphics Server** > **Authentication Settings**, select **User Authentication** and then select **Configure**.



6. To register System Platform with AVEVA Identity Manager, in the left pane, under **AVEVA System Platform**, select **Identity Manager Registration**.

   Register AVEVA System Platform with Identity Manager only after you have completed the System Management Server configuration.

   Select **Configure** to register with Identity Manager.



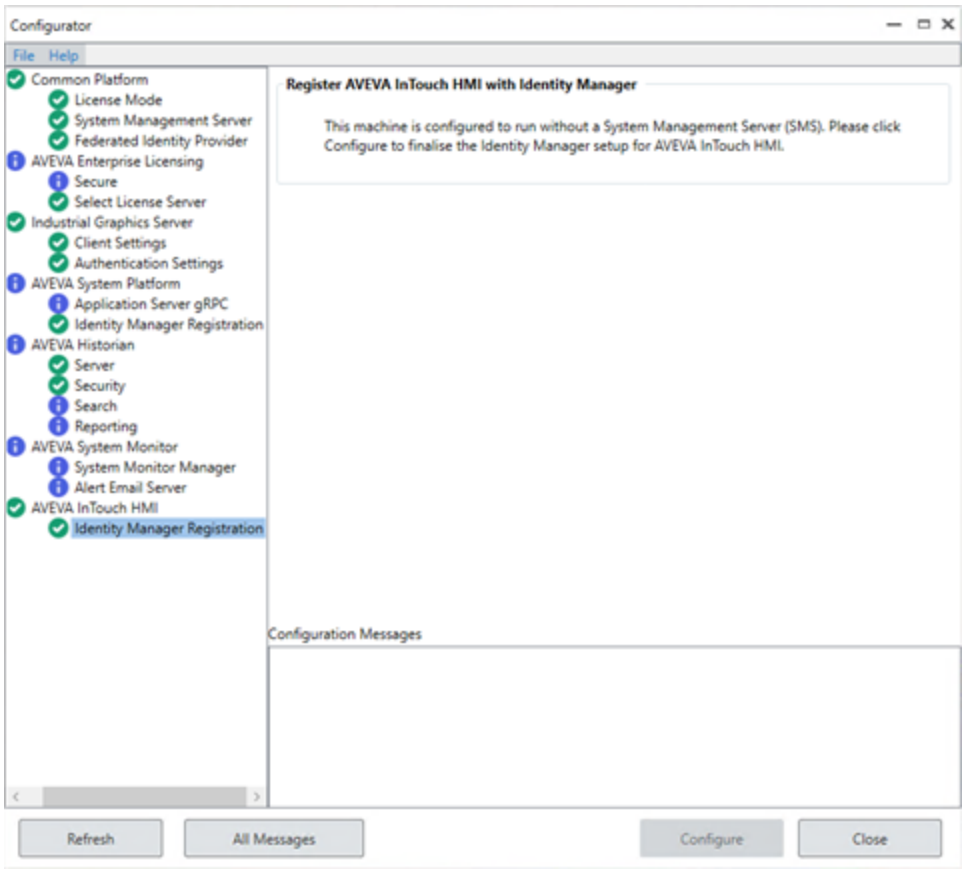
7. To register InTouch with AVEVA Identity Manager, in the left pane, under **AVEVA InTouch HMI**, select **Identity Manager Registration**.

   Register InTouch HMI with Identity Manager only when you have completed the preceding steps.

   - You have enabled AVEVA Operations Control connected experience as your license mode.
   - You have configured the System Management Server.
   - You have selected CONNECT in the Federated Identity Provider.

   Slect **Configure** to register with Identity Manager.

   Note: If you have not configured System Management Server (not recommended), you can still configure InTouch, but cannot use Operations Control connected experience.



Authentication and entitlement

Authentication and entitlement, along with authorization, are security elements that serve to control user access to the AVEVA products that your organization uses.

- *Authentication* refers to verifying the identity of a user, through the use of a set of unique credentials (user name and password).
- *Entitlement* refers to the products that your organization has subscriptions for, and to which the authenticated user has been granted access. A user, even though they have been authenticated, may not be entitled to use a particular product, if the user does not have a valid Operations Control subscription.
- *Authorization* refers to the level of privileges associated with an authenticated user. This is generally done through the assignment of a user to a user group or through federated identity management.

  Federated identity is managed through a *Federated Identity Provider* (FIDP), and allows unified user management and single sign-on for users across the complete portfolio of AVEVA products. This is enabled through the Configurator, and is generally completed at the time of installation but can be reconfigured at any time.

The license mode you select plays an important role in how authentication, entitlement, and authorization are handled within your System Platform environment. See License mode for more information.

When Operations Control connected experience is enabled, each time a user logs in to a System Platform component, such as the IDE, the Historian, an OMI ViewApp, and so on, the log in is saved as an event in the CONNECT audit log.

View the Data Log

Audit logs let authorized users from both your organization and AVEVA view the log of events or operations that occur in your account from the CONNECT portal. The logs display all operations that have taken place in your organization's account during the selected time period. You must be an Account Administrator, or must have the Report Viewer role to view the audit logs.

Note: Data in the audit logs is available from the time when the Audit functionality is available for your account.

To view the audit logs, go to the CONNECT web site. On the site navigation menu, select Audit. The Audit log page is displayed.

The audit logs provide the following information:

- Timestamp (date and time) when the operation was performed.
- Details of the operation, such as username, machine name, and the application name (for example, AVEVA OMI or System Platform IDE. The information that is logged includes which product or products were accessed by which authenticated user, as well as their available entitlement access.
- The audit log data can be filtered by using the **Entitlement Check** function in CONNECT. This function provides all data related to product usage through connected experience.

  Note: Local time is used while displaying the data on the user interface. However, the UTC time is shown in the operation details and when the data is exported.

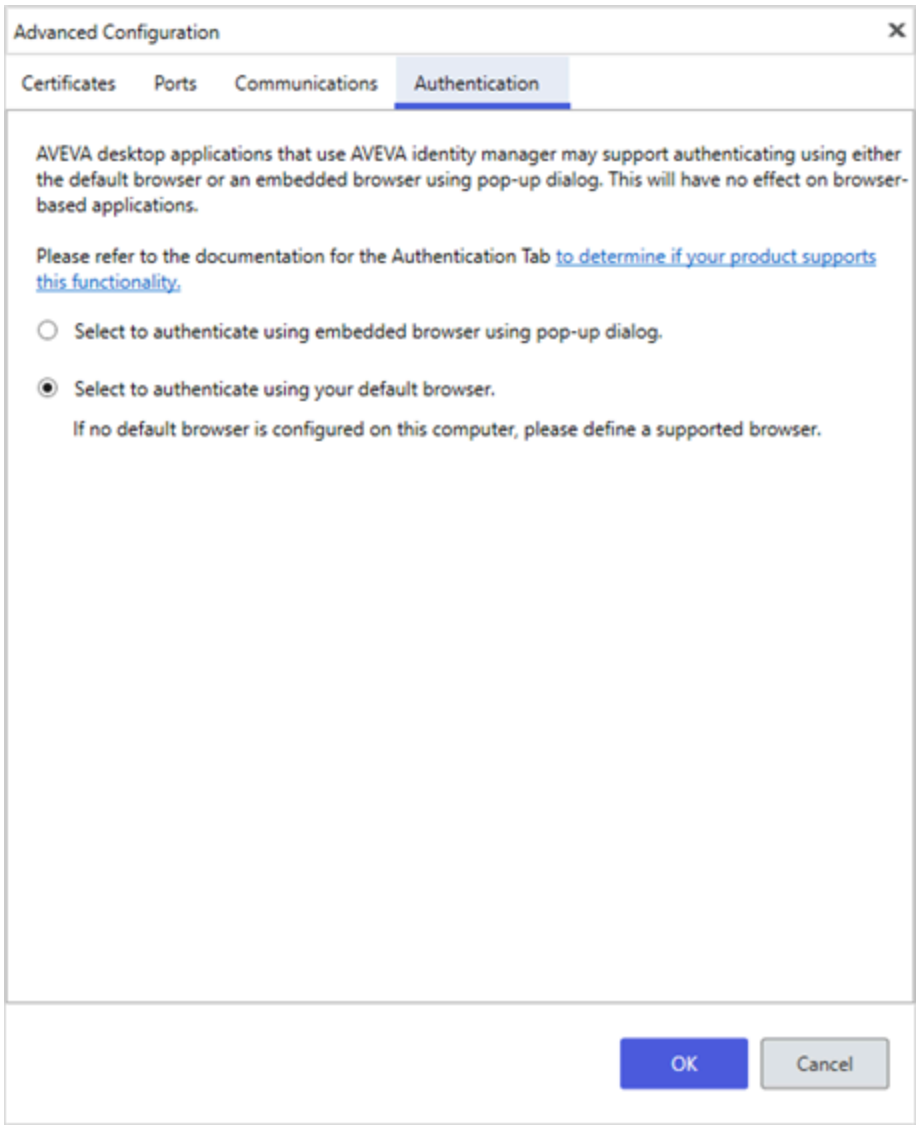
See the CONNECT help for more information about audit logs.

Authenticate using AVEVA Identity Manager

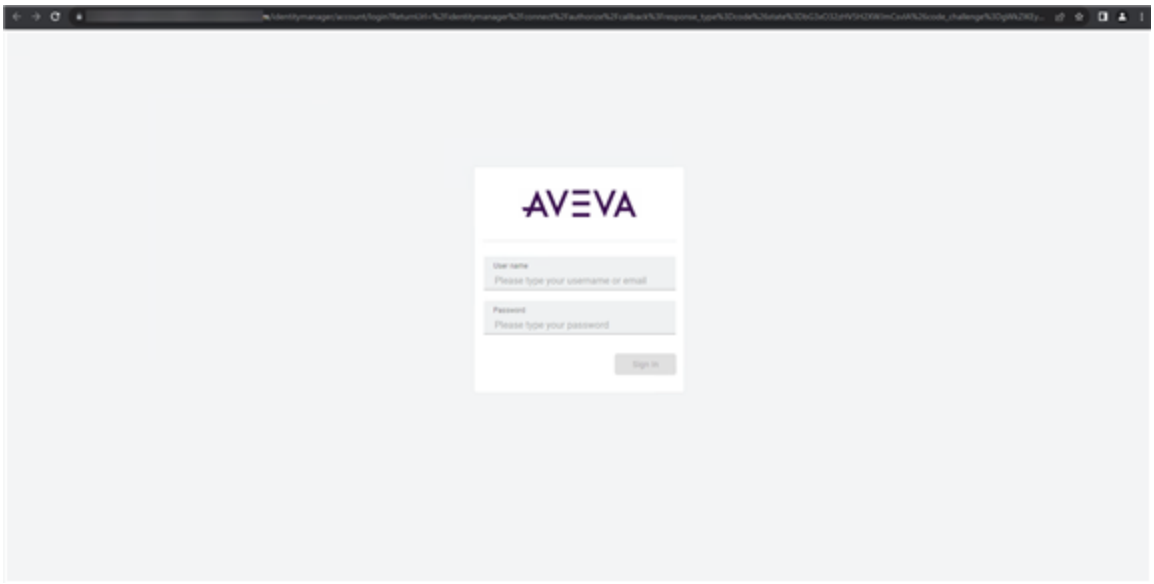
When Operations Control connected experience is enabled in the Configurator, you will be prompted to authenticate with AVEVA Identity Manager, when Application Manager, WindowMaker, or WindowViewer is launched for the first time. In the subsequent launches, you will be authenticated with SSO.

Authenticate using AVEVA Identity Manager on a web browser

1. To open the AVEVA Identity Manager sign in screen on a web browser you have to select the **Select to authenticate using your default browser** option under **Common Platform** > **System Management Server** > **Advanced** > **Authentication** in the Configurator.



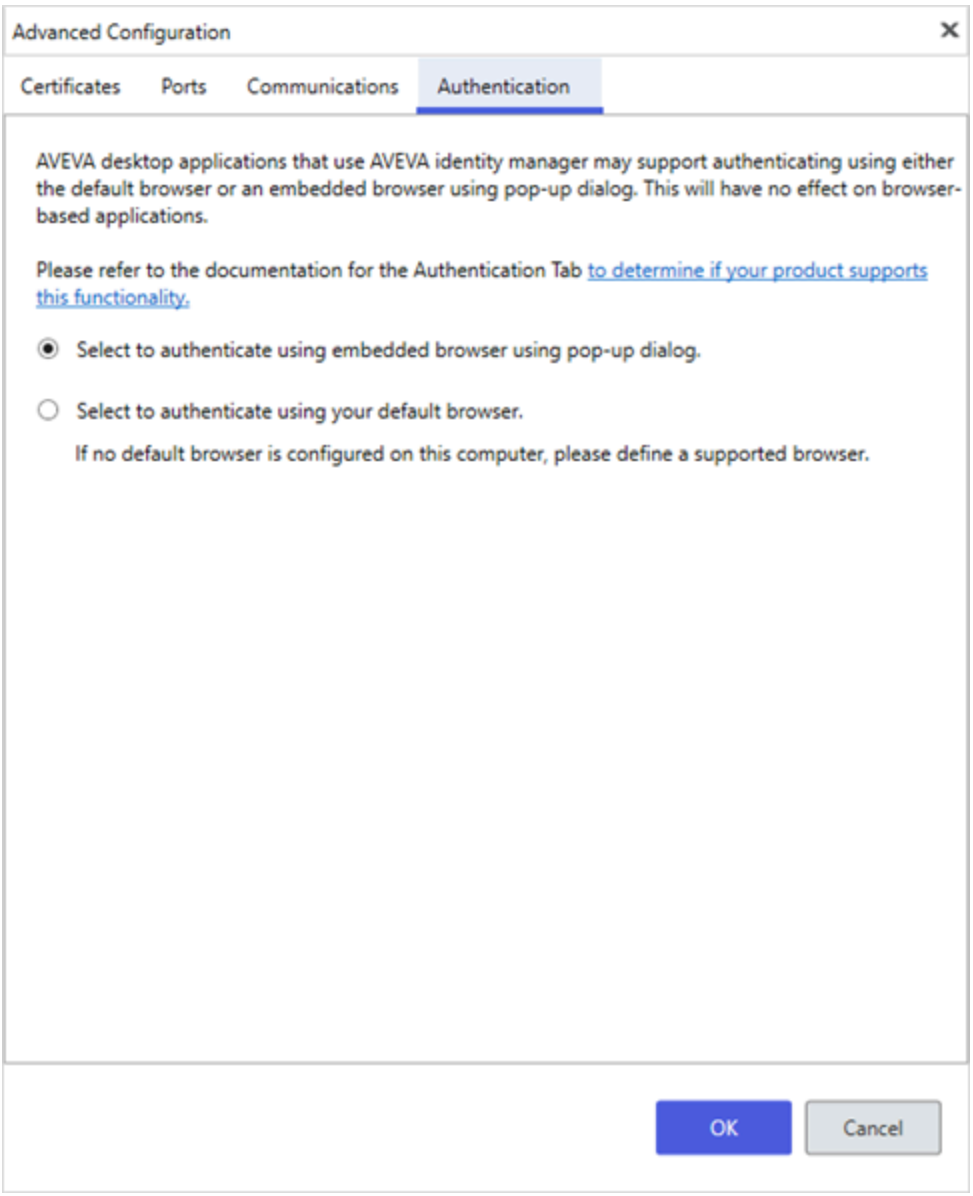
2. When you launch any AVEVA Operation Control application, the AVEVA Identity Manager login page opens in the default browser of your system.
3. In the AVEVA Identity Manager login page, in the **User name** field, enter your email address that is registered with a CONNECT account to login to AVEVA Identity Manager.
4. In the **Password** field, enter the respective **Password**.
5. Select **Sign In**.



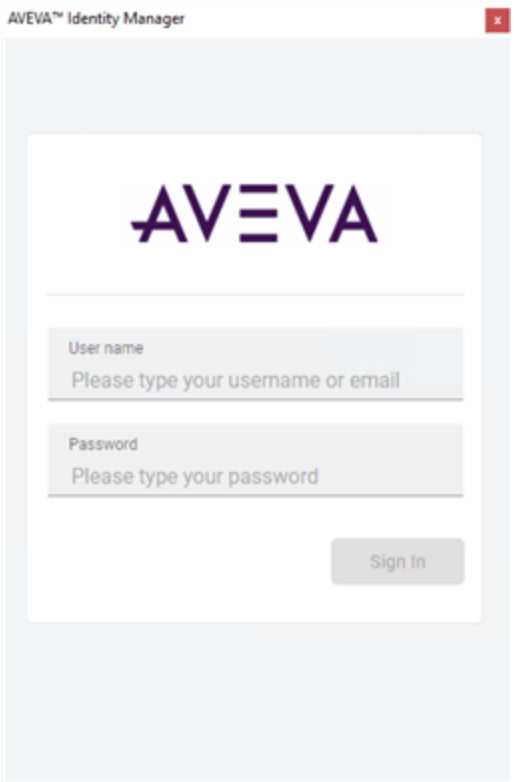
Authenticate using AVEVA Identity Manager on an embedded browser

1. To open the AVEVA Identity Manager sign in screen on a embedded browser you have to select the **Select to authenticate using embedded browser using pop-up dialog** option under **Common Platform** > **System Management Server** > **Advanced** > **Authentication** of the Configurator. By default this option will be selected.

   For InTouch Access Anywhere, it is recommended to use embedded browser option for security reasons.



2. When you launch any AVEVA Operation Control application, the AVEVA Identity Manager login page opens in an embedded browser.
3. In the AVEVA Identity Manager login page, in the **User name** field, enter your email address that is registered with a CONNECT account to login to AVEVA Identity Manager.
4. In the **Password** field, enter the respective **Password**.
5. Select **Sign In**.



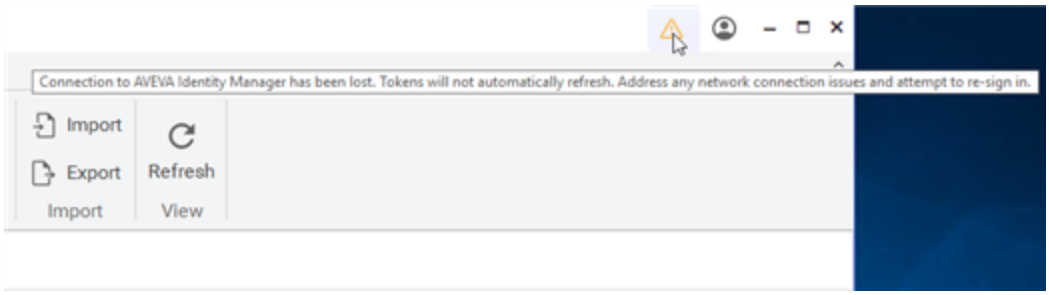
Behavior during connection loss

In case of connection loss:

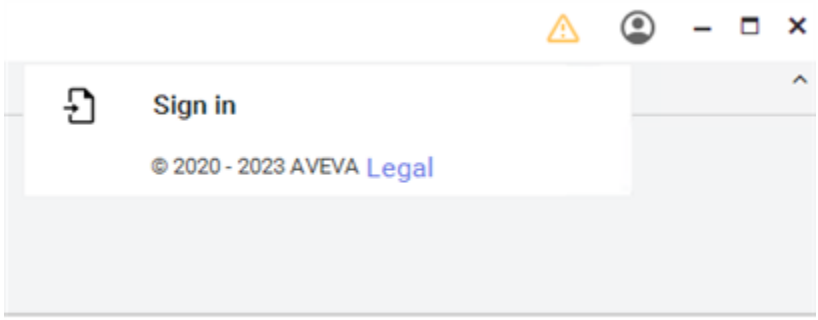
- After authenticating successfully and the application runs successfully, if there is a connection loss between AVEVA Identity Manager and CONNECT, the on-premise capabilities will continue to function.
- After authenticating successfully and the application runs successfully, if there is a connection loss between InTouch applications and AVEVA Identity Manager, Operations Control connected experience capabilities will stop working. You have to sign in again using the **Sign in** option.
- Once you have successfully authenticated and there is a connection loss, and then if you try to open the application in WindowMaker, the WindowMaker runs in demo mode.
- Once you have successfully authenticated and there is a connection loss, and then if you try to open the application in WindowViewer, the WindowViewer runs in Read-Only mode.
- If you have never signed-in to AVEVA Identity Manager, and there is a connection loss, if you try to launch WindowMaker or WindowViewer, then WindowMaker will not run and the WindowViewer will run in the demo mode.

Address the connection loss issue

1. If there is loss of connectivity to CONNECT or AVEVA Identity Manager, in Application Manager and Window Maker, a warning icon will be displayed and will notify you with the connection state. When you hover the cursor over the warning icon a tooltip with the connection loss message appears.



2. Address any network connection issues and select on the profile icon and then select **Sign in** to sign in again.



When connection is restored and the application is re-launched, the tooltip will no longer be visible.

License and entitlements

The licensing for InTouch in Operations Control connected experience is the same for Managed or Standalone applications. There are two kind of subscriptions available:

AVEVA Operations Control Edge subscription

When InTouch application is not configured as ViewApp and the License Mode is set to Operations Control connected experience in the Configurator then this subscription is required to launch WindowViewer.

AVEVA Operations Control Supervisory subscription

When InTouch application is configured as ViewApp and the License Mode is set to Operations Control connected experience in the Configurator, then this subscription is required to launch WindowViewer.

Both subscriptions are unlimited number of sessions and will enable full functionalities of WindowMaker.

View the Data Log

Audit logs enable you to view the log of events or operations in your account. The logs display all operations that have taken place in your account in the selected time period. You must be an Account Administrator, or must have been assigned the Report Viewer role to view the audit logs.

Note: Data in the audit logs is available from the time when the Audit functionality is available for your account.

View the audit logs

- On the site navigation menu, select Audit. The Audit log page is displayed.

The following information is provided by the audit logs:

- Date – The date on which the operation was performed
- Message – Details of the operation such as which user has performed which operation
- Function – The function impacted by the operation

Note: The browser's local time used while displaying the data on the user interface. However, UTC time zone is used while exporting the data to a .csv file.

For more information refer to the CONNECT help.

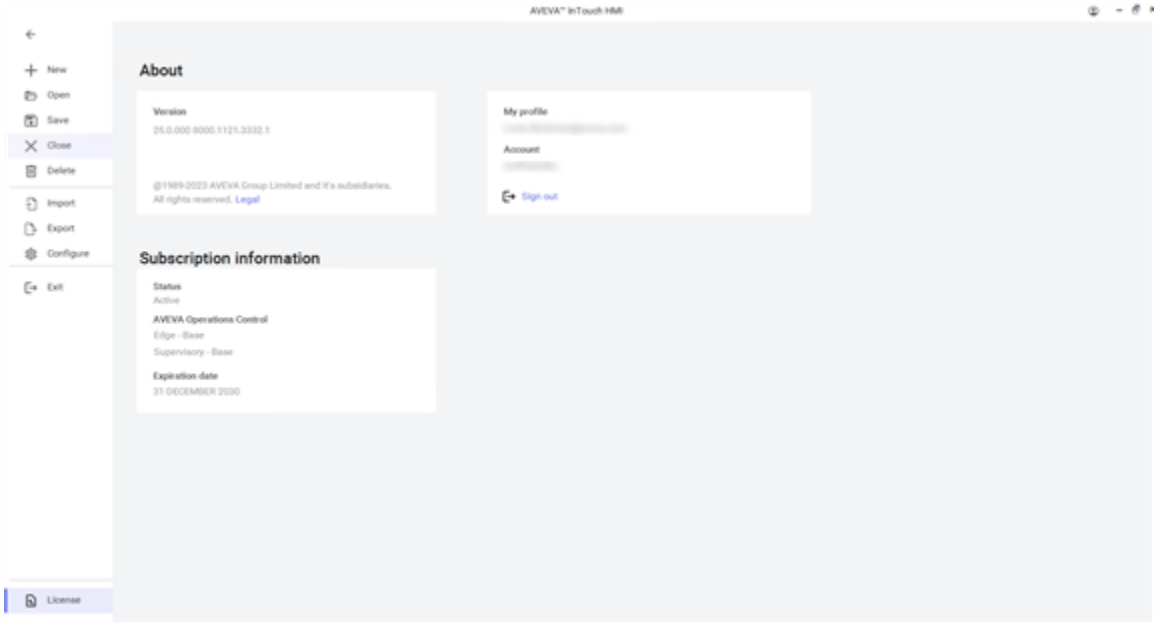
View subscription information

You can view the specific information for the current subscription consumed by WindowMaker or WindowViewer.

View WindowMaker subscription information

1. Open WindowMaker.
2. On the **File** menu, at the bottom-left corner of the screen, select **License**.

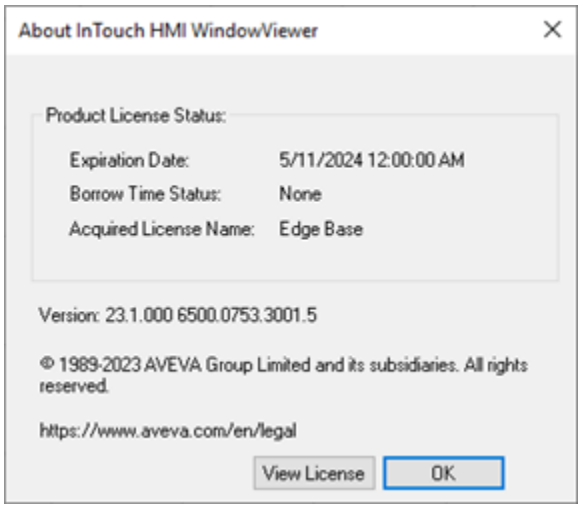
   The **About** screen appears. The **About** screen displays the WindowMaker **Version** and **Subscription information**.



View WindowViewer Subscription information:

1. Open WindowViewer.
2. Select **File** and then **About WindowViewer**.

The **About WindowViewer** dialog box appears.



3. Select **View License** to view the details for the subscription.

Start and run WindowMaker

This section explains about the behavior of WindowMaker in Operations Control connected experience.

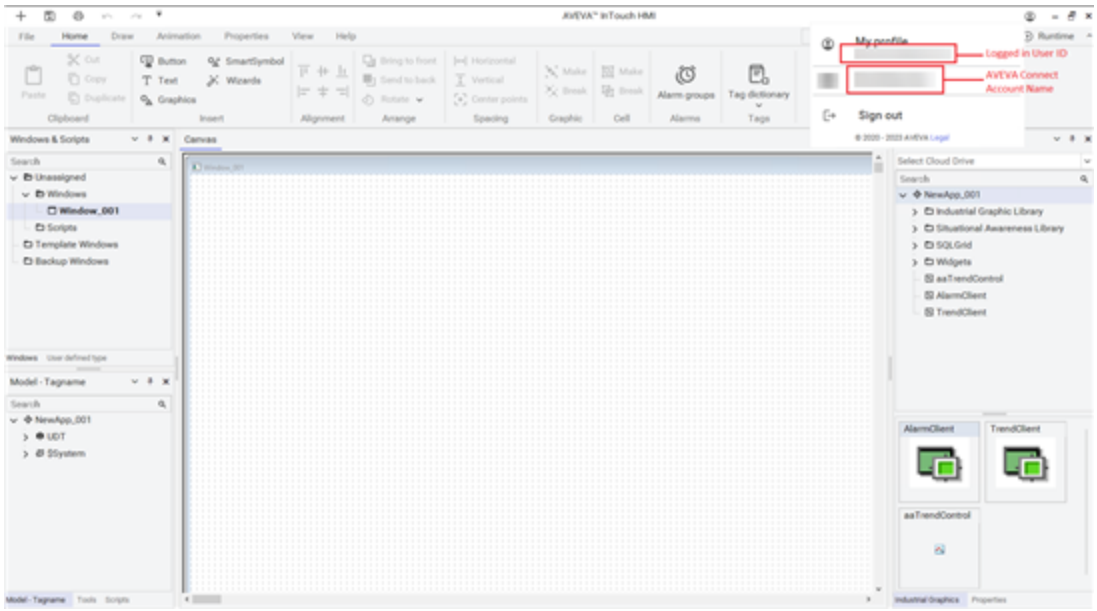
Work with WindowMaker in Operations Control connected experience

1. Launch WindowMaker.

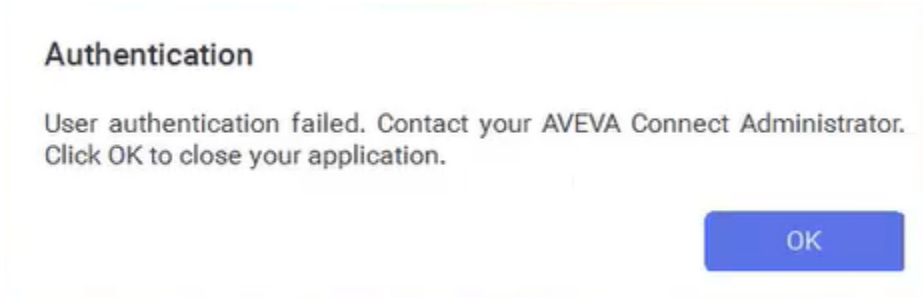
   If you are launching WindowMaker for the first time on this node and no other Operations Control product was launched and authenticated on the same node before, then you are prompted to authenticate with AVEVA Identity Manager on a web browser or embedded browser (based on the choice that you have made while configuring **Authentication** tab in System Management Server of the Configurator). If you have already launched and authenticated WindowMaker on this node or launched and authenticated any other Operations Control product on the same node before, then you will be authenticated using Single Sign-On. For more information refer to Authenticate using AVEVA Identity Manager section.

2. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.

   Once you signed in successfully, if you are entitled to an Edge or Supervisory subscription WindowMaker shows the sign in information in the user profile found on the title bar. My profile will display the logged in user ID and the name of the CONNECT account.



3. If valid credentials are not provided, or authentication is canceled, then you will be prompted with an error message. Contact your CONNECT Administrator for the credentials and try to sign in again. WindowMaker will close once you select **OK**.



   - If you close the embedded browser login page that was launched to authenticate with AVEVA Identity Manager, the authentication failure dialog will be displayed immediately.
   - If you do not provide credentials or close the web browser that was launched to authenticate with AVEVA Identity Manager, the authentication failure dialog will be displayed after 3 minutes.
4. If you do not have Edge or Supervisory subscription or the subscription is expired:
   - WindowMaker will run in demo mode, if the application has 64 or less tags (excluding system tags) and 32 or less windows configured.
   - WindowMaker will exit, if the application has more than 64 tags and 32 or less windows configured.

     Contact your CONNECT Administrator to get a valid subscription. For more information on subscription refer to Flex Subscription.
5. When WindowMaker is running successfully, a check is made for valid AIM and CONNECT refresh tokens.

   If a valid AIM token is received, and AVEVA Identity Manager is disconnected from CONNECT, on-premise functionality will work. You cannot access the CONNECT Industrial Graphics drive. You must sign in AVEVA Identity Manager to get the access to CONNECT Industrial Graphics drive.

Start and run WindowViewer

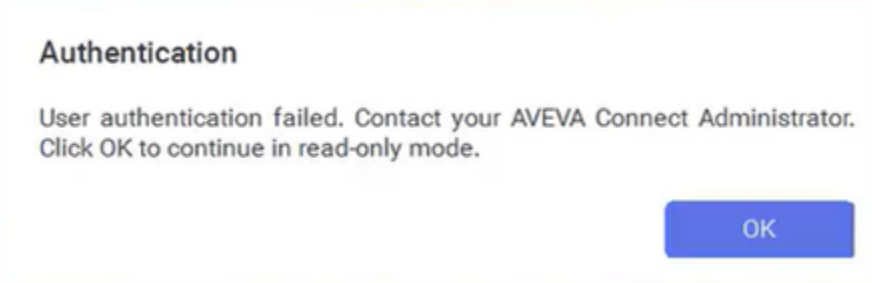
This section explains about the behavior of WindowViewer in Operations Control connected experience.

Work with WindowViewer in Operations Control connected experience

1. Launch WindowViewer.

   If you are launching WindowViewer for the first time on this node and no other Operations Control product was launched and authenticated on the same node before, then you are prompted to authenticate with AVEVA Identity Manager on a web browser or embedded browser (based on the choice that you have made while configuring **Authentication** tab in System Management Server of the Configurator). If you have already launched and authenticated WindowViewer on this node or launched and authenticated any other Operations Control product on the same node before, then you will be authenticated using Single Sign-On. For more information refer to Authenticate using AVEVA Identity Manager section.
2. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.
3. If authentication is not successful and if you had not signed in before, then WindowViewer will launch in demo mode.

4. If authentication is not successful or you have cancelled the authentication, but you have successfully authenticated and logged in at least one time on that node before, then WindowViewer will launch in Read-Only* mode.
   - If you close the embedded browser login page that was launched to authenticate with AVEVA Identity Manager, the authentication failure dialog will be displayed immediately.
   - If you do not provide credentials or close the web browser that was launched to authenticate with AVEVA Identity Manager, the authentication failure dialog will be displayed after 3 minutes.



5. If you get disconnected from AVEVA Identity Manager, the WindowViewer will run in Read-Only* mode.
6. Once you signed in successfully, and if you have Edge or Supervisory subscription, WindowViewer shows the sign in information on the title bar.
7. If you signed in successfully, but do not have Edge or Supervisory subscription, then

- It will open in demo mode, if application has 64 tags or less configured. It will not open in demo mode and will exit the WindowViewer, if application has more than 64 tags configured.

8. When WindowViewer is running successfully, a check is made for a valid AVEVA Identity Manager and CONNECT refresh tokens. If only a valid AVEVA Identity Manager token is received, and AVEVA Identity Manager is disconnected from CONNECT, then:
   - WindowViewer will run in read-only mode. No warning icon will be displayed. You have to use GetTokenConnectionstatus() function script to know the status of connection. You can use the scripting methods to configure graphics that represent user interfaces and their proper status. When there is a connection loss, the configured scripts will run, and the WindowViewer user will view the appropriate user interface associated with the current connection state. For more information on GetTokenConnectionstatus() script refer to the GetTokenConnectionStatus() function section of the AVEVA™ InTouch HMI Application Development Guide (ITBuild.pdf).

     Note:

     - In Operations Control connected experience, the **Configure users** and **Change password** options under **File** > **Configure** > **WindowViewer** > **Window** in WindowMaker will be disabled. These options are not applicable in Operations Control connected experience, and therefore you will not be allowed to configure these options in WindowMaker. They will be unchecked and disabled by default, and they cannot be used in WindowViewer.

     - WindowViewer as a service will not use a subscription from CONNECT, and will use the traditional perpetual license.

***Read-Only mode**

Read-Only refers to the inability of writing to the I/O tags, but you will still be able to write to memory tags. In Read-Only mode you can only view the screen and cannot make any changes. This helps you to run your application in WindowViewer even if the authentication is unsuccessful. This Read-Only mode is an implementation on the WindowViewer, and it is different from the usual commercially supported Read-Only mode. This Read-Only mode does not cater of the Read-Only enabling setting on the InTouch products and is not related to the Read-Only license.

WindowViewer runs in Read-Only mode if you had signed in before atleast once and the subscription has not been expired yet with any of the following conditions:

- authentication is unsuccessful
- you do not want to authenticate
- network failure or timeouts

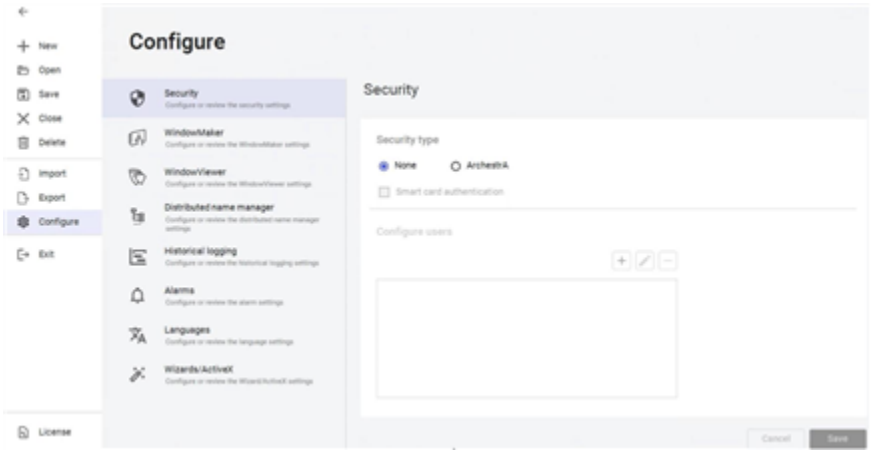
When the contract date for a subscription has ended, WindowViewer will run in demo mode.

Configure application security in Operations Control connected experience

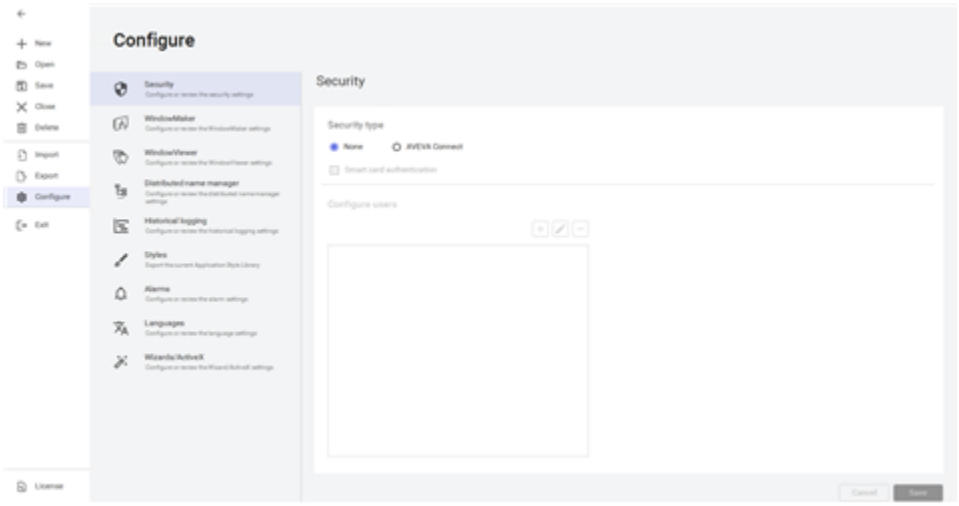
You have to configure any one of the following security types in the WindowMaker, to work in Operations Control connected experience mode:

- Configure security in managed application
  - None
  - ArchestrA (this requires CONNECT based security group configuration under **Authentication mode** tab to

be configured on the System Platform IDE. Refer to Configure security in managed application for more information)



- Configure security in standalone application
  - None
  - AVEVA Connect

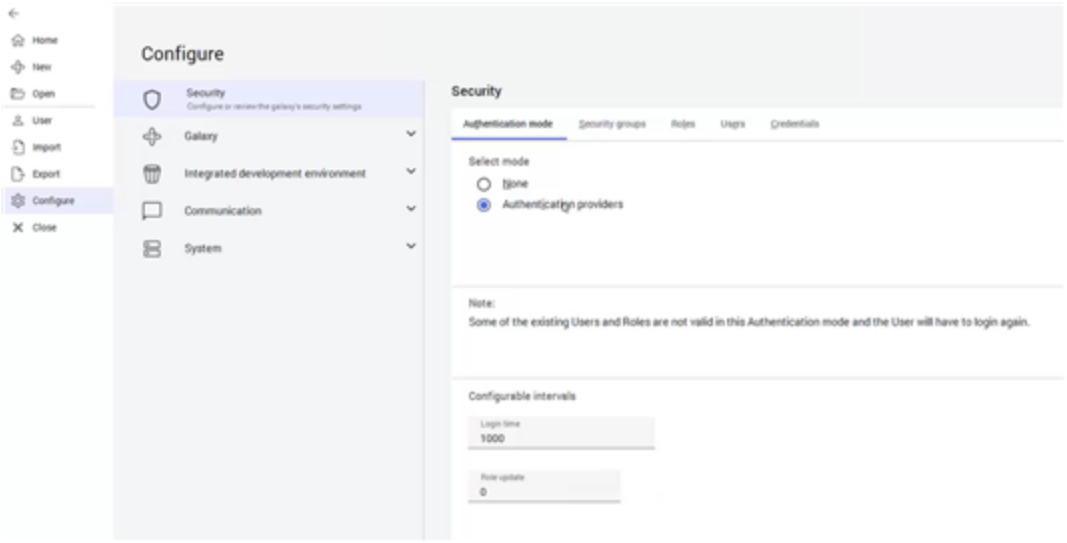


With Operations Control connected experience mode enabled and an incompatible security type configured, if you open WindowViewer or WindowMaker you will be prompted with the message asking you to reconfigure the the security in WindowMaker with any of the above supported security types. The WindowViewer will close on selecting **OK** and in the WindowMaker you will have option to edit the security type.

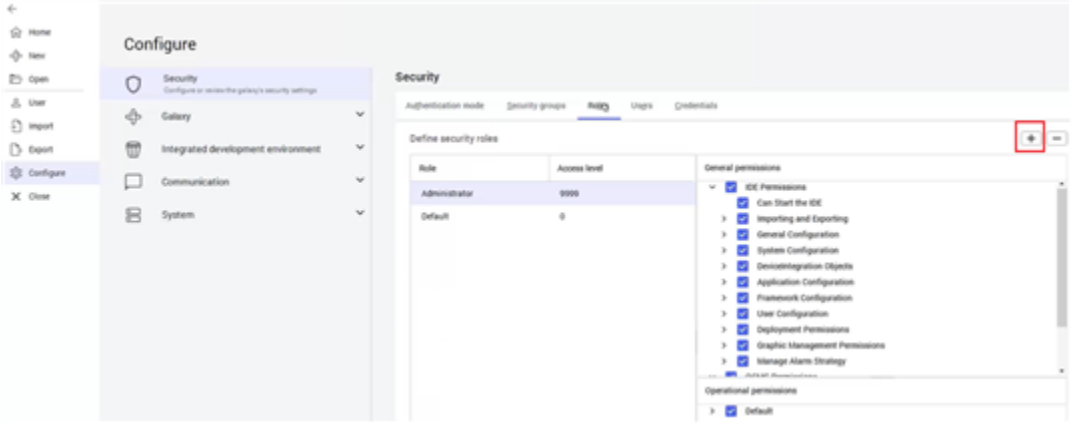
Configure security in managed application

Configure security as ArchestrA in managed application

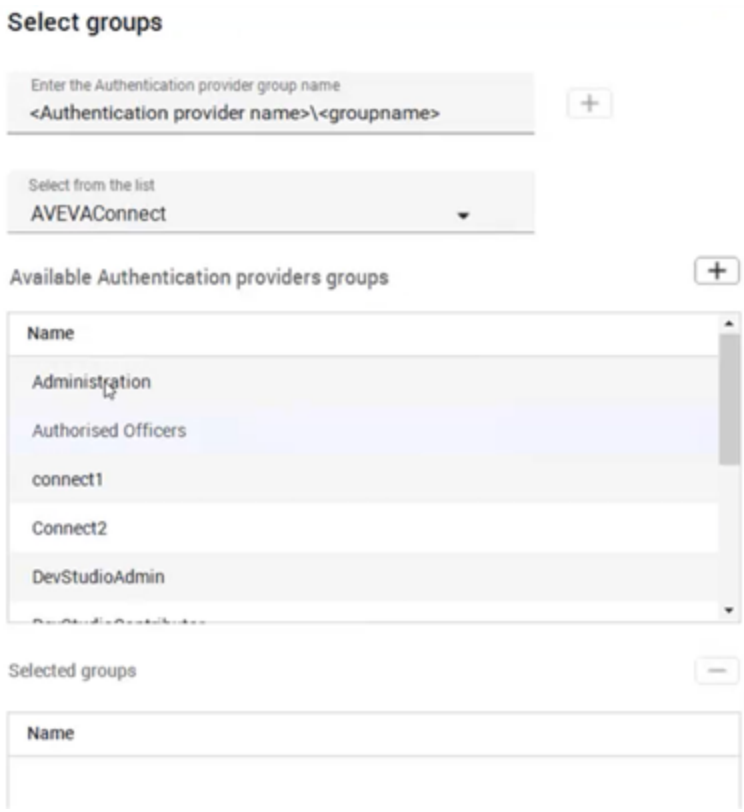
1. Launch System Platform IDE.
2. Go to **Galaxy** > **Configure** > **Security** to open the **Security** page. In the **Authentication mode** tab, select **Authentication providers** option.



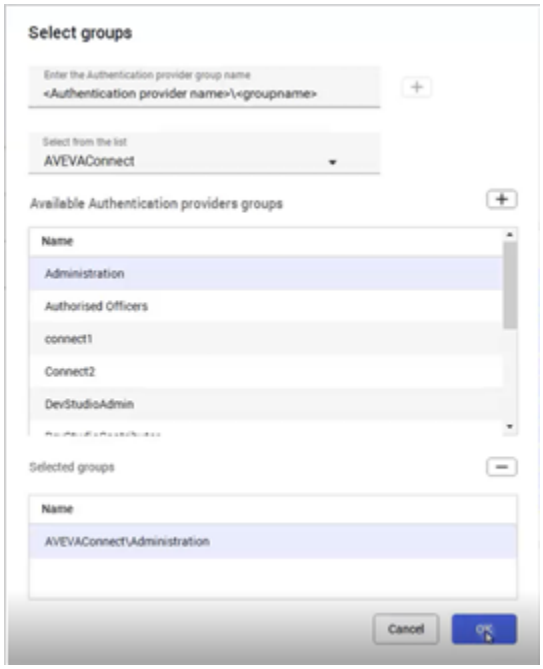
3. To map access levels to CONNECT groups, go to **Configure** > **Security** > **Roles** and select the plus icon.



4. Verify that in the **Select from the list** dropdown, **AVEVA Connect** is selected. A list of available groups in your account from CONNECT is displayed.



5. Add the required groups from the available list and they will be displayed in the Selected groups list and select **OK** when finished.



6. Configure **Access level**, **General permissions**, and **Operational permissions** to groups as applicable and select **Save**.
7. Select ViewApp and launch WindowMaker.
8. In WindowMaker, under **File** > **Configure** > **Security** select **ArchestrA** as the security option and select **Save**.

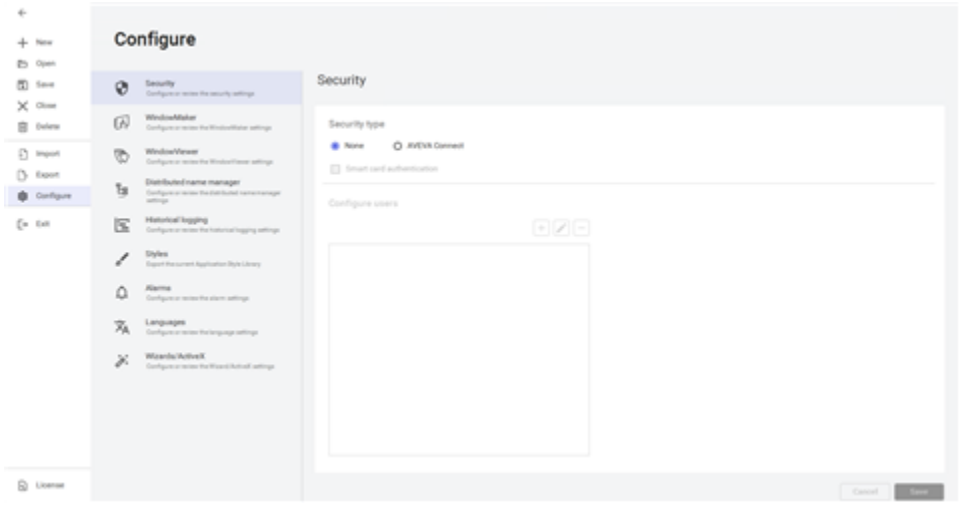
Configure security as None in managed application

1. Launch System Platform IDE.
2. Select ViewApp and launch WindowMaker.
3. Under **File** > **Configure** > **Security** select **None** as the security option and select **Save**.

Configure security in standalone application

Configure security in standalone application

1. In WindowMaker, under **File** > **Configure** > **Security** select **None** or **AVEVA Connect** as the security option and select **Save**.



   With Operations Control connected experience enabled and incompatible security type configured, if you open WindowViewer or WindowMaker you will be prompted with the message asking you to reconfigure the security in WindowMaker with any of the above supported security types. The WindowViewer will close on selecting **OK** and in the WindowMaker you will have option to edit the security type. **InTouch** and **Operating System** security types are not supported in Operation Control connected experience, and you cannot use the CONNECT security type in non-connected experience.
2. To map access levels to groups in standalone InTouch use the script function **AddPermission()**.

   The AddPermission() method accepts only two parameters in Operations Control connected experience:

   - AVEVA Connect Group
   - Access Level

     Script function for AddPermission() in Operations Control connected experience:

     ```
     DiscreteTag=AddPermission("", "AVEVA Connect group", AccessLevel);
     ```

     If a runtime user is a member of multiple groups from CONNECT, the access level will be determined by the group with the highest access level.

Operate in mixed mode

This section explains about the behavior of WindowMaker and WindowViewer if the security type is not

compatible in Operations Control connected experience.

Start WindowMaker with incompatible security type

1. Launch WindowMaker.

   If you are launching WindowMaker for the first time on this node and no other Operations Control product was launched and authenticated on the same node before, then you are prompted to authenticate with AVEVA Identity Manager on a web browser or embedded browser (based on the choice that you have made while configuring **Authentication** tab in System Management Server of the Configurator). If you have already launched and authenticated WindowMaker on this node or launched and authenticated any other Operations Control product on the same node before, then you will be authenticated using Single Sign-On. For more information refer to Authenticate using AVEVA Identity Manager section.
2. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.
3. After successful sign in and with valid entitlement, if the security type is not compatible you will be prompted to configure the security.

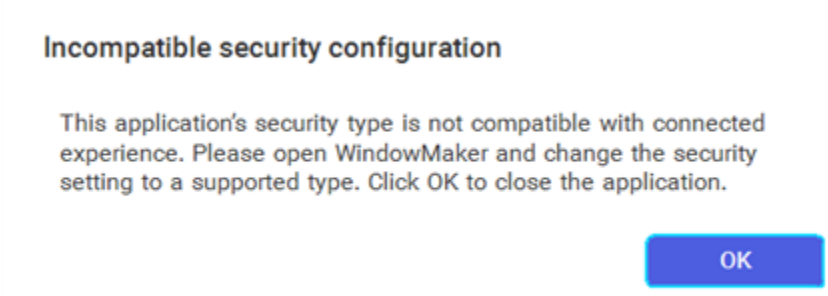
   Note: InTouch and **OS** security types are not supported in Operation Control connected experience.
4. Select **Yes** to continue and change the security type or select **No** to exit.

   If you choose to change the security type, configure the security in WindowMaker. For information on how to configure security refer to Configure application security in Operations Control connected experience

   Note: You must check that your scripting functions are compatible with AVEVA Operations Control connected experience. For more information refer to the Supported InTouch HMI functionalities section.

Start WindowViewer with incompatible security type

1. Launch WindowViewer.
2. You are prompted to authenticate with AVEVA Identity Manager using a web browser.
3. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.
4. After successful sign in and with valid entitlement, if the security type is not compatible you will be prompted to configure the security in WindowMaker. Select **OK** to exit the WindowViewer.



   Note: InTouch and **OS** security types are not supported in Operation Control connected experience.
5. Configure the security type in the WindowMaker. For information on how to configure security refer to Configure application security in Operations Control connected experience.
6. Then launch the WindowViewer.

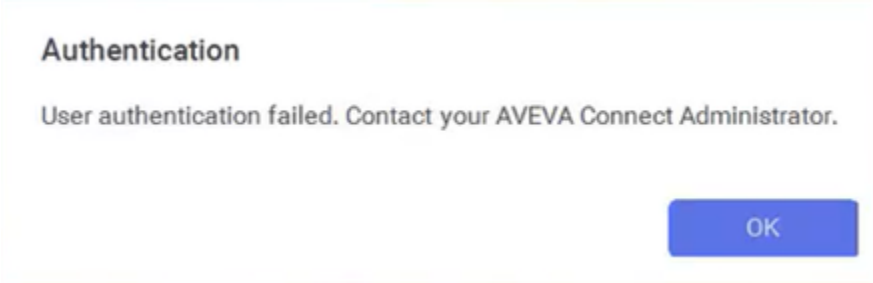
Start and run Application Manager

This section explains about the behavior of Application Manager in Operations Control connected experience.

Work with Application Manager in Operations Control connected experience

1. Launch Application Manager.

   If you are launching Application Manager for the first time on this node and no other Operations Control product was launched and authenticated on the same node before, then you are prompted to authenticate with AVEVA Identity Manager on a web browser or embedded browser (based on the choice that you have made while configuring **Authentication** tab in System Management Server of the Configurator). If you have already launched and authenticated Application Manager on this node or launched and authenticated any other Operations Control product on the same node before, then you will be authenticated using Single Sign-On. For more information refer to Authenticate using AVEVA Identity Manager section.
2. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.
   - When successfully authenticated:
     - Your subsequent logins will be authenticated with Single Sign-On.
     - The Application Manger ribbon shows the information of the logged-in user.
     - You will be able to publish tags, access AVEVA Industrial Graphics drive and Credential Manager, run View applications in Read-Only or Read/Write mode, and perform DBLoad.
   - If the authentication is not successful or authentication is canceled:

     - Application manager will stay on without any logged-in user.
     - You will have to authenticate with AVEVA Identity Manager when you attempt to launch WindowMaker or WindowViewer.
     - You will be able to run WindowViewer in Read-Only mode only. In Read-Only mode, you can only view the application and cannot make any changes. WindowMaker will exit.
   - If you close the embedded browser login page that was launched to authenticate with AVEVA Identity Manager, the authentication failure dialog will be displayed immediately.
   - If you do not provide credentials or close the web browser that was launched to authenticate with AVEVA Identity Manager, the authentication failure dialog will be displayed after 3 minutes.

Select and run an application from Application Manager

This section explains about the behavior of opening an application in Application Manager in Operations Control connected experience.

Open an application with Application Manager in Operations Control connected experience

1. Launch Application Manager. To launch the Application Manager, select **Start**, point to **Programs**, point to **AVEVA InTouch HMI**, and then select **InTouch HMI Application Manager**.

   If you are launching Application Manager for the first time on this node and no other Operations Control product was launched and authenticated on the same node before, then you are prompted to authenticate with AVEVA Identity Manager on a web browser or embedded browser (based on the choice that you have made while configuring **Authentication** tab in System Management Server of the Configurator). If you have already launched and authenticated Application Manager on this node or launched and authenticated any other Operations Control product on the same node before, then you will be authenticated using Single Sign-On. For more information refer to Authenticate using AVEVA Identity Manager section.

   The AVEVA Application Manager opens.
2. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password. Once you successfully authenticate or if the authentication is not successful, the Application Manager will launch.
3. Select the required application to run.

   If the authentication was not successful in the previous step, then you will be prompted to authenticate with AVEVA Identity Manager.

   After successful authentication, if the application has the security type that is not compatible with Operations Control connected experience, you will be prompted with a message that the security type is not compatible and if you want to change the security type or if you want to exit.
4. If you select **Yes**, the application will open in WindowMaker.
5. Configure the **Security type**.

For more information on security type, refer to Configure application security in Operations Control connected experience.

Start and run InTouch Access Anywhere

This section explains about the behavior of InTouch Access Anywhere in Operations Control connected experience.

Work with InTouch Access Anywhere in Operations Control connected experience

1. Launch InTouch Access Anywhere using your web browser, go to

   `http://ITAA_Server_Node_Name:8080/`

   or

   `http://ITAA_Server_IP_Address:8080/`
2. In the logon page, enter your user name, password, and select the InTouch application you want to view from the drop-down list of the **Application Name** field.
3. Select **Connect** to initiate an InTouch Access Anywhere browser session.

   If you are launching InTouch Access Anywhere for the first time on this node and no other Operations Control product was launched and authenticated on the same node before, then you are prompted to authenticate with AVEVA Identity Manager on a web browser or embedded browser (based on the choice that you have made while configuring **Authentication** tab in System Management Server of the

Configurator). It is recommended to use embedded browser option for security reasons. If you have already launched and authenticated InTouch Access Anywhere on this node or launched and authenticated any other Operations Control product on the same node before, then you will be authenticated using Single Sign-On. For more information refer to Authenticate using AVEVA Identity Manager section.

4. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password. Once you successfully authenticate, the WindowViewer will launch.
5. After authenticating successfully:
   - Your subsequent sign ins will be authenticated with Single Sign-On.
   - WindowViewer verifies that the Edge Entitlement access is available.
   - If the Entitlement is verified, WindowViewer logs the use of Edge in Audit Log page of CONNECT website. For more information refer to the CONNECT help.

Start a Network Application Development (NAD) application

Network Application Development (NAD) supports Operations Control connected experience. Note that for NAD to work in Operations Control connected experience, you must enable Operations Control connected experience on all nodes of your system.

Work with NAD application in Operations Control connected experience

1. Launch NAD client.

   If you are launching NAD for the first time on this node and no other Operations Control product was launched and authenticated on the same node before, then you are prompted to authenticate with AVEVA Identity Manager on a web browser or embedded browser (based on the choice that you have made while configuring **Authentication** tab in System Management Server of the Configurator). If you have already launched and authenticated NAD on this node or launched and authenticated any other Operations Control product on the same node before, then you will be authenticated using Single Sign-On. For more information refer to Authenticate using AVEVA Identity Manager section.
2. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.

   Once you signed in successfully and if you have Edge or Supervisory subscription, Single Sign-On (SSO) will be used to sign in and the application runs. You will not have to enter the credentials again on subsequent logins.

   Note:

   - If the sign is successful and do not have valid entitlement then contact your administrator for valid entitlement. If the sign in is not successful, you will be prompted an error message. Contact your administrator for valid credentials.

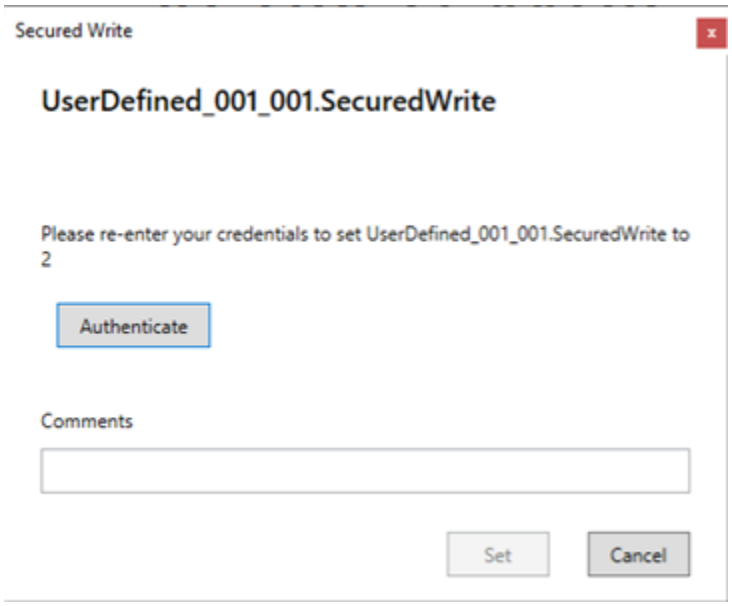
   - If a machine gets disconnected, you will not be able to authenticate again and therefore you will not be able to run applications.

Execute Secured and Verified Writes

Execute Secured Write

1. Change a value

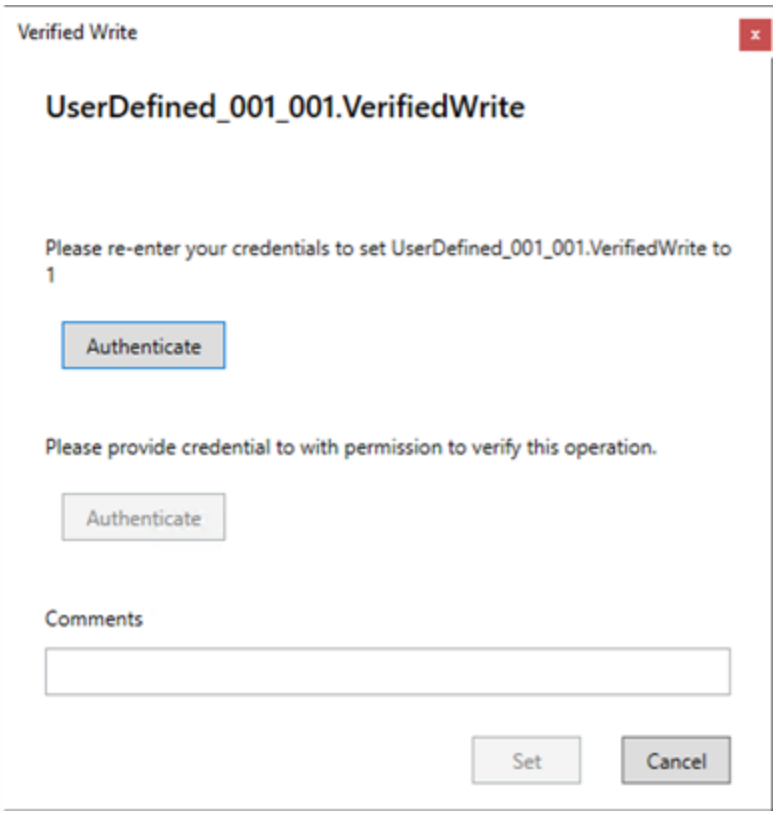
2. Secured Write dialog appears.
3. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.



Note: Single Sign-On is not supported for Secured Write operations. That is, the operator must explicitly enter their credentials for a Secured Write operation.

Execute Verified Write

1. Change a value
2. Verified write dialog appears.
3. Operator selects **Authenticate**, and enters the email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.
4. Verifier selects **Authenticate**, and enters the email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.



Note: Single Sign-On is not supported for verified write operations. That is, both the operator and verifier must explicitly enter their credentials for a Verified Write operation.

Start and run Web Client (on-premise)

When Operations Control connected experience is enabled in the Configurator, you will be prompted to authenticate with AVEVA Identity Manager, when the Web Client is launched for the first time. You have to use your AVEVA ID and password to sign in. In the subsequent launches of Web Client, you will be authenticated with Single Sign-On (SSO). If you have already signed in the AVEVA Identity Manager while launching the other AVEVA applications like Application Manager, and WindowMaker, and if you have valid entitlements and permissions, you will be signed in using SSO. Note that SSO between desktop applications and Web Client will work only when desktop applications use the system browser to authenticate. SSO is browser specific. That is, if you had entered the credentials in one browser for example Google Chrome, and if you open the Web Client in another browser for example Microsoft Edge, or if you open Web Client in incognito mode, then you will have to enter the credentials again.

Note: If you are upgrading from System Platform 2023 to System Platform 2023 R2, and if you have already configured AVEVA Identity Manager in System Platform 2023, then post-upgrade you must reconfigure user authentication by switching to windows authentication and then to user authentication to make Web Client work in Operations Control connected experience. That is, you have to first select **Windows Authentication** under **Industrial Graphics Server** > **Authentication Settings** and select **Configure**. Then, again select **User Authentication** under **Industrial Graphics Server** > **Authentication Settings** and select **Configure**.

Work with Web Client (on premise) in Operations Control connected experience

1. To enable the Web Client in Operations Control connected experience, in the left pane of the Configurator,

under **Industrial Graphics Server** > **Authentication Settings**, select **User Authentication** and then select **Configure**.

2. Launch the Web Client in a browser.

   If you are launching Web Client for the first time on this node and no other Operations Control product was launched and authenticated on the same node before, then you are prompted to authenticate with AVEVA Identity Manager on a web browser or embedded browser (based on the choice that you have made while configuring Authentication tab in System Management Server of the Configurator). If you have already launched and authenticated Web Client on this node or launched and authenticated any other Operations Control product on the same node before, then you will be authenticated using Single Sign-On. For more information refer to Authenticate using AVEVA Identity Manager section.

3. Enter your email address that is registered with a CONNECT account to sign in to AVEVA Identity Manager and the respective password.

   On successful authentication and valid entitlement, the Web Client runs. You will not have to enter the credentials on subsequent logins as it signs in using Single Sign-On (SSO).

   Note: If the credentials are not correct then you will be prompted with an error message. Contact your administrator for the credentials and enter the correct credentials again.

Scenarios for Invalid Entitlements

This section describes the different scenarios that could occur if the entitlement is not valid.

- If you do not have access to entitlement, WindowViewer will launch in demo mode.

  Note: Demo mode will be offered, if application has 64 tags or less configured. demo mode will not be offered if application has more than 64 tags configured.

- If current authentication is unsuccessful, but if you have successfully authenticated and logged in at least one time on that node previously, WindowViewer will launch in Read-Only mode. Read-Only mode will continue to run until that locally stored expiration date passes. When the expiration date has passed, application will run in demo mode.
- If you launch the WindowViewer and cancel authentication, and there is a login timeout, WindowViewer will start in demo mode or Read-Only mode.
- If you try to authenticate and only receive a valid AVEVA Identity Manager token, and is disconnected from CONNECT, then on-premise functionality will work and Cloud access will fail. You need to login to regain cloud access.
- If you attempt to authenticate, and do not receive a valid AVEVA Identity Manager token, and is disconnected from AVEVA Identity Manager, WindowViewer will start in demo mode and Read-Only mode.
- When WindowViewer is running successfully, a check is made for valid AVEVA Identity Manager and CONNECT refresh tokens.If only a valid AVEVA Identity Manager token is received, and AVEVA Identity Manager is disconnected from CONNECT, on-premise functionality will work, but cloud access will fail.
- If disconnected from AVEVA Identity Manager, WindowViewer will start in demo mode or Read-Only mode.

Unsupported Functionalities

The following features are not supported by Web Client:

- Windows authentication is not supported in Operations Control connected experience.

- Share functionality is not supported in Operations Control connected experience.
- The Web Client does not support redundant single sign on functionality in any licensing mode.

License and Entitlements for Web Client in Operations Control connected experience

If you switch from other licensing mode to Operations Control connected experience, you have to reconfigure the **Authentication** settings.

In Operations Control connected experience when **InTouchView Application** checkbox is enabled, to launch the Web Client you need to have entitlement to a Supervisory subscription in your account. Once you start the web server, if you modify the **InTouchView Application** checkbox then you have to restart the CONNECT Industrial Graphics Web Server.

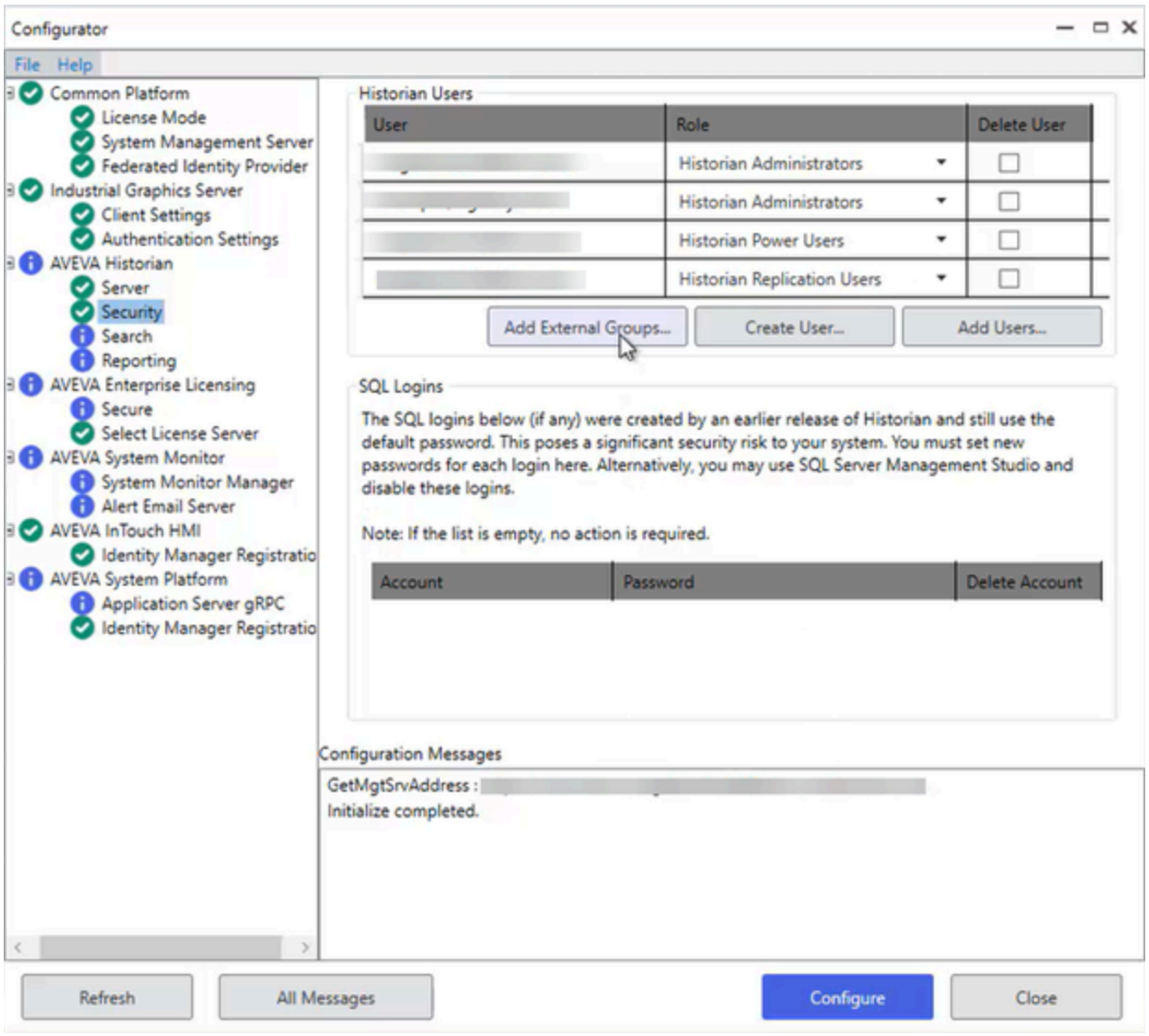
Start and run Trend Control

The Trend Control leverages Single Sign-On functionality, so you experience a seamless Single Sign-On from WindowViewer. This section explains about the behavior of Trend Control in Operations Control connected experience.

Work with Trend Control in Operations Control connected experience Configurator settings

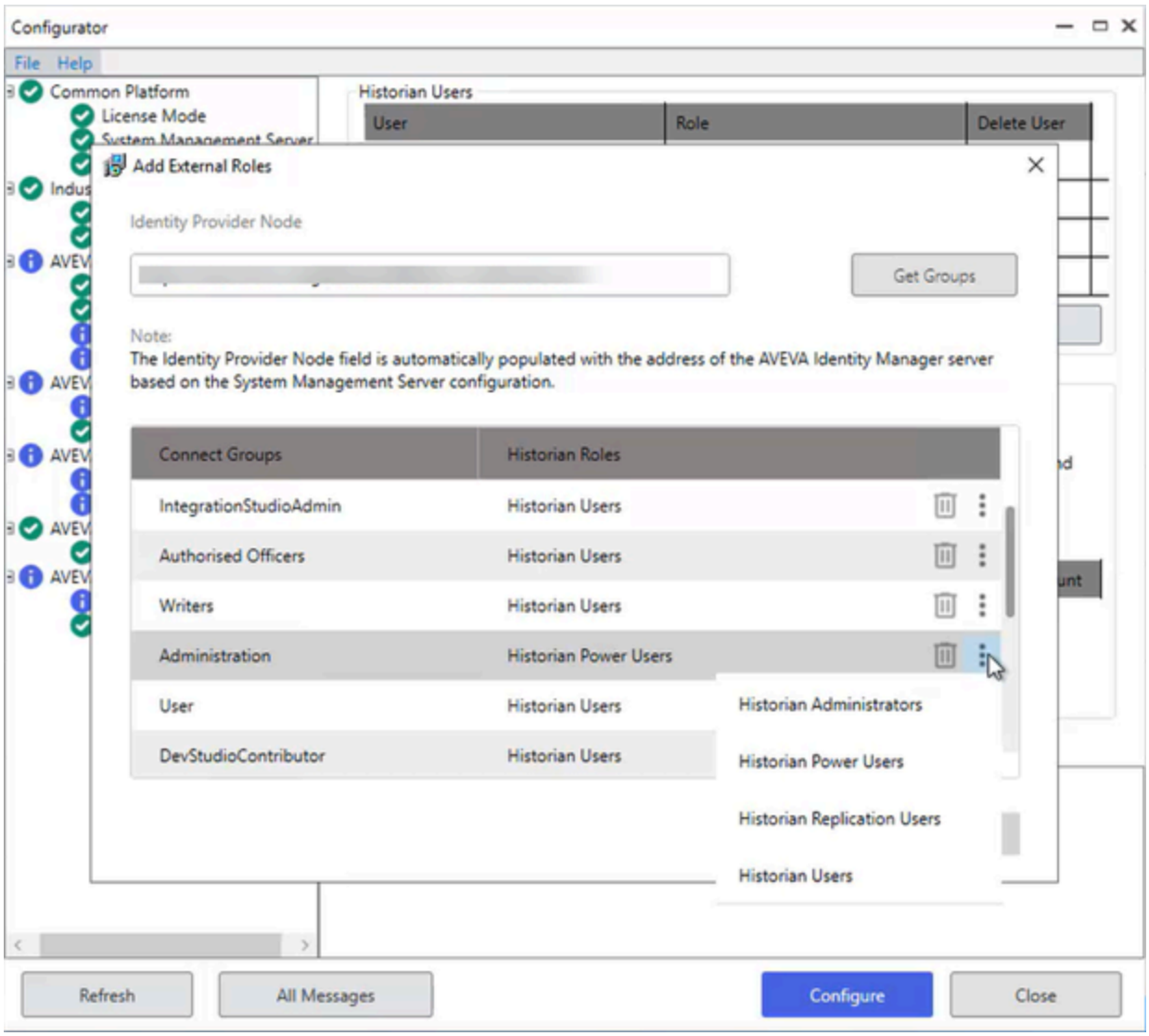
Apart from the other Configurator settings mentioned in the Configure Operations Control connected experience section, you have to configure the the following:

1. Configure AVEVA Historian Server. For more information, refer to the Configurator help.
2. In the Configurator, go to **AVEVA Historian** > **Security**.
3. Select **Add External Groups**.



4. In the **Add External Roles** window, groups that are in CONNECT are listed. Under **Connect Groups** column, select the required CONNECT group that the Trend Control user will be part of, and then select the vertical ellipsis (three dots) and select either **Historian Administrators** or **Historian Power Users**.

Note: For the Trend Control to work in Operations Control connected experience, the user of Trend Control must be part of CONNECT group and that group must have **Historian Roles** as **Historian Administrators** or **Historian Power Users**.



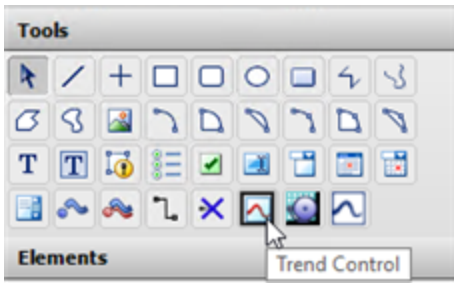
5. Select **Save** and then select **Configure**.

Configure Historical Logging

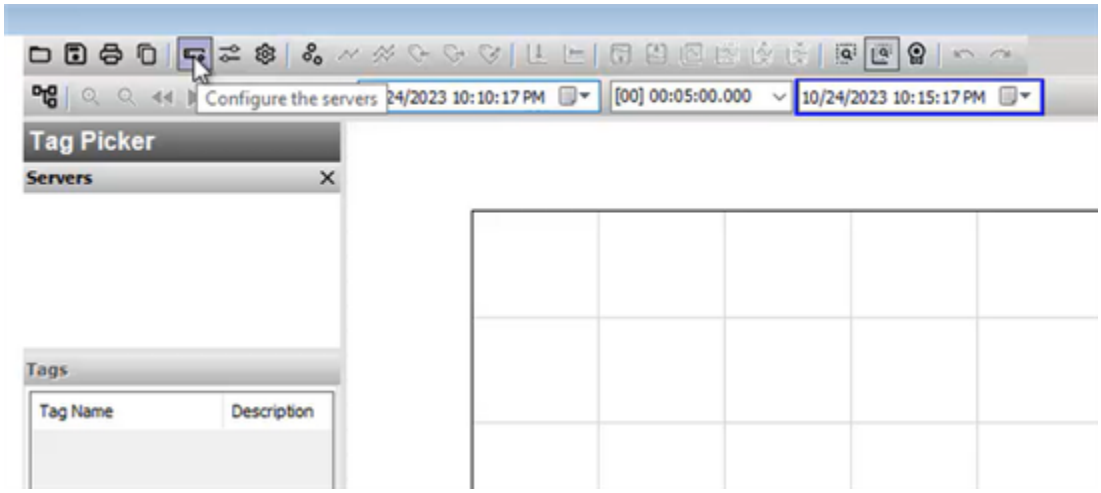
Once you configure the Configurator, you must configure the **Historical logging** in WindowMaker. For more information refer to the Configure general logging properties storage to Historian section of the AVEVA™ InTouch HMI Application Development Guide (ITBuild.pdf).

Configure the login as Single Sign-On in Server List Configuration:

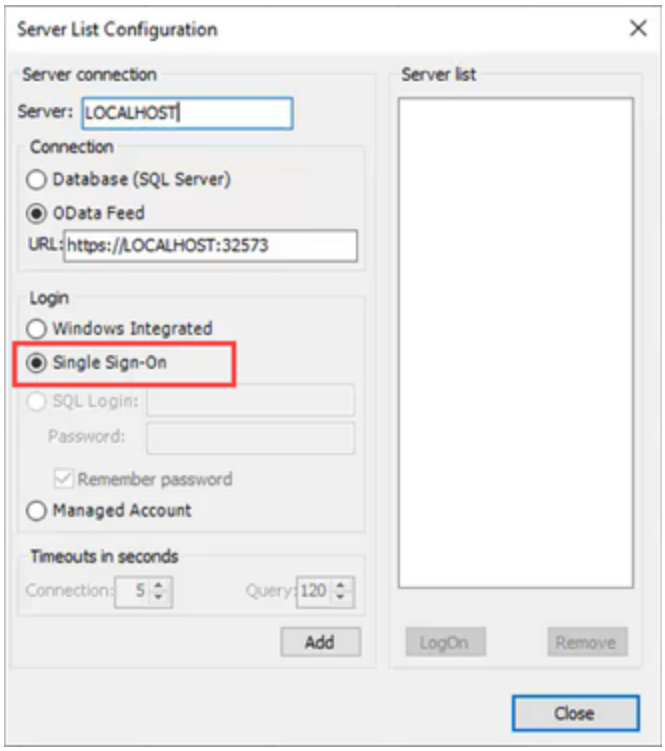
1. In WindowMaker, under **Industrial Graphics** pane, right-click and select **New Graphic** to create a graphic.
2. Right-click the newly created graphic and select **Open or** double-click the graphic to open it in Industrial Graphic Editor.
3. From the **Tools** pane, select **Trend Control** and add it to the canvas. Save and close the graphic.



4. In the WindowMaker, add the newly created graphic to the canvas and Save.
5. Select **Runtime** on the top-right corner of the WindowMaker to open the window in WindowViewer.
6. Select **Configure the servers**.



7. In the **Server List Configuration** dialog box, if you enter the **Server** field, then the **URL** field under **OData Feed**, is updated accordingly.
8. Under **Login** section, select **Single Sign-On** and then select **Add**.



The server gets added to the **Server list**.

9. Select **LogOn**.

Sign out from AVEVA applications

You can sign out from InTouch, ViewApp, or other Operations Control connected experience application by selecting the **Sign Out** option under the **Profile** icon. When you sign out of an application:

- Sign out is closing the session from CONNECT.
- When you sign out of a desktop application, you are only signed out of that specific desktop application. Other desktop applications will continue to run with their current signed in user.
  - If you launch the desktop application a subsequent time, you will be prompted to authenticate again.
- When you sign out of a desktop application using the embedded browser, web applications will continue to run.
- Closing an application without signing out, does not do anything with the session. Only that instance of the application closes. The session is still on the CONNECT server. When you open that application again, your session is available and you will be authenticated with Single Sign-On.
  - If the session has expired, then when you launch the application, you will be required to authenticate again.
- When you sign out of WindowMaker, the session is closed and WindowMaker exits.
- When you sign out of WindowViewer, the WindowViewer runs in Read-Only mode.

Supported InTouch HMI functionalities

Supported client controls

The following InTouch HMI client controls are supported in Operations Control connected experience using AVEVA Identity Manager authentication.

- **Alarm Client Control:** The Alarm Client Control leverages SSO functionality so that you can experience a seamless Single Sign-On from the view application for History Blocks. The authenticated user with entitlements can view Historian alarms in the WindowViewer.

  Note: For Alarm Client Control to work in Operations Control connected experience, you must configure External Roles from CONNECT.
- **Trend Control:** The Trend Control leverages SSO functionality so you experience a seamless Single Sign-On from the view application.
- **Trend Pen:** The Trend Pen leverages SSO functionality so that you can experience a seamless Single Sign-On from the view application. The authenticated user with entitlements can view Historical Trend in the WindowViewer.

Supported system tags

The following InTouch HMI system tags are supported in Operations Control connected experience using AVEVA ID.

- **$AccessLevel:** The $AccessLevel system tag has been updated to be used with the AddPermission() method. If a runtime user is a member of multiple groups from CONNECT, the access level will be determined by the group with the highest access level.
- **$AccessTokenChanged:** The $AccessTokenChanged System Discrete tag is a Read-Only tag.
- **$Operator:** The $Operator system tag will display the signed in user from CONNECT.
- **$OperatorName:** The $OperatorName system tag will display the signed in user from CONNECT.

Scripting methods

InTouch HMI also supports scripting methods in Operations Control connected experience. The following scripting methods were either added or modified to support Operations Control connected experience.

- GetAccessToken()
- GetAccessSecureToken()
- AddPermission()

  The AddPermission() method accepts only two parameters in Operations Control connected experience:
  - AVEVA Connect Group
  - Access Level

    Script function for AddPermission() in Operations Control connected experience:

    ```
    DiscreteTag=AddPermission("", "AVEVA Connect group", AccessLevel);
    ```

If a runtime user is a member of multiple groups from CONNECT, the access level will be determined by the group with the highest access level.

- GetTokenConnectionstatus()
- PostLogonDialog() function
- SignedWrite()
- IsAssignedRole()
- Logoff()

Note: The ChangePassword() script function is not supported in Operations Control connected experience and will be disabled.

CONNECT Industrial Graphics drive

Operations Control connected experience provides Single Sign-On functionality for CONNECT Industrial Graphics drive in WindowMaker. If you have authenticated in CONNECT under the connected experience, you can view your Industrial Graphics drive when launching WindowMaker. This allows you to start using the CONNECT Industrial Graphics drive in WindowMaker without having to authenticate again. Therefore, in Operations Control connected experience, you will not see the **AVEVA Connect** tab under **File** menu of the WindowMaker.

- In Operations Control connected experience when you launch WindowMaker, if you authenticate successfully with CONNECT using your credentials or through Single Sign-On and if you have valid subscription, the WindowMaker will open.
  - You can view your CONNECT Industrial Graphics drives that are available in the Graphic Toolbox. You can also navigate to other tenants in your account using the drop-down menu.
  - You can interact with the Industrial Graphics cloud drive based on the permissions you have on the CONNECT Industrial Graphics drive.

    Note: You must have the Content Contributor role to access CONNECT Industrial Graphics drive. This is required for both Operations Control connected experience and non-connected experience use.

Secured write/verified write

- WindowViewer supports Secured Write and Verified Write with authentication through AVEVA ID.
- You have to authenticate with CONNECT to perform secured write or verified write action.
- SSO is not supported for secured write or verified write operations.

Network Application Development (NAD)

NAD is supported for Operations Control connected experience and authentication with AVEVA Identity Manager.

InTouch Access Anywhere

InTouch Access Anywhere is supported for Operations Control connected experience and authentication with AVEVA Identity Manager.

Web Client

- On launching the Web Client URL, you will be prompted to authenticate with CONNECT.
- After logging in successfully, if you have valid entitlements and permissions, the Web Client runs.

Unsupported InTouch HMI functionalities

The following InTouch products and features will not be supported in Operations Control connected experience at the time of this release:

- Insight Publisher
- Widgets

The following InTouch products and features are not supported in Operations Control connected experience:

- Alarm DB Logger
- Alarm DB Purge Archive
- Alarm DB Restore
- Alarm Hot Backup Manager
- Smart Card
- Application Manager Credential Manager

Use InTouch in non-Operation Control mode

This section explains functions those are not controlled by AVEVA Operations Control connected experience authentication and entitlements.

License in InTouch HMI

InTouch HMI uses the AVEVA Enterprise License Server to make licenses available to InTouch. The AVEVA Enterprise License Manager manages one or more License Servers.

To make licenses available to InTouch HMI, complete the following steps:

1. Import the entitlement XML file received upon purchase of the license.
2. Using the **License Manager** interface, select the licenses on the entitlement that you want to activate on the License Server.
3. Once the licenses are activated, they becomes available to WindowMaker or WindowViewer upon start up.

The activated licenses appear in the License Manager under the License Grid.

InTouch releases and returns the consumed license to the License Server when:

- The machine running InTouch is shutdown or
- The InTouch application is shutdown

Note: In the event of InTouch HMI shutting down abnormally, licenses will not be returned. InTouch HMI must be restarted and manually shut down to release licenses.

The License Manager and License Server are installed with InTouch HMI. InTouch HMI will point to your local License Server by default. You can change this configuration in the post-install Configurator. Refer to the *AVEVA Enterprise Licensing Guide* for the detailed procedure.

InTouch licenses are based on varying numbers of tags you can use while running an application. You need to understand how tags are counted with the InTouch license scheme. You can use a set of functions to calculate the anticipated number of remote reference tags in your applications.

Important: Licensing numbers are subject to change at any time.

Understand license tag counts

While an InTouch application is running, tag handles are stored in a memory database. Each tag must be assigned a handle. Tag handles are initialized and used by WindowViewer, but never saved permanently to disk after the application stops.

InTouch HMI now supports unlimited tag count. This means that the run-time database can theoretically store unlimited tag count license, which includes both local tags and tags that reference a remote tag source. However, we have tested up to 300000 tag handles (300K) only.

An InTouch application can never have more simultaneously active tags than the available memory handles in this run-time database. Your InTouch license determines how many local and remote tags can be assigned handles within the run-time database.

Also, the maximum number of active tags in an application is restricted by the functional limits of the run-time database. The actual maximum tag count is less than the theoretical maximum tag handles. A set of constants is subtracted from the maximum potential tag count.

- The invalid tag handle bit is reserved to indicate if an invalid handle value occurs within the WindowViewer run-time database.
- InTouch version 10 includes 34 system tags, which cannot be replaced by user-defined tags. If you migrate a version 7.11 or earlier application to the current version of InTouch, the system tag count is 37.
- At configuration time, 4096 database handles are reserved to store placeholder tags. When you import windows, scripts, or symbols during configuration time, placeholder tags are assigned to this memory segment. During run time, all 4096 placeholder handles are available to be assigned to remote reference tags.

InTouch license options are based upon the maximum number of local and remote reference tags that can be used in an application.

If an InTouch tag license is for less than 60K tags, then a sticky tag licensing scheme is enforced. A sticky tag is a remote tag reference that is bound at run-time when WindowViewer receives a data change notification for the remote reference. WindowViewer updates remote tag references during run-time up to the maximum limit of the InTouch license. WindowViewer does not update any additional remote tag references beyond the license limit. WindowViewer does not decrement the remote reference tag count when a window is closed. Each remote reference tag count sticks while the application is running.

A single message appears when you exceed the maximum remote reference tag count of your InTouch license. After the license maximum is reached, the values associated with invalid remote reference tags are never updated in the application. You must stop and restart the application before you can open other windows that

include one or more remote tag references that are not already associated with those counted against the license limit.

A license that permits 60K or more tags means that sticky tags are not enforced and there is no enforced limit to the number of tags or the combinations of local and remote tags that can be used within an InTouch application.

For example, using a 60K licence, the implementation limits for local and remote reference tags are:

- Total possible local tags

  61404 = 65535 – (4096 + 37 + 1)
- Total possible remote reference tags

  Maximum = 65535 – (37 + 1 + Number of Local Tags)

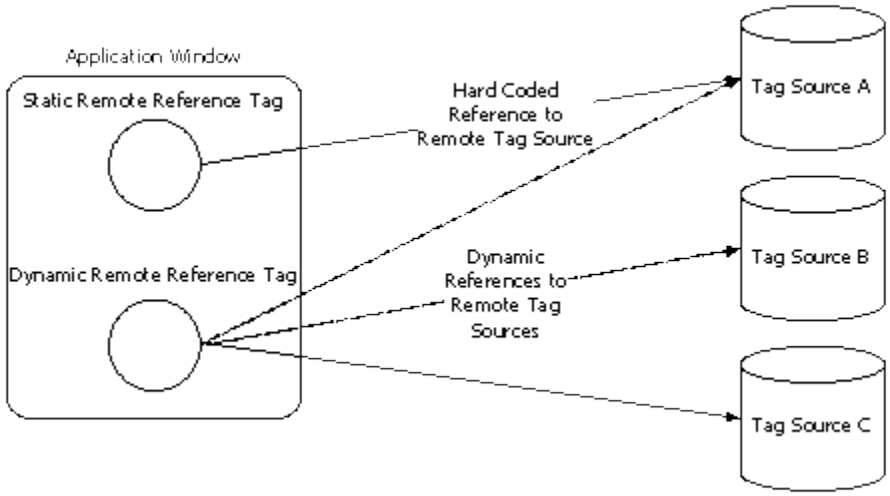
So, running an application with a 60K license, you are effectively trading a potential remote reference for every user-defined tag in the local tag database. You can never have less than 4096 possible remote reference tags available in a run-time configuration. For a license that permits 100000 tags, the Maximum = 100000 – (37 + 1 + Number of Local Tags)

Understand InTouch remote reference limits

There are two types of InTouch remote reference tags. A *static* remote reference is hard coded to a fixed remote address when you define the tag from the Tagname Dictionary. A static remote reference is assigned a tag handle in the tag database when the application starts running. A static remote reference tag count sticks while the application is running.

A *dynamic* remote reference resolves the target address while the application is running. If the dynamic remote reference tag is assigned a database handle, the target address can be changed during run time by using the .Reference dotfield or IOSetRemoteReference() function within a script.

The following figure shows an example of an InTouch application running under a 300K license without sticky remote reference tags. The count for the static remote reference tag sticks while the application is running. But, the count for the dynamic remote reference tag is only for the active tag source. The previous connections to remote tag sources do not stick and are not counted in the remote reference or total tag count.



The InTouch 60K license does not use sticky tag counts that impose limits on the number of dynamic remote tag references. This allows an application to dynamically access more than 60K tags during the period the application is running. The tag use count for dynamic remote references fluctuates up and down as windows with remote

references are opened and closed. But, the application can never have more simultaneously active tags than the implementation limit of the run-time tag database.

Dynamic reference tag counts only fluctuate up and down when WindowViewer uses disk storage to save the contents of a running application. If WindowViewer is configured to cache InTouch windows or Industrial graphics, the remote reference tag counts may not decrease unless a window is removed from the cache.

When a window is not visible does not mean the remote tag references are not still bound to their sources. However, if window caching is completely disabled, and no high priority windows have been specified, then WindowViewer operates much like the legacy "Always load from disk" scenario. In this case, all windows are removed from memory when they are closed and the dynamic remote tag references are reclaimed in a 60K license environment.

A remote reference from an I/O tag is not included in the sticky remote reference count of the InTouch license. An I/O tag's remote reference can change an unlimited number of times without counting against the sticky remote reference limit.

When fail-over to the secondary tag source occurs, the application can access the same items from the secondary source without increasing the licensed tag count. After failover, accessing new items from the secondary tag source increases the tag count. These items are accessible after fail-back to the primary tag source.

After the tag count reaches the licensed maximum, no further items can be activated regardless of whether they are accessed from the primary or secondary tag source.

Licenses available for InTouch HMI

InTouch HMI provides different types of licenses to manage various scenarios. Licenses are determined based on various parameters such as:

1. **Console Type**: specifies the console type; Remote Desktop Services/Terminal Services or non-RDS nodes. RDS/TSE is a console running on a machine that is configured with terminal server, while Non-RDS is a console running on a machine that is not configured with terminal server. For more information, see Deploying and Working with Terminal Services and Remote Desktop Services.
2. **Access Type**: specifies the access type the node is configured as - Read-Only or ReadWrite. For more information see, Configure user access to applications running in remote sessions.
3. **Data Source**: specifies the data source the application will use - Galaxy or InTouch Tag Server. For more information, see InTouchView applications.

InTouch HMI Unlimited RDS License

The InTouch HMI Unlimited RDS License is consumed by WindowViewer for unlimited client sessions, only in an RDS enabled machine. When WindowViewer acquires the license, the application is licensed. The Read/Write access depends on the Remote Access configuration. The same unlimited license will provide both Read-Only and ReadWrite access. If the license is not acquired, then the licensing will depend on the existing RDS handling and Remote Access configuration.

InTouch licensing in RDS and non-RDS environments

If an InTouch application is running on a server node enabled with Remote Desktop Services (RDS), the console

will behave the same as an RDS client session. Each session will consume a license. Each session will also consume a separate InTouch development license.

In this case, the InTouch application's ReadWrite capability is defined by its remote access configuration and confirmed by the consumed license. For example, if an application with ReadOnly remote access configuration is launched in WindowViewer in an RDS client session, it will look for a ReadOnly InTouch license. If an RDS ReadOnly license is not available in the License Server, startup license validation will fail.

On a node without RDS enabled, you can also login with a RDS client session that is allowed by the operating system. If an InTouch application is running in this non-RDS environment, the client session will behave the same as a console. In this case, the application's remote access configuration does not determine the ReadWrite access. ReadWrite access is determined only by the license in non-RDS environments.

In non-RDS environment, when the tag count is less than or equal to 64:

- The application will always run in Read/Write mode.
- With ReadOnly InTouch license, after migration, the InTouch application will not consume the license and will always run in Read/Write mode.

  Note: You cannot run an application in ReadOnly mode when the tag count of an application is less than or equal to 64.

About InTouchView application licensing

An InTouchView application shows visual interfaces designed specifically for use in an Application Server environment. See InTouchView applications for details on this application type.

The type of license an InTouchView application consumes depends on whether it is running in an RDS environment.

If an InTouchView application is running in a RDS client session, it will look for a ReadOnly or ReadWrite client connection license, depending on the remote access type configuration of the application.

Only one connection license will be consumed per RDS session.

If InTouchVew application is running in a non-RDS client session and if tag count is less than or equal to 64, then the application will be always in Read/Write mode.

Note: An InTouchView application consumes the same license as the Graphic Run Time Module's ViewApp application, if configured with a Galaxy data source.

InTouchView application licensing

You can configure an InTouch HMI application as an InTouchView Application, where it can be used as a client to an InTouch Tag Server or Galaxy.

If you configure the InTouchView application to connect to data from an InTouch Tag Server then the licenses available are:

	InTouch HMI Client ReadWrite License	InTouch HMI Client Read-Only License	InTouch HMI Unlimited Client License*
Remote Access	ReadWrite	Read-Only	ReadWrite &

			Read-Only
RDS/TSE	Yes	Yes	Yes
Non-RDS	Yes	Yes**	No
Supports MarkAppReadOnlyNonRDS	Yes	Yes	N/A

* This license serves unlimited number of RDS clients.

** If the tag count is grater than 64.

If you configure the InTouchView application to connect to data from a Galaxy then the licenses available are:

	Supervisory Client ReadWrite License	Supervisory Client Read-Only License	Supervisory Client Server License
Remote Access	ReadWrite	Read-Only	ReadWrite & Read-Only
RDS/TSE	Yes	Yes	Yes
Non-RDS	Yes	Yes*	No
Shared with OMI	Yes	Yes	Yes
Supports MarkAppReadOnlyNonRDS	Yes	Yes	N/A

* If the tag count is grater than 64.

Best practices for administering InTouch licenses on the server

There are several best practices to follow when administering InTouch licenses that will ensure InTouch license consumption is deterministic. Deterministic license consumption allows you to consume appropriate licenses on demand for a particular system. This type of license consumption will make it easier to administer InTouch licenses using the server-based AVEVA Enterprise Licensing system.

The two best practices for deterministic license consumption are license reservations and floating licenses. Refer to the sections below for details.

Reserve licenses

You can reserve licenses to specific devices in the License Manager. Reserving a license to a particular device ensures that the license cannot be acquired by another InTouch application and interrupt or prevent your application from running.

User-based License Reservation

In the AVEVA Enterprise License Manager license reservation page, it is possible to mark a license to be reserved to a specific user. While the reservation page allows this particular configuration, it's important to know that neither InTouch OMI nor InTouch HMI ViewApps support user-based license reservations. The end-result will be the inability for the software to acquire the license reserved. Therefore, only use device-based reservations for Supervisory Client licenses.

Device-based License Reservation

When reserving a Supervisory Client license for a specific device, the Device Name needs to be the name of the computer running the InTouch HMI/OMI ViewApp. In the case where the ViewApp is running inside of an RDS or Terminal Server, the Device Name needs to follow this naming pattern:

<RDSHostName>-<RDPClientName>-<index>

where RDSHostName is the name of the RDS or Terminal Server, and RDPClientName is the name of the PC running the RDP client software, and "index" is 1, unless there will be multiple RDP sessions from a single client machine, in which case the index should be incremented (starting at 1) for each reservation for that specific RDP client, up to the total number of RDP sessions from that specific RDP client.

Example 1: A computer with a hostname of "ControlRoomA" runs InTouch OMI

Device Name: "ControlRoomA"

Example 2: A computer with a hostname of "ControlRoomB" running a single Remote Desktop Client (RDP), connecting to the Remote Desktop Server (aka: Terminal Server) with a hostname of "PrimaryRDS"

Device Name: "PrimaryRDS-ControlRoomB-1"

Example 3: Two computers with hostnames "SupervisorPC1" and "LineMgrA", respectively, each running a single Remote Desktop Client (RDP) connecting to the Remote Desktop Server (aka: Terminal Server) with a hostname of "PrimaryRDS"

Device Names:

License Reservation 1: "PrimaryRDS-SupervisorPC1-1"

License Reservation 2: "PrimaryRDS-LineMgrA-1"

Example 4: A computer with a hostname of "ExecutiveDesktop" running four (4) Remote Desktop Clients (RDPs), connecting to the Remote Desktop Server (aka: Terminal Server) with a hostname of "PrimaryRDS"

Device Names:

License Reservation 1: "PrimaryRDS-ExecutiveDesktop-1"

License Reservation 2: "PrimaryRDS-ExecutiveDesktop-2"

License Reservation 3: "PrimaryRDS-ExecutiveDesktop-3"

License Reservation 4: "PrimaryRDS-ExecutiveDesktop-4"

For RDS load balancing support, all RDS licenses can be activated on a single License Server that multiple RDS client sessions can point to. The licenses on the server must be of the same capability so that the licenses can be shared amongst each RDS client session. Licenses are considered to be of the same capability if their internal parameters have the same value. No reservations are needed in this scenario. If different license types for different RDS client sessions are required, then a License Server must be installed on each RDS server.

Refer to the *AVEVA Enterprise Licensing Guide* for detailed license reservation procedures.

About floating licenses

Floating licenses are not reserved to any machine. It is recommended to have floating licenses of the same

product name and capabilities on a single License Server. For example, you could have a License Server with several activated InTouch 2017 Runtime 60K tags licenses with the same capabilities. Similarly, you could have a License Server with several activated InTouch 2023 Runtime Unlimited tags licenses with the same capabilities.This is a recommended practice to ensure deterministic license consumption.

However, it is not recommended to have licenses of the same product name but different capabilities activated on the same License Server. For example, you could have a mix of InTouch 2023 Runtime Unlimited tags activated licenses with InTouch 2017 Runtime 500 tags activated licenses on the same license server. In this scenario, there is no way to ensure which instance of WindowViewer will consume the license with the higher tag count.

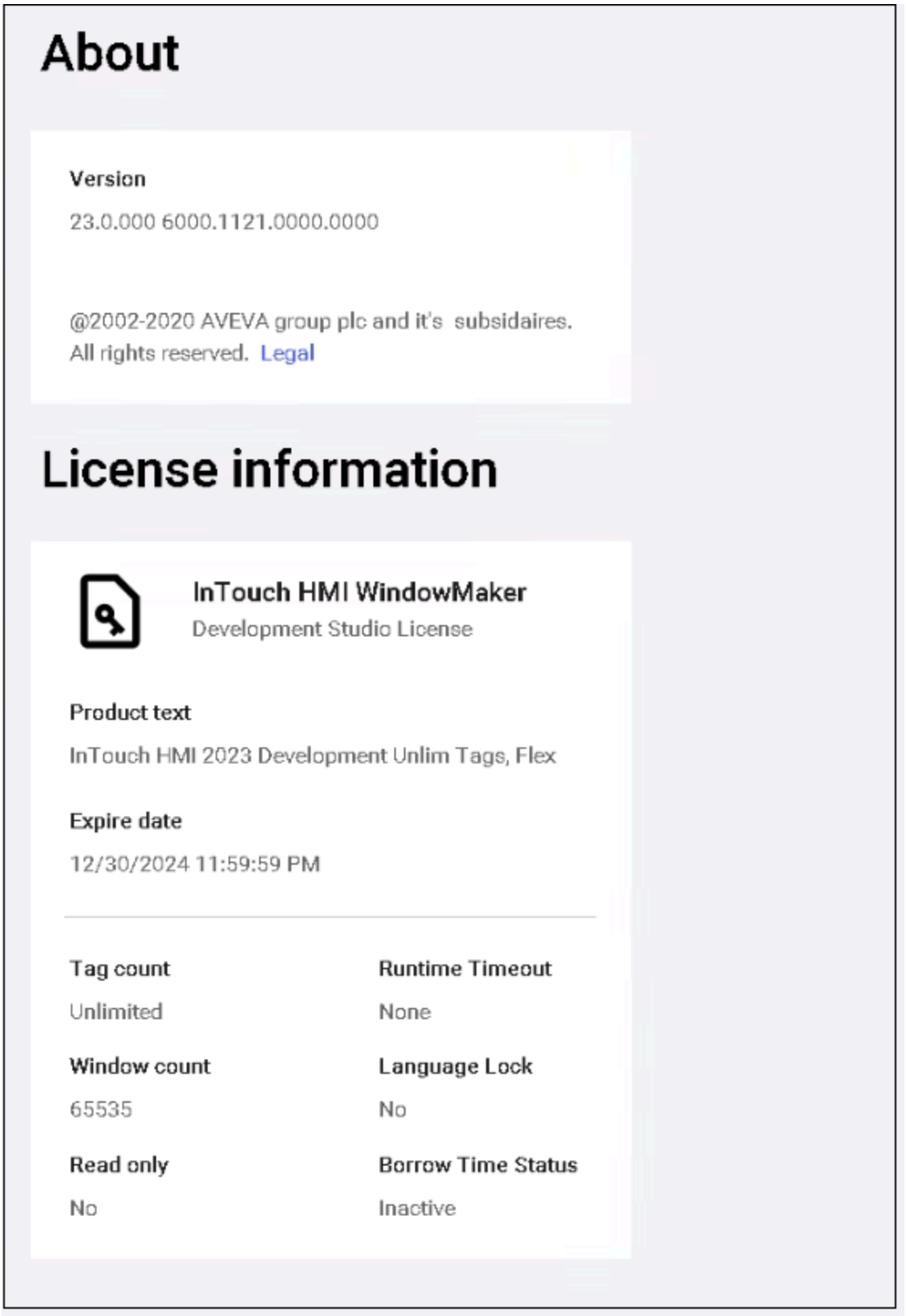
View license information

You can view the specific information for the current license consumed by WindowMaker or WindowViewer.

View WindowMaker License Information

1. Open WindowMaker.
2. On the **File** menu, at the bottom-left corner of the screen, select **License**.

   The **About** screen appears. The **About** screen displays the WindowMaker **Version** and **License information**.

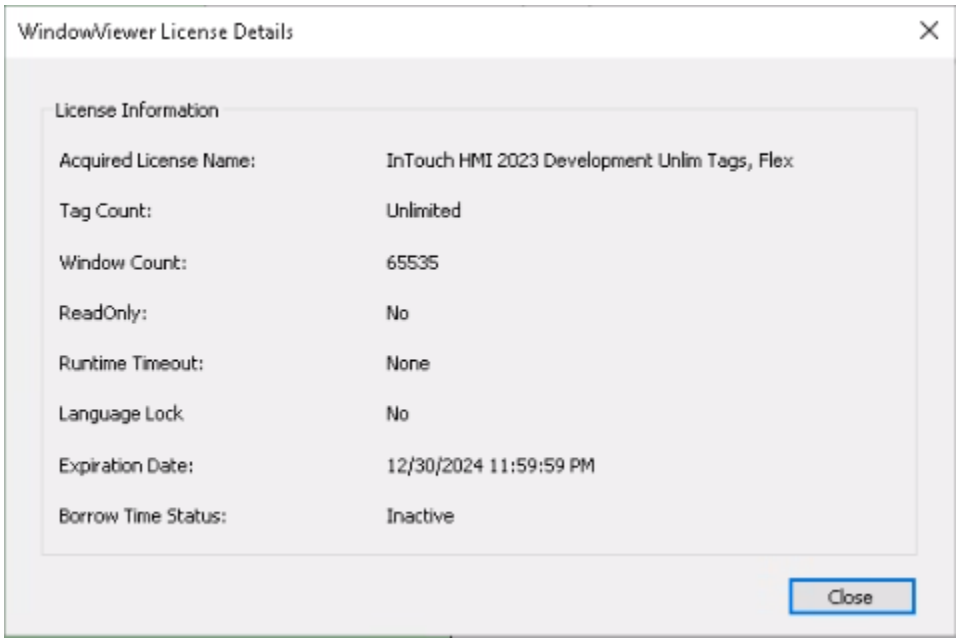


The company name and license serial number do not appear in this dialog. This information appears in the AVEVA Enterprise License Manager interface.

View WindowViewer License Information

1. Open WindowViewer.
2. Select **File** and then **About WindowViewer**.

   The **About WindowViewer** dialog box appears.
3. Select **View License Agreement** to view the End User License Agreement.
4. Select **View License** to view the details for the license. The License Information dialog box displays.



The parameters displayed in the license information are as follows:

- **Acquired License Name:** the full name of the license consumed by the product from the License Server.
- **Tag Count:** the number of tags allowed by the consumed license.
- **Window Count:** the number of windows allowed by the consumed license.
- **ReadOnly:** displays the I/O Read/Write permissions allowed by the license. No indicates that the application can write to I/O tags.
- **Runtime Timeout:** the application runtime allotted by the license. The InTouch session will end when the timeout period elapses.
- **Language Lock:** applies to licenses for InTouch on Chinese operating systems only. A Chinese license or English license must be consumed for InTouch to run on a Chinese operating system.

  The Language Lock restriction does not apply for a connection license.
- **Expiration Date:** the date the consumed license expires.
- **Borrow Time Status:** intended to notify the user when the license is 50% past whatever the allotted borrow time is. The status will change to **Active** when the 50% has been reached. If the license is not renewed when the borrow time has fully elapsed, InTouch will become unlicensed.

Note: The **View License** option is disabled for InTouch Application Manager. Because Application Manager does not consume a license, you can only view the EULA. The **About Application Manager** dialog box does not display any license-related information.

Manage consumption of a different license after startup

As part of standard upgrade and maintenance activities, different licenses can be deactivated or activated on the License Manager. If WindowMaker or WindowViewer consumes a valid license after startup time and cannot renew the license, it can consume a different license, from the one initially assigned or consumed. If this occurs, InTouch must then validate if the capabilities of the newer license is appropriate. The comparison in license capabilities between the last good license and the currently consumed licenses will determine if the WindowMaker or WindowViewer can continue running without entering the Grace Period. See Work with the Grace Period for details on how to exit the Grace Period.

The two possible scenarios for a different license being consumed after startup are described below.

Scenario 1: A less capable license is consumed after startup

In this scenario, the license consumed after startup time is of lesser capabilities than the last good license. This is considered a license downgrade, and will result in WindowMaker or WindowViewer entering the Grace Period.

A license downgrade is considered when the new license's parameters fall into the below check:

The parameter changes that will trigger the Grace Period are described below:

- Tag Count: if the tag count is reduced, the Grace Period is triggered.
- Window Count: if the window count is reduced, the Grace Period is triggered.
- Runtime Timeout: if the Timeout value changes from **None** to any other value, Grace Period is triggered.
- Language Lock: if the Language Lock value changes from **No** to **Yes**, or vise versa, Grace Period is triggered.
- ReadOnly: if the ReadOnly parameter is changed from **No** to **Yes** or vise versa, Grace Period is triggered.

If a license downgrade occurs, the downgraded, or "violated" parameters display in the **View License** dialog box. For example, if WindowMaker originally consumed a sixty-thousand tag count license and downgraded to a three thousand tag count license, the tag count parameter would display as follows:

Important: The application will continue to run with the capabilities of the last good license. In this example, the original sixty-thousand tag count will remain when entering the Grace Period.

Scenario 2: A more capable license is consumed after startup

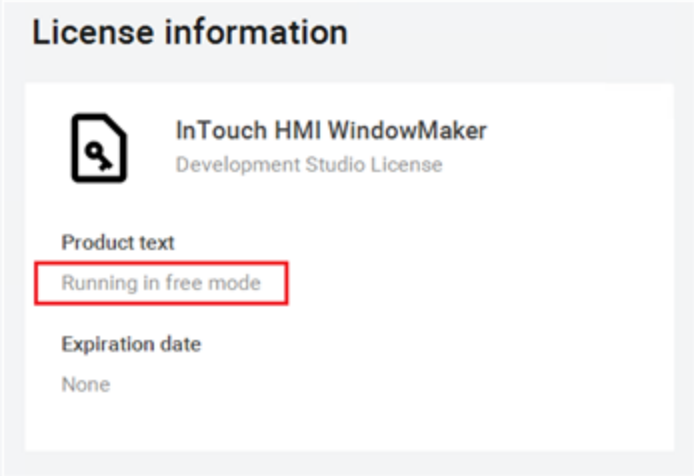
In this scenario, the license consumed after startup time is of greater or equal capability than the original. This is a license upgrade, and will not cause WindowMaker or WindowViewer to enter the Grace Period.

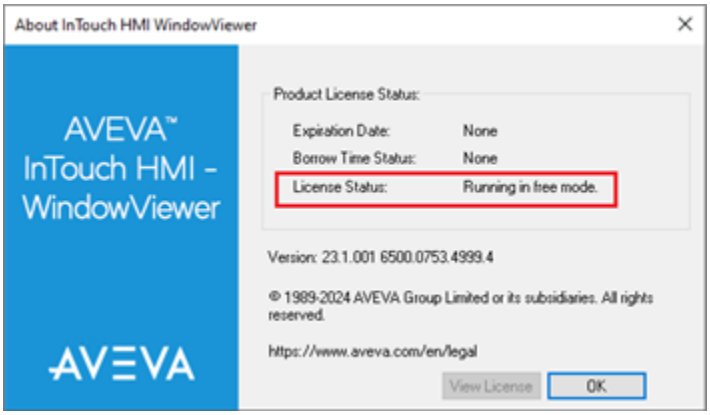
If a license upgrade occurs, all parameters in the **View License** dialog box are updated to display the higher capability values.

Work in free mode and demo mode

If an application has less than or equal to 64 tags, then WindowMaker and WindowViewer are allowed to run with full capability without any license check. This is called free mode. If an application has more than 64 tags, WindowMaker will still run in free mode.

Note: Free mode is not supported in Operations Control connected experience.





WindowViewer will run in demo mode if an application has more than 64 tags and when it cannot consume a valid license at startup time. When a valid license is not present while launching WindowViewer, a pop-up message stating that license could not be acquired will be displayed. You can choose to do one of the following:

- Update a valid license and select **Retry**.
- Select **Demo Mode** to continue with demo mode for next 120 minutes.
- Select **Exit** to exit WindowViewer.

While running in demo mode, there is no tag count or window count limit. However, WindowViewer will only run in demo mode for two hours before timing out. When demo mode timeout is reached, it will prompt to exit.

Note: While in demo mode, even if you activate a valid license, you need to exit WindowViewer and restart WindowViewer to consume the valid license.

Configuring the ViewLicenseRetryCount key in the InTouch.ini file, will instruct WindowViewer to continue to attempt acquiring a license in the background for the number of times specified in the parameter. If a license is acquired, the dialog box will close and WindowViewer will be launched.

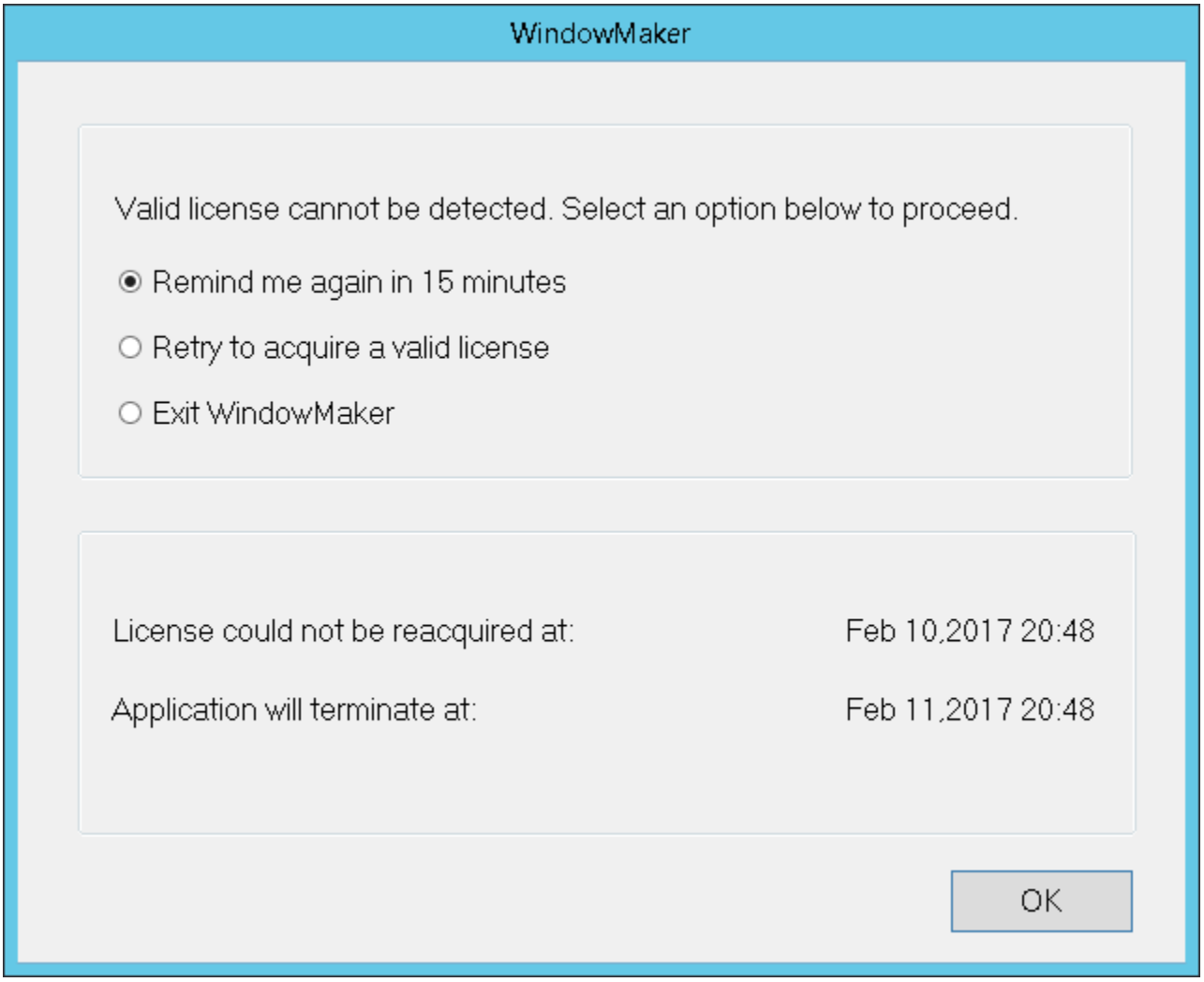
Work with the Grace Period

The Grace Period is a twenty-four hour period in which InTouch can continue to run with the last good license's capabilities after certain conditions have occurred. Both WindowMaker and WindowViewer can enter the Grace Period. At the end of the Grace Period InTouch will terminate if a valid license is not reacquired within the

allotted time.

The Grace Period will be triggered by the following scenarios. If valid license is not reacquired within twenty-four hours, WindowMaker or WindowViewer will terminate.

When WindowMaker or WindowViewer enters the Grace Period, you will be prompted with the following dialog:



You have the options to be reminded again, to retry to acquire the license, or to exit the application.

Scenario 1: Consumed License is Lost

WindowMaker or WindowViewer will go in to Grace Period if it consumes a valid license from the License Server and the license is deactivated while the product is still running.

To exit the Grace Period and resume normal operation, activate a valid license on the License Server. When the license is consumed, WindowMaker or WindowViewer will exit the Grace Period and normal operation will resume.

Scenario 2: License Expired

WindowMaker or WindowViewer will go in to Grace Period if it consumes a valid license from the License Server and the license expires. To exit the Grace Period and resume normal operation, activate a valid license on the License Server.

If WindowMaker or WindowViewer fails to acquire the license, the first dialog will appear again.

Scenario 3: License is Downgraded

AVEVA Enterprise License Manager is a server-based licensing system, which means that licenses need to be renewed periodically. If a WindowMaker or WindowViewer license is downgraded during this renewal to one of lesser capability, it will enter the Grace Period. See Manage consumption of a different license after startup for a detailed description of downgrade scenarios.

To exit the Grace Period and resume normal operation, retry to acquire the last good license or a better license.

Important: The functionality enabled by your last good license will persist in Grace Period mode.

In all of the above scenarios, if you select the option to retry to acquire the license and an appropriate license is successfully acquired, you will see the message that the license has been acquired successfully.

If InTouch fails to acquire the license, the first dialog will appear again.

Note: Configuring the ViewLicenseRetryCount key in the InTouch.ini file, will instruct WindowViewer to continue to attempt acquiring a license in the background for the number of times specified in the parameter. If a license is acquired, the dialog box will close and WindowViewer will be launched.

Remote tag count functions

The InTouch HMI includes a set of functions to verify if your applications conform to the remote tag requirements of your license. You can write temporary scripts that include these functions to test and identify any potential licensing issues with your application before deploying it into production. After you verify your application does not have any licensing issues, remove the scripts.

IORRGetSystemInfo() function

The IORRGetSystemInfo() function returns a tag count for a running InTouch application. Based upon an argument value, the IORRGetSystemInfo() function returns a numerical value, which can be:

- Maximum number of remote tag addresses specified by the InTouch license.
- Number of remote tag addresses counted against the license over the period that an InTouch application has been running.
- Number of remote tags currently activated in an InTouch application.
- Number of available remote tags in a running InTouch application.
- Number of remote reference tags, which are currently disabled.
- Number of local tags in a running InTouch application.
- -1 if an error occurs during the function call or the *Option* argument is assigned an invalid value.

Category

Miscellaneous

Syntax

```
IORRGetSystemInfo(Option);
```

Argument

Option

An integer tag or integer constant that specifies the type of remote reference tag count to return. Possible values are:

1	Returns the maximum number of permitted remote tag addresses based upon the InTouch license. Local I/O tags are not counted in the remote tag count. This number is constant while the InTouch application is running.
2	Returns the number of unique remote tag addresses counted against the licensed limits that are activated while an InTouch application is running. Local I/O tags are not counted in the remote tag count. If the license permits more than 60K remote reference tags, this number may be 0, regardless of how many remote tag addresses are activated. While running under an unlimited license, WindowViewer does not count the activated remote tag addresses. While running an application under a license that has a remote tag limit, this count increments up to the limit of the remote tag license count. After the remote tag limit is reached, no further remote tag addresses can be activated. Only addresses currently activated can be reactivated. Use IORRWriteState with the *Option* argument set to 3 to obtain a list of remote reference tags that count against the license limit.
3	Returns the number of remote reference tags currently activated within an InTouch application.
4	Returns the number of remote reference tags that can be activated in an InTouch application without running out of remote tag handles. This count typically changes while an application is running. Stopping and starting scripts and opening and closing windows containing remote reference tags affect the remote reference tag count. This number can be less than the number still remaining on the license, especially if the license is unlimited. This occurs because there is an internal limit to how many remote reference tags can be active simultaneously.
5	Returns the number of remote tags currently in the disabled state within an InTouch application.

6	Returns the number of local tags currently in the InTouch application.

Example

The following example returns the number of remote reference tags counted against the license limit while an InTouch application is running. The returned remote reference tag count is assigned as the value of the RRTagCount integer tag.

```
RRTagCount = IORRGetSystemInfo(2);
```

IORRWriteState() function

The IORRWriteState() function saves information about the current state of an application's remote tags to a text file. The function creates the file if it does not exist. Each time the script containing the function runs, new information is concatenated to the file.

You can specify what remote tag information is saved to the file. Also, the function's return value indicates whether information is added successfully to the file.

Category

Miscellaneous

Syntax

```
IORRWriteState(FilePath, Option, " ");
```

Arguments

FilePath

Full folder path to the text file containing information about an application's remote tags. The *FilePath* argument can be a string constant or a message tag.

Option

An integer tag or integer constant that specifies the remote tag count information written to the file. Possible values are:

1	List of current remote tag addresses. The information also includes each remote tag's state, activation time, and deactivation time.
2	List of all currently active addresses, including activation time.

3	List of all remote reference tag addresses that have been activated and counted against the license limit. New items are added to the list as new remote tag addresses are activated. When the license limit is reached, no further items are added to the list. However, no addresses are added to this list if the remote tag license limit is unlimited.
4	List of current addresses not activated because the remote reference tag count exceeds the license limit or because the internal tag handle limit was reached. Does not return addresses related to licensing if the remote tag license limit is unlimited. If the licensing is unlimited, the list contains any items that are not currently active because of implementation limitations. If any item in this list becomes active, it is removed from the list. When an item is deactivated because of licensing limitations, it is removed from the list. This list updates while the InTouch application is running.

Empty String " "

This argument is reserved for future use, but must be included with the IORRWriteState() function in a script.

Results

These are some examples to interpret information saved to the file by the IORRWriteState() function call.

Current Addresses Listing

The following line from the output file shows an example of a fully activated remote reference tag, which can be updated. This line shows that address 65535 is assigned to the TestProt:di000 remote reference tag:

```
65535 <TestProt:di000> (RAA) {C:5/23/2007 9:58:35 AM} {A:5/23/2007 9:58:35 AM}
```

Following the remote reference tag name are three flags enclosed within parentheses:

- The first flag, R, indicates the tag's remote reference has been successfully resolved. The first flag is assigned a value of X if the tag's remote reference is still pending.
- The second flag indicates if the remote tag is currently active, A, or inactive, D.
- The third flag indicates if the address is allowed, A, or disallowed, D, because of licensing restrictions.

The date following C indicates when the remote tag was created. The date following A shows when the tag was most recently activated. The line includes a trailing deactivation time if the remote reference tag has been deactivated while the InTouch application is running.

The following line from the output file shows an example of an active remote reference tag that exceeds the tag count limit of the InTouch license. The tag's remote reference is successfully resolved and the tag is currently active:

```
65414 <TestProt:di121> (RAD) {C:5/23/2007 9:58:35 AM} {A:5/23/2007 9:58:35 AM}
```

But, no tag value updates occur within the InTouch application because the address assigned to the remote reference tag exceeds the tag count limit of the InTouch license.

Active Address Listing

The following line from the output file shows an example of a fully activated remote tag whose assigned values update the InTouch application:

```
65429 <TestProt:di106> (A) {C:5/23/2007 9:58:35 AM} {A:5/23/2007 9:58:35 AM}
```

The first number is the remote tag handle, followed by the address, then the flags, A, for allowed or D for disallowed. The creation, most recently activated, and most recently deactivated times follow the flags in the output line. The deactivation time does not appear if the remote tag has never been deactivated while the application is running.

The following line from the output file shows an example of an active remote tag that exceeds the license limits:

```
65342 <TestProt:di193> (D) {C:5/23/2007 9:58:35 AM} {A:5/23/2007 9:58:35 AM}
```

Licensed Addresses

The following line from the list shows the address assigned to the remote reference tag and when it was added to the list.

```
<testprot:di000> {C:5/23/2007 9:58:36 AM}
```

Denied Addresses

Denied addresses appear in the list because of implementation limitations, or because the tag count exceeds license maximum.

This example shows a remote tag address, which exceeds the license limit:

```
testprot:di125 [1] (L) {F:5/23/2007 9:58:39 AM} {R:5/23/2007 9:58:39 AM}
```

The address is listed along with the count to indicate how many times an attempt was made to reference the item. The flag indicates if the address is in the list because of an exceeded license, L, or because of an internal implementation limit, I. The two times represent the first time it was added to the list and its most recent access time.

Example

This example writes the current activated remote tag addresses to a file located in the c:\intouch\data folder. The ReturnValue tag is assigned an integer, which indicates whether the function call successfully wrote remote tag information to the file.

```
ReturnValue = IORRWriteState("c:\intouch\data", 2, "");
```

IORRGetItemActiveState() function

The IORRGetItemActiveState() function returns the status of a specified remote tag address.

Category

Miscellaneous

Syntax

```
IORRGetItemActiveState(ItemPath, Option);
```

Arguments

ItemPath

ItemPath is a string that represents the address of interest. *ItemPath* can be a string constant or a message tag.

Option

An integer tag or integer constant that specifies the type of remote reference tag count to return. Possible values are:

1	Determine if a current remote tag address is currently active. The return value is 1 if the address is current and active. The return value is -1 if the address is not current. The return value is 0 if the address is current but inactive.
2	Determine if a current remote tag address has ever been activated while an application is running. The return value is 1 if the address is current and has been activated at least once. The return value is -1 if the address is not current. The return value is 0 if the address is current and never been activated.
3	Determine if a current remote tag address has ever been deactivated. The return value is -1 if the address is not current. The return value is 0 if the address is current and never been deactivated.
4	Determine if a current remote tag address is disabled. The return value is 1 if the address is current and has been deactivated at least once. The return value is -1 if the address is not current. The return value is 0 if the address is not disabled. The return value is 1 if the address is current and is disabled.
5	Determine if the address is in the allowed list. The return value is 0 if the address is not in the list. The return value is 1 if the address is in the list.
6	Determine if the address is in the disallowed list. The return value is 0 if the address is not in the list. The return value is 1 if the address is in the list.

Examples

This example determines if the TestProt:di000 remote tag address is currently active:

```
ReturnValue = IORRGetItemActiveState("TestProt:di000", 1);
```

This example determines if the TestProt:di121 remote tag address is currently disabled:

```
ReturnValue = IORRGetItemActiveState("TestProt:di121", 4);
```

This example determines if the TestProt:di001 remote tag address is currently counted against the license limit.

```
ReturnValue = IORRGetItemActiveState("TestProt:di001", 5);
```

License the InTouch Web Client

You need a valid Web Server license to login and view application graphics from a web browser. Licensing also extends to hosting application graphics on external websites. For detailed information on licensing, see the *AVEVA Enterprise Licensing Help*. The Web Server provides different types of licenses which allow you to connect to unlimited sessions for viewing and interacting with application graphics in a web browser, including:

- InTouch Web Server *Flex license*
- InTouch Web Server *Connection ReadWrite license*
- InTouch Web Server *Unlimited ReadWrite license*
- InTouch Web Server *Unlimited Read-Only license*

License Status Change

If the Web Server has acquired a valid license but then failed to renew the license, the Web Client will be in Grace Period mode. A notification message will be displayed in the notification page. The message will be logged every time Web Client attempts and fails to renew the license.

For an InTouch HMI application, WindowViewer must run with Read/Write license in order for the Web Client to connect to WindowViewer and receive InTouch tag data.

Acquire a license for InTouch application graphics

Acquiring the license is a two-stage process. After the Web Server starts up, it will first Authenticate the user and then determine the user’s Authorization. If anonymous access has been enabled then the authentication step is bypassed and InTouch Web Client is launched in a read-only mode with the 'Guest' user.

Authentication

1. InTouch Web Client supports Windows authentication. The web server will validate if the user is part of either the "aaInTouchUsers" or the "aaInTouchRWUsers" to be authenticated to use the Web Client. Both the user groups will be created during installation. For a remote authentication server, the domain user groups must be created on the server.
2. The login user at the time of Web Client installation will be automatically added to both user groups. Additional users can be added later. After adding the new user to the group, the new user must log off and then re-login for the change to take effective.

Acquisition

Upon start-up and after the user is authenticated, the Web Server follows the workflow below to acquire licenses.

1. The Web Server will check if the Flex option is enabled in the Configurator.

   If the Flex option is not enabled, the Web Server will attempt to acquire the Unlimited ReadWrite license. (Go to Step 4)
2. If the Flex and Enterprise options in Configurator are enabled, the Web Server will attempt to acquire the Enterprise license.

   If the Flex option is enabled and the Enterprise option is not enabled in the Configurator, the Web Server will attempt to acquire the Flex license. (Go to Step 3)
3. The Unlicensed Web Server will then attempt to acquire the Flex license.

   If the Flex license is available, the Web Server will acquire the Flex license

   If Flex license is not available, then the Web Server is said to be Unlicensed state. If the Web Client attempts to connect when the Web Server is Unlicensed, a License Error Page will be displayed.
4. The Unlicensed Web Server will attempt to acquire the Unlimited ReadWrite license.

   If Unlimited ReadWrite license is available, the Web Server will acquire the Unlimited ReadWrite license.

   If Unlimited ReadWrite license is not available, the Web Server will attempt to acquire the ReadWrite Connection license.
5. If ReadWrite Connection license is available, the Web Server will acquire the ReadWrite Connection license..

   If ReadWrite Connection license is not available, the Web Server will attempt to acquire the Read-Only Unlimited license
6. If the Read-Only Unlimited license is available, the Web Server will acquire the read-only Unlimited license.

   If the Read-Only Unlimited license is not available, the Web Server will enter the Read-Only Single Session mode.

License features

The InTouch Web Client supports the following license features:

- Flex license
- ReadWrite license
- Read-Only Unlimited license

  The features of each of these modes are described below:

License acquired	Supported Features	Conditions
Flex license	• ReadWrite license allows unlimited connections • Personal Workspace with support for both InTouch and AppServer namespaces	See Note below.
ReadWrite license	• Write to external references such as AppServer attributes or InTouch tags • Ack alarm and its user credential will be recorded as the operator who acked the alarm. • Connection ReadWrite license allows a number of InTouch Web Client connections (for example, set of 5, 10, 15, etc.) to the Web Server.	You can access the Write and Alarm Ack capability if the following conditions are met: • Web Server has acquired the Flex, or Brand Neutral R/W License, or Unlimited ReadWrite license or the Connection ReadWrite license • InTouch Web Client session is within the allowed Read/ Write connection • User logged in to InTouch Web Client session belongs to Web Client ReadWrite group.

Read-Only license	• The client session cannot write to external references or acknowledge alarms.	The user can access the read-only mode only if: • Web Server has acquired the Flex, or Brand Neutral R/W or Unlimited ReadWrite license or Connection ReadWrite, but the user logged in does not belong to the InTouch Web Client aaInTouchRWUsers user group, or • The Web Server has acquired the Unlimited ReadOnly license, or • The Web Server does not acquire any license and this web client session is the first client session.

Note: Fallback is not allowed for Flex license. The Web Server will be unlicensed if it is unable to acquire Flex license. In the unlicensed state of the Web Server, if the InTouch Web Client attempts to connect, a License Error Page will be displayed.

Single session mode

If the Web Server does not acquire any license at startup, it will operate in the Single Web Session mode, which is assigned on a first come first serve basis.

- If a valid license is unavailable and the first session license has been acquired, the InTouch Web Client will notify the user that no licenses are available.
- The Web Server allows sessions a grace period if the Web Server has acquired an unlimited license, but subsequently loses the license.

Subscription to Application Server data will require a supervisory license. If you do not have a supervisory license, the subscription to application server data will cease after two hours.

Grace period

The Web Server will enter the grace period under the below conditions:

- If a valid license is acquired and later expires.
- If a valid license is acquired, but cannot be acquired later.

During the grace period, all InTouch Web Client features will be available for connected and new sessions. The Web Client page will display a visual notification that the Web Server is in grace period mode. The grace period is 14 days. After the grace period has ended, the Web Server will continue to function and immediately switch to a

single license for one session, on a first come first serve basis. If the same license is acquired, then the Web Server will exit the Grace Period mode.

Periodic renewal

After a valid license is acquired, a periodic license renewal will occur. The periodic license renewal operation will only check and ensure if the same license can be renewed. If the same license cannot be renewed, then Web Server will enter into a Grace Period mode.

The renewal frequency depends on the license type as listed below:

License Type	Renewal Frequency
Flex license Unlimited ReadWrite license Unlimited Read-Only license	Every 7 minutes
Connection ReadWrite license	Every 24 hours

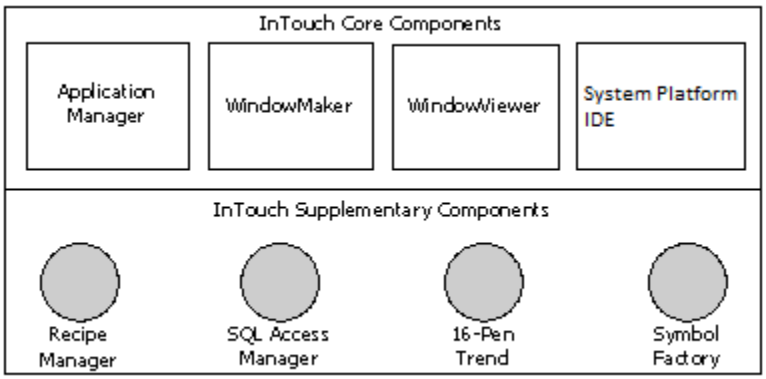
License status will not be upgraded or downgraded during renewal. To acquire a different license, restart the Web Server and initiate the startup sequence of license acquisition.

License release

The Web Server will release all the licenses it has acquired, if the Web Server is shutdown, regardless of the number of client sessions and the current state of the Web Server license. The Web Server will also terminate all Web Client sessions immediately.

About supplementary components

You can optionally install a set of four supplementary components along with InTouch HMI core components. These supplementary components provide additional functions for your InTouch HMI application.



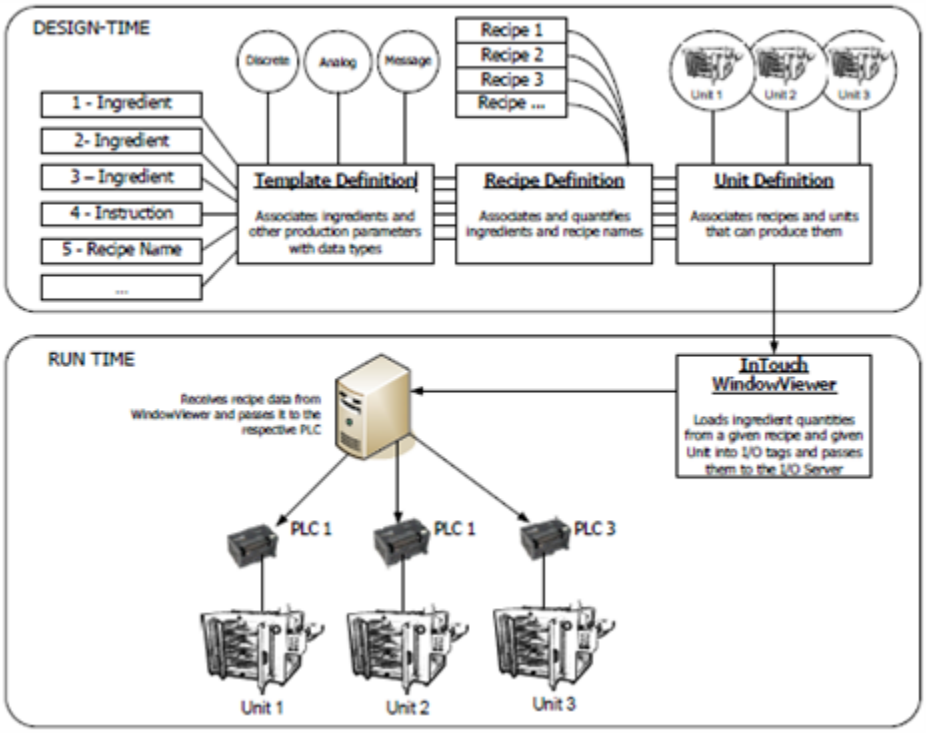
- Recipe Manager includes a set of spreadsheets and script functions to create manufacturing recipes.
- SQL Access Manager consists of a program and a set of SQL functions to store InTouch data to a database.

- 16-Pen Trend includes a trend wizard and script functions to create real-time and historical trends.
- Symbol Factory provides a set of industrial symbols that can placed in InTouch applications to represent process components.

Use Recipe Manager

Manufacturing industries build products according to repeatable procedures that use standardized quantities of raw materials. In essence, products are manufactured according to recipes. A recipe describes the raw materials, their quantities, and how they are combined to produce a finished product. In the most intuitive case, a bakery may follow a basic recipe that lists all ingredients and procedural steps to make cookies.

Recipe Manager is a supplementary component for the InTouch HMI that you can use to simplify the process of creating manufacturing recipes. The following figure summarizes how Recipe Manager obtains information from recipe templates to manage a process that creates a product.



Overview of Recipe Manager

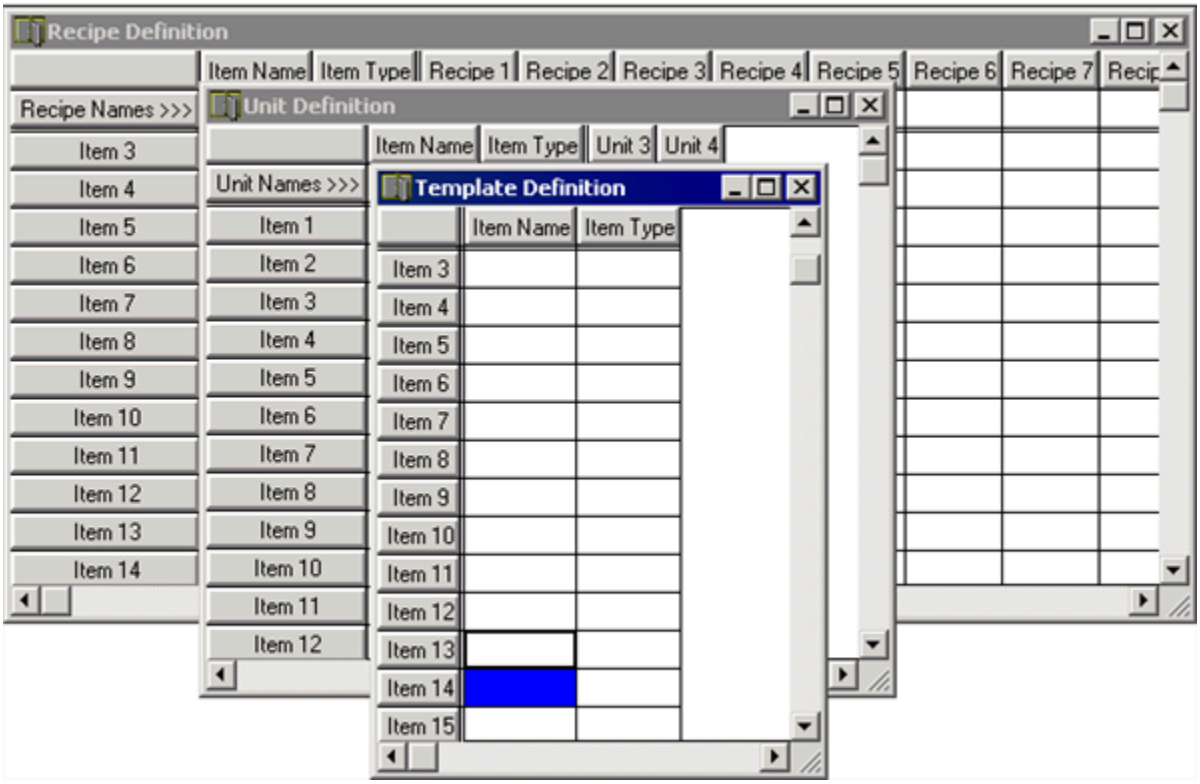
Recipe Manager can be installed with InTouch as an optional component. Recipe Manager consists of the Recipe Manager utility and a set of InTouch recipe script functions.

You can access the Recipe Manager utility from WindowMaker or by launching the Recipe.exe file from C:\Program Files (x86)\Wonderware\InTouch\. The Recipe Manager utility includes an interface for you to create and edit recipe templates. Recipe Manager saves your templates in a recipe file.

Typically, tags associated with a manufacturing process use QuickScripts to access data within recipe template files. Recipe Manager includes a set of QuickScript functions to select, load, modify, create, and delete the manufacturing recipes contained in template files.

Recipe Manager utility

The Recipe Manager utility provides a spreadsheet-like user interface to create and maintain a recipe template file. A file consists of three templates. You create and edit these templates by adding or modifying data within the cells of each template's spreadsheet.



You save these templates to a Comma Separated Value (CSV) file. You can create and edit your recipe template definitions with any program that supports the .csv file format like Notepad or Excel. But, Recipe Manager provides preformatted spreadsheets and a set of editing tools to create and maintain templates reliably and easily.

Recipe template files

A Recipe Manager template file contains the following information:

- Names of ingredients and their data types used in a recipe.
- Unit names that associate InTouch tags with recipe ingredient values.
- Recipe names containing the quantities or values for each ingredient used in a recipe instance.

Template definition

The Template Definition template defines all recipe ingredients. A data type is associated with each recipe ingredient. An ingredient data type can be analog, discrete, or message. Ingredient names are not required to be InTouch tags.

Unit Definition

The Unit Definition template associates InTouch tags with recipe ingredients. Many different loading definitions can be created. These definitions are called units. You can use the RecipeLoad() function to load specific instances of a recipe to associated InTouch tags. A Unit Definition can consist of all ingredients defined in the file or just a subset of these ingredients.

Note: Unit tags can be memory types that can be viewed and edited in an InTouch window or I/O tags that can be loaded directly to PLCs.

Recipe Definition

The Recipe Definition template specifies the name of each recipe and ingredient quantities used by the recipe. Recipe instances can be modified, created, or deleted in run time through the recipe functions.

Edit recipe data in Recipe Manager

You create manufacturing recipes by completing a set of sequential tasks. The following list shows the Recipe Manager tasks to create recipes and the order in which they should be completed:

- Configuring the Recipe Manager editing grid.
- Editing data within a template.
- Assigning ingredient names and unit types to the Template Definition template.
- Mapping InTouch tags to ingredients in the Unit Definition template.
- Assigning values to recipe ingredients in the Recipe Definition template.

Configure the Recipe Manager editing grid

Before you create manufacturing recipes, you should configure Recipe Manager. There are two tasks to configure Recipe Manager editing functions:

- Set the maximum limit for template items.
- Set the ENTER key scroll function.

Before you create recipes, you need to configure the maximum number of items that can be entered in your recipe templates. You must assign a set of maximum limits for items, units, and recipe names.

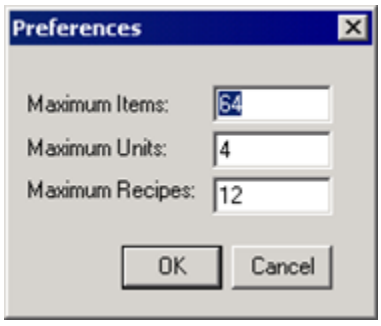
Templates can contain up to a maximum of 9999 items, units, and recipe names. However, large maximum limits can potentially affect system performance. Also, you may see an error message if the maximum limits you set will require more memory than the computer's available memory.

Configure recipe template maximum limits

1. Start Recipe Manager by one of the following methods:
   - Start WindowMaker. On the **Tools** view, expand **Applications**, and then select **Recipe Manager**.

     The Recipe Manager dialog box appears.

2. On the **Options** menu, select **Preferences**. The Preferences dialog box appears.



3. In the **Maximum Items** box, enter the maximum number of item names allowed in your Template Definition template.
4. In the **Maximum Units** box, enter the maximum number of units allowed in your Unit Definition template.
5. In the **Maximum Recipes** box, enter the maximum number recipe names allowed in your Recipe Definition template.

   Caution: The values you set in the **Preferences** dialog box are applied to all recipe template files that you create. When you modify these values, all existing recipe template files are also modified.
6. Select **OK**.

Recipe Manager includes an option that simplifies entering data in recipe templates. When you select the **Auto Down on [Enter]** option, you can press enter to move the cursor down to the next cell in the column.

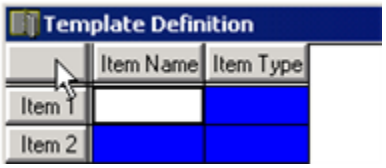
Set ENTER key template scrolling

1. Open Recipe Manager.

   By default, Recipe Manager does not scroll automatically to the next cell in the template spreadsheet.
2. On the **Options** menu, select **Auto Down on [Enter]** to set cell scrolling.
3. Select **Auto Down on [Enter]** again if you want to turn off cell scrolling.

Work with the editing grid

Recipe Manager includes a set of editing commands to add, modify, or delete data from the templates. Generally, you select the data that you want to edit in the template, and then take an editing action. The following table describes common features of recipe templates to enter and select template data.

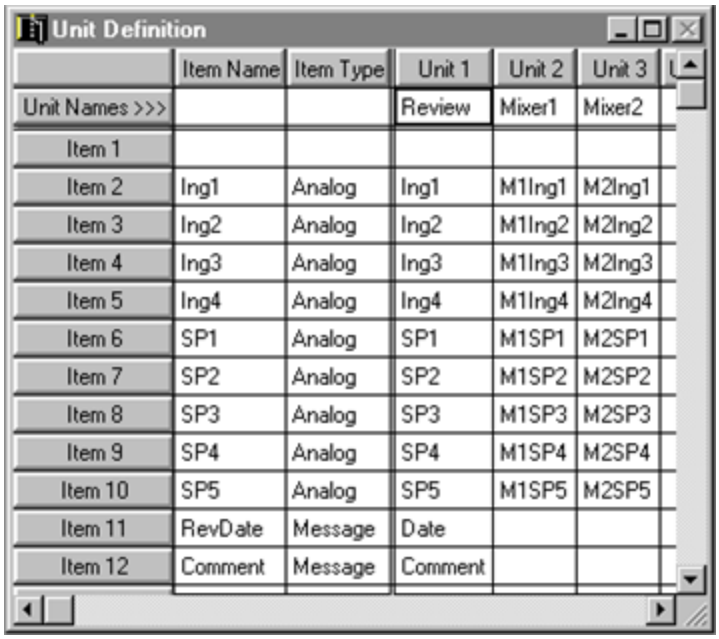
Feature	Description
Input Box	Text input box used to enter data for the selected template cell. When a template cell is selected, its contents are shown in the text input box near the top of the Recipe Manager dialog box.
Select All Cells	Select the top left cell of the template to select all cells. 
Select All Rows	Select on a template's row name to select all cells within the row.
Select All Columns	Select on a template's column heading to select all cells within the column.
Auto-Size All Columns	Double-click on a template to auto-size all columns in the template to the width of the longest entry.
Auto-Size a Column	Double-click on the heading to auto-size the column to the width of its longest entry. **Note:** The Item Type column in the Template Definition template cannot be auto-sized.

When you edit a template, you can do the following:

- Clear data from a range of cells.
- Copy data from a range of cells to an adjacent range of selected cells.
- Insert a row within the **Template Definition** template.
- Insert a column within a template.
- Delete a row from the **Template Definition** template.
- Delete a column from a template.

Clear data from a range of cells

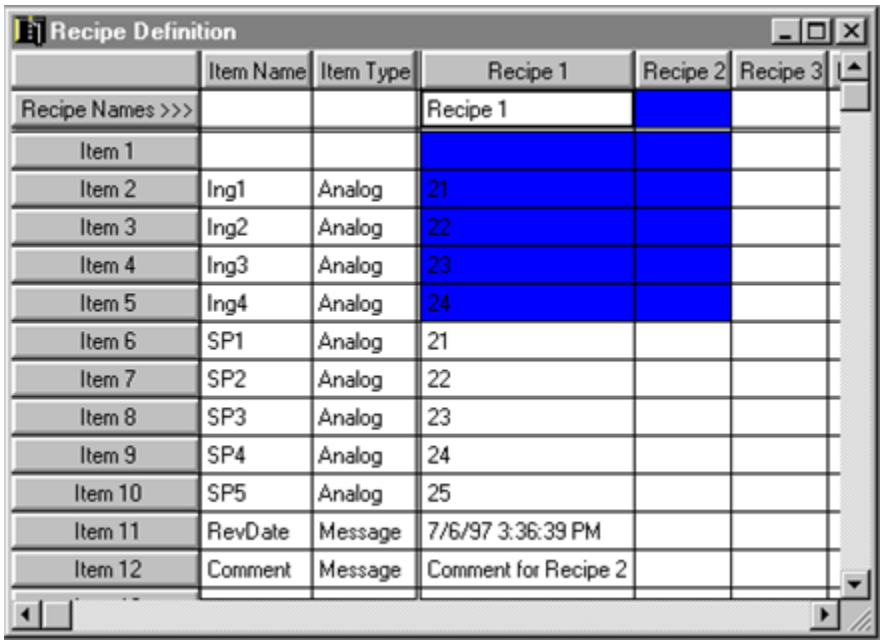
1. Select a range of cells from the template.



2. On the **Edit** menu, select **Clear**. A message appears requesting confirmation that the selected range of cells should be cleared.
3. Select **Yes**. The template clears data from the selected range of cells.

Copy a range of cells to an adjacent selected range

1. Select the cell or the range of cells to be copied.
2. Select the adjacent range of cells that you want to copy the data.



The selected ranges must be the same size and can be above, below, to the right, or to the left of the original range of selected cells.

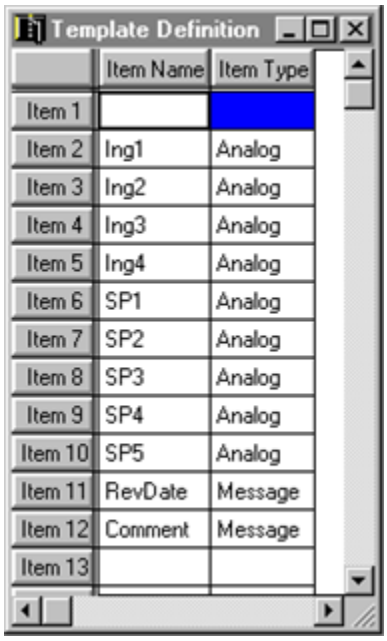
3. On the **Edit** menu, select the appropriate fill command. The data is copied to the selected range of cells.

Note: If the new column that the data is copied to is not big enough to accommodate the largest entry, double-click on the column heading to change the width to the longest entry.

Insert a row in the Template Definition template

1. Select the **Template Definition** template.
2. Select the **Item #** to select the row in the Template Definition template to insert a new row above it.

   You cannot directly insert rows in either the **Recipe Definition** or **Unit Definition** templates. Instead, all modifications to the **Template Definition** are automatically inherited by the **Recipe Definition** and **Unit Definition** templates.
3. On the **Edit** menu, select **Insert**. A new row is inserted immediately above the row you selected and all subsequent rows are automatically renumbered.

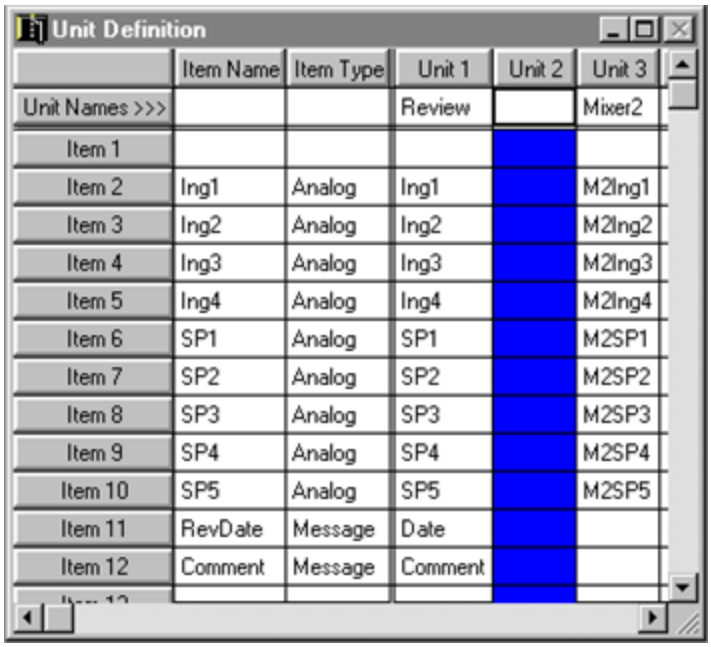


Note: If the maximum values configured for the Recipe Manager Preferences have been reached, the **Insert** command is inactive. You must increase the numbers specified to add Items/Units/ Recipes to your recipe templates.
When you modify the Preferences, the changes are applied to all existing recipe template files.

Insert a column

1. Select **Unit #** or **Recipe #** to select the column that will be to the right of the inserted column.

   You can insert columns in the Recipe Definition or Unit Definition templates.
2. On the **Edit** menu, select **Insert**. A new column is inserted to the left of the selected column.



In this example, **Mixer2** data is moved to the **Unit 3** column and a new column inserted as **Unit 2**.

Delete a column

1. Select the **Unit #** or **Recipe #** column heading to select the column that you want to delete.

   You can delete columns from the Recipe Definition or Unit Definition templates.
2. On the **Edit** menu, select **Delete**. A confirmation message dialog box appears asking you to confirm the deletion.
3. Select **Yes**. The column is deleted from the template and the remaining columns are renumbered.

Delete a row

1. Select the **Template Definition** template.

   You can delete rows from the Template Definition template, but not the **Recipe Definition** or **Unit Definition** templates.
2. Select the **Item #** row header to select the row that you want to delete.
3. On the **Edit** menu, select **Delete**. A confirmation message dialog box appears asking you to confirm the deletion.
4. Select **Yes**. The row is deleted from the template.

Define ingredient names and data types

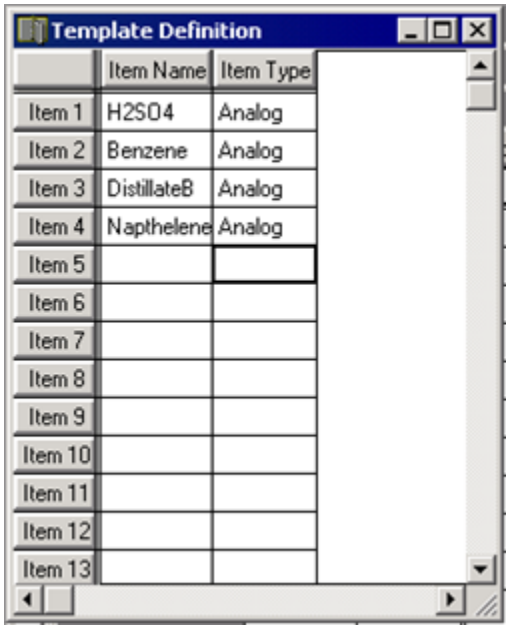
The Template Definition template lists recipe ingredients and the item type associated with each ingredient. You must complete the Template Definition template first before adding data to the other recipe templates.

To define a Template Definition template

1. Start Recipe Manager.
2. On the File menu, select **New**. The three Recipe Manager templates appear.

3. Select the **Template Definition** title bar to select the template window.
4. In the **Item Name** column cells, type the names you selected for recipe ingredients.

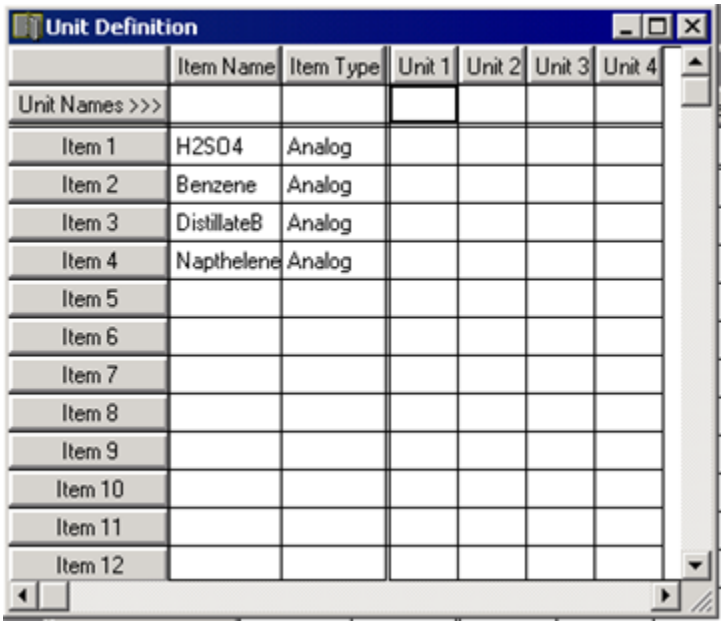
   You can only type one ingredient per cell.
5. In the **Item Type** column cells, type a valid item type for the respective recipe ingredient.



Valid item types are; analog, discrete, or message. Type A for analog, D for discrete, or M for message. Recipe Manager automatically completes the remainder of the item type when you press **Enter**.

Map InTouch tags to ingredients

The **Unit Definition** template associates InTouch tags with recipe ingredients for given units. As shown in the following figure, the first two columns of the **Unit Definition** template list the Item Names and Item Types from the **Template Definition** template.

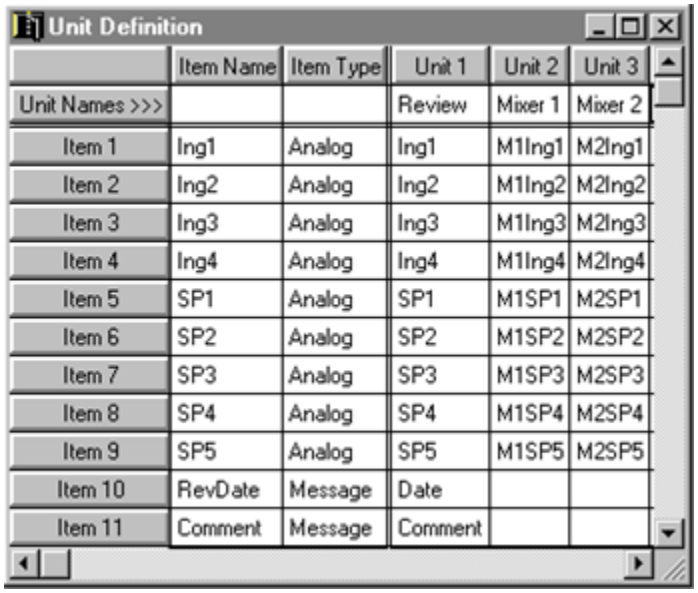


The tags defined for a unit can be memory tags or remote tags that obtain PLC data from an I/O Server.

When you use the RecipeLoad() function in an InTouch QuickScript, you must specify a Unit Name. The values contained in that Recipe Name definition are then loaded into the tags specified in the Unit Name when the QuickScript runs.

Define a Unit Definition template

1. Select the **Unit Definition** template's title bar to select the template window.



	Item Name	Item Type	Unit 1	Unit 2	Unit 3
Unit Names >>>			Review	Mixer 1	Mixer 2
Item 1	Ing1	Analog	Ing1	M1Ing1	M2Ing1
Item 2	Ing2	Analog	Ing2	M1Ing2	M2Ing2
Item 3	Ing3	Analog	Ing3	M1Ing3	M2Ing3
Item 4	Ing4	Analog	Ing4	M1Ing4	M2Ing4
Item 5	SP1	Analog	SP1	M1SP1	M2SP1
Item 6	SP2	Analog	SP2	M1SP2	M2SP2
Item 7	SP3	Analog	SP3	M1SP3	M2SP3
Item 8	SP4	Analog	SP4	M1SP4	M2SP4
Item 9	SP5	Analog	SP5	M1SP5	M2SP5
Item 10	RevDate	Message	Date		
Item 11	Comment	Message	Comment		

2. In the **Unit Names** row, type the name of each unit that you want to define.
3. In the **Unit #** column cells, use one of the following methods to enter the name of the InTouch tag for each respective recipe ingredient:
   - Type the tag name.
   - If WindowMaker is running, double-click the cell to display the **Select Tag** dialog box. Then, double-click the desired tag to insert it into the cell or select it, and then select **OK**.
4. Repeat this procedure for each Unit/Recipe combination.

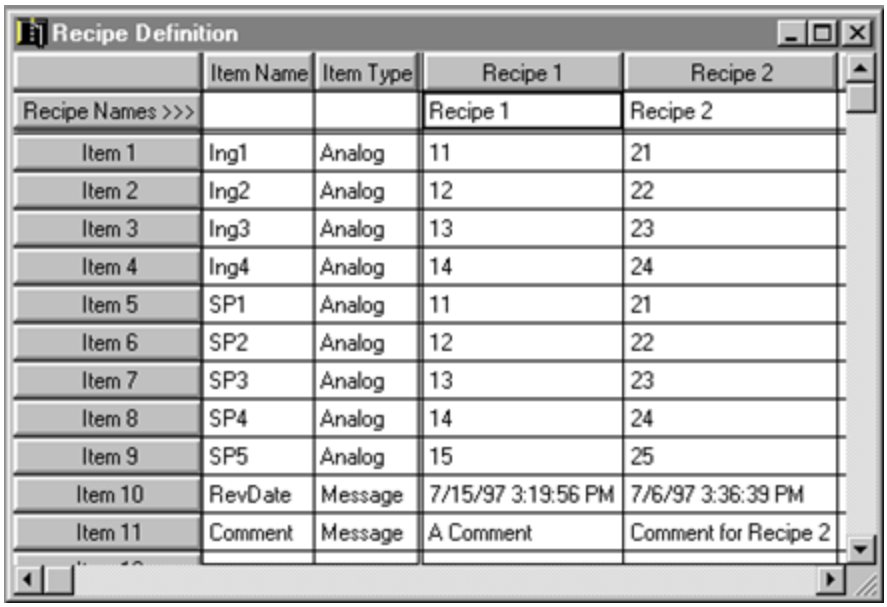
Define values for ingredients in different recipes

The Recipe Definition template specifies the name of each recipe and ingredient quantities used by the recipe. The Recipe Definition template displays the Item Name and Item Type information from the previously defined Template Definition template.

Ingredient values are loaded into the InTouch tags when the **RecipeLoad()** function is executed in an InTouch QuickScript.

Define a Recipe Definition template

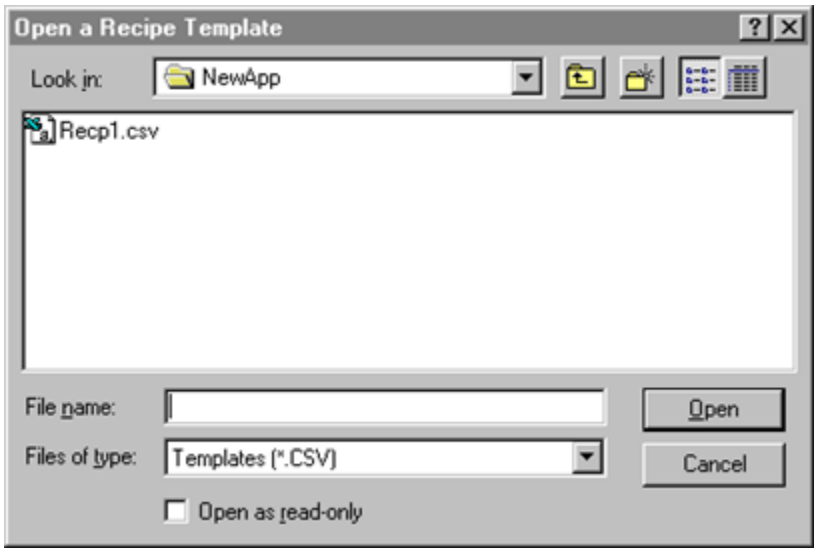
1. Select the **Recipe Definition** template's title bar to select the template window.
2. In the Recipe Names row, type the name of each recipe that you want to define.



3. In the **Recipe #** column cells, type the values for each respective recipe ingredient in the Item Name column.
4. On the **File** menu, select **Save** to save your recipe template file.

Open an existing recipe template file

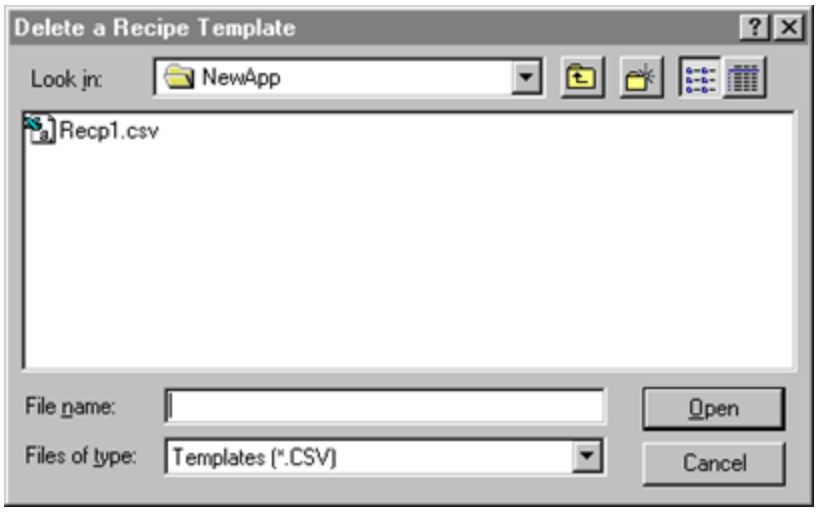
1. Open **Recipe Manager**.
2. On the **File** menu, select **Open**. The **Open a Recipe Template** dialog box appears.



3. Locate and select the Recipe file, then select **Open**. The three recipe templates in the file appear within Recipe Manager for editing.

Delete a recipe template file

1. On the **File** menu, select **Delete**. The **Delete a Recipe Template** dialog box appears.



2. Locate and select the recipe file, then select **Open** or double-select the file name. A message box appears asking you to confirm the deletion.

   Note: Open recipe template files cannot be deleted.

3. Select **Yes** to delete the file.

Edit recipe data in other applications

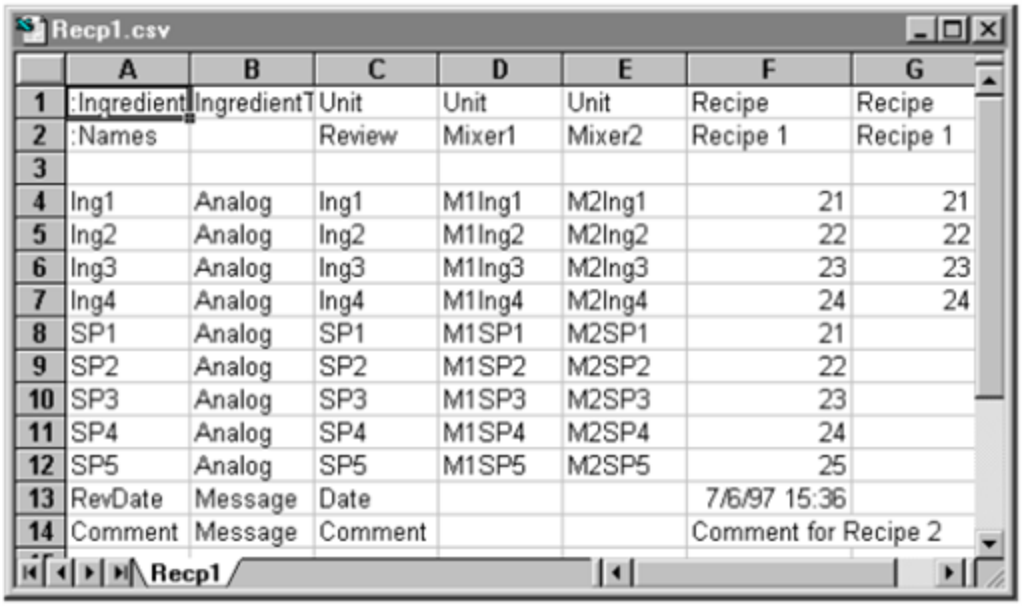
You can create and edit your recipe template definitions with any program that supports comma delimited data. You can use Microsoft Excel or Notepad to create and edit the Recipe Manager template file.

Use Excel with a recipe template file

You can use Excel to create or edit a recipe template if you do not want to use the Recipe Manager utility. You must save the Recipe Manager template created or edited with Excel to a file with a .csv file name extension.

Open an existing recipe template file in Microsoft Excel

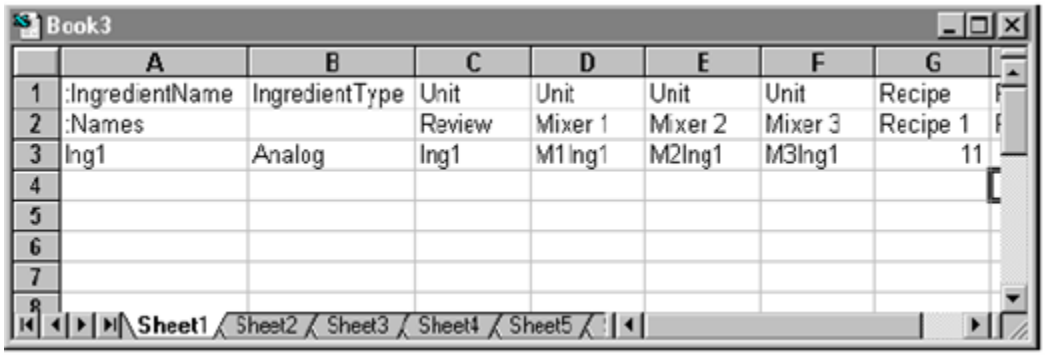
1. Start Excel.
2. On the **File** menu, select **Open**. The **Open** dialog box appear.
3. Locate and select the .csv file then, select **Open** or, double-click the file name. Excel shows the contents of the file.



4. Edit the contents of the recipe file and save your changes.

Create a new recipe template file in Excel

1. Start Excel.
2. Create a new workbook.
3. Enter recipe data in the spreadsheet, as shown in the following figure.



The entries must be made in the order shown in the figure. Unit Names must be defined in columns to the left of columns containing Recipe Names.

4. Save the spreadsheet with a .csv file name extension.

Use Notepad with a recipe template file

You can use Notepad to create or edit a recipe template if you do not want to use the Recipe Manager utility. You must save the Recipe Manager template created or edited with Notepad to a file with a .csv file name extension.

Open an existing recipe template file in Notepad

1. Start Notepad.
2. On the **File** menu, select **Open**. The Open dialog box appear.

3. Locate and select the recipe file, then select **Open** or double-click the file name.
4. Edit the contents of the recipe file and save your changes.

Create a new recipe template file in Notepad

1. Start Notepad.
2. On the **File** menu select **New**.
3. Enter following data in this format:

```
:IngredientName,IngredientType[,Unit]...[,Recipe]...
:Names,,[,UnitName]...[,RecipeName]...
IngredientName,{Analog,Discrete,Message},[,tag]...[,value]
```

Note: All Unit Names must be defined in the file before any Recipe Names are defined.

4. Save the file with a .csv file name extension.

Nest recipes to create complex structures

Multiple recipe template files can be linked to each other with InTouch QuickScripts to create complex applications. You link recipe template files by defining an ingredient name that is associated with a message tag in the Unit Name template. Then, you load the message tag with the name of a recipe.

Linking recipe template files allows you to create master recipe template files that define machine configuration parameters used by various recipes in different recipe files. Keeping this type of information in one central file greatly reduces the time it takes to maintain and update data whenever it changes.

In the following figure, the Item Name Setup tag is defined as a message type and the units contain the Setup message tag for this item. Each recipe contains a second recipe name defined in a different recipe file that is loaded into the Setup tag when the recipe is selected.

RECFILEA.CSV

	1	2	3	4	5	6	7	8	9
1	:Item Name	Item Type	Unit	Unit	Unit	Unit	Recipe	Recipe	Recipe
2	:Names		**Review**	Mixer 1	Mixer 2	Mixer 3	Recipe 1	**Recipe 2**	Recipe 3
3	Ing1	Analog	Ing1	M1Ing1	M2Ing1	M3Ing1	11	21	31
4	Ing2	Analog	Ing2	M1Ing2	M2Ing2	M3Ing2	12	22	99
5	Ing3	Analog	Ing3	M1Ing3	M2Ing3	M3Ing3	13	23	66
6	Ing4	Analog	Ing4	M1Ing4	M2Ing4	M3Ing4	14	24	34
7	SP1	Analog	SP1	M1SP1	M2SP1	M3SP1	11	21	31
8	SP2	Analog	SP2	M1SP2	M2SP2	M3SP2	12	22	32
9	SP3	Analog	SP3	M1SP3	M2SP3	M3SP3	13	23	33
10	SP4	Analog	SP4	M1SP4	M2SP4	M3SP4	14	24	34
11	SP5	Analog	SP5	M1SP5	M2SP5	M3SP5	15	25	35
12	**Setup**	Message	**Setup**	LinkFile	LinkFile	LinkFile	Setup2A	**Setup3A**	Setup1A
13									

To do so, the following script would be entered:

```
RecipeName="Recipe2";
RecipeLoad("c:\recipe\recfilea.csv", "Review", RecipeName);
```

When this script runs, the value of the **Setup** tag becomes Setup3A and is loaded into the Review unit. The value of the Setup tag is then used as the Recipe Name in the next recipe loading that loads the machine setup parameters into the tags defined for the PLC1 unit by running the following script:

```
RecipeLoad("c:\recipe\machine.csv", "PLC1", Setup);
```

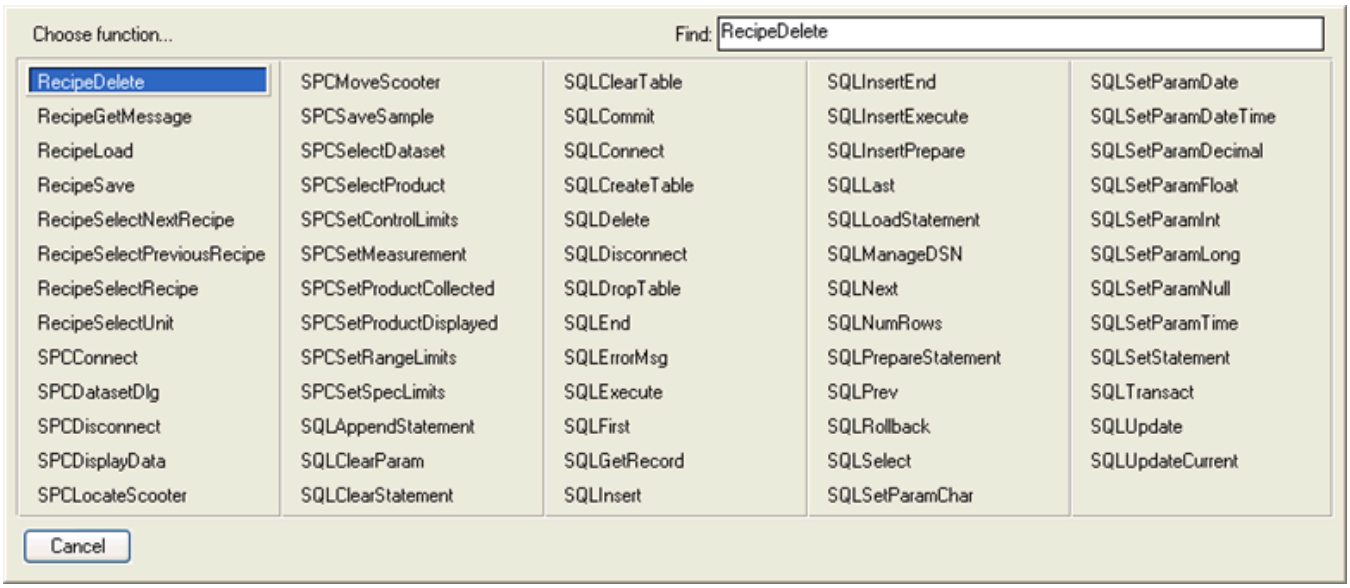
MACHINE.CSV

	1	2	3	4	5	6	7	8
1	:Item Name	Item Type	Unit	Recipe	Recipe	Recipe		
2	:Names		PLC1	Setup1A	Setup2A	Setup3A		
3	PARM1	Analog	PARM1	11	21	31		
4	PARM2	Analog	PARM2	12	22	99		
5	PARM3	Analog	PARM3	13	23	66		
6	PARM4	Analog	PARM4	14	24	34		
7	PARM5	Analog	PARM5	11	21	31		
8	PARM6	Analog	PARM6	12	22	32		
9	PARM7	Analog	PARM7	13	23	33		
10	PARM8	Analog	PARM8	14	24	34		
11	PARM9	Analog	PARM9	15	25	35		
12								

Use recipes in InTouch

You use InTouch QuickScripts to interact with your recipe template files. Recipe Manager includes a set of script functions that can be inserted into QuickScripts. Using scripts containing these functions, you can select, modify, insert, or delete records in your recipe template file.

You add recipe functions to any type of script using the InTouch script editor. The following figure shows the InTouch script editor's window listing recipe functions. All recipe functions are identified by the prefix Recipe as part of the function name.



InTouch recipe functions read and write directly to the recipe template file. Therefore, the Recipe Manager program does not need to be running in order for the recipe functions to run properly in InTouch QuickScripts.

If the recipe template file is being used by the InTouch HMI, any new recipes you create or any changes you make to existing recipes cannot be written to the recipe template file. Recipe Manager only creates recipe template files. After you create Recipe template files, close Recipe Manager.

Automatically insert a recipe function into a script

1. Open an InTouch script editor.
2. Place the cursor within the script where you want to insert a recipe function.

3. Under **Functions**, select **Add-ons**. The Choose function dialog box appears.
4. Select the recipe function that you want to insert into your QuickScript. The dialog box closes and the function is inserted at the cursor position.

Load and Save Recipe Data From/to a Recipe File

Recipe Manager provides separate InTouch QuickScript functions to load and save recipe data within a recipe file.

RecipeLoad() function

The RecipeLoad() function loads data from a recipe to a specific unit of tags in an InTouch application.

Category

Recipe

Syntax

```
RecipeLoad("Filename","UnitName","RecipeName");
```

Arguments

FileName

Name of the recipe template file. The value associated with FileName can be a string constant or a message tag containing the name of the recipe template file.

UnitName

Name of the specific unit in the designated recipe template file. The RecipeSelectUnit() function returns a value to this parameter. The value associated with UnitName can be a string constant or a message tag that contains the name of the unit.

RecipeName

Name of the specific recipe in the designated recipe template file. The value associated with *RecipeName* can be a string constant or a message tag that contains the name of the recipe.

Example

The following statement loads the values of the Recipe1 recipe to the tags defined for Unit:

```
RecipeLoad("c:\recipe\recfile.csv", "Unit1", "Recipe1");
```

RecipeSave() function

The RecipeSave() function saves a new recipe or changes made to an existing recipe to a specified recipe

template file.

Category

Recipe

Syntax

```
RecipeSave("Filename","UnitName","RecipeName");
```

Arguments

FileName

Name of the recipe template file. The value associated with FileName can be a string constant or a message tag containing the name of the recipe template file.

UnitName

Name of the specific unit in the designated recipe template file that will be used by the function. The RecipeSelectUnit() function returns a value to this parameter. The value associated with UnitName can be a string constant or a message tag that contains the name of the unit.

RecipeName

Name of the specific recipe in the designated recipe template file. The value associated with *RecipeName* can be a string constant or a message tag that contains the name of the recipe.

Example

The following example saves changes made to the Recipe3 recipe in the recfile.csv file. If Recipe3 does not currently exist in the recfile.csv file, it is created. The values set the values of the tags defined for Unit2:

```
RecipeSave("c:\recipe\recfile.csv", "Unit2", "Recipe3");
```

Delete recipes from a recipe file

Use the **RecipeDelete** function to delete a recipe from a specified recipe template file.

RecipeDelete() function

The **RecipeDelete** function deletes a recipe from a specified recipe template file.

Category

Recipe

Syntax

```
RecipeDelete("Filename","RecipeName");
```

Arguments

FileName

Name of the recipe template file. The value associated with FileName can be a string constant or a message tag containing the name of the recipe template file.

RecipeName

Name of the specific recipe in the designated recipe template file. The value associated with *RecipeName* can be a string constant or a message tag that contains the name of the recipe.

Example

The following statement deletes the Distlt1 recipe from the recfile.csv file:

```
RecipeDelete("c:\recipe\recfile.csv", "Distlt1");
```

Select Units (tag ingredient mappings)

Use the RecipeSelectUnit() function to select the unit of tags to which the current recipe values are loaded.

RecipeSelectUnit() function

The RecipeSelectUnit() function opens the **Select a Unit** dialog box so that the run-time user can select a unit. The selected unit name is returned to a message tag. Both the RecipeSelectRecipe() and RecipeSelectUnit() functions are used in conjunction with the RecipeLoad() function.

Category

Recipe

Syntax

```
RecipeSelectUnit("Filename","UnitName",Number);
```

Arguments

FileName

Name of the recipe template file. The FileName argument can be a string constant or a message tag containing the name of the recipe template file.

UnitName

Message tag to which the name of the selected unit is written. Actual message tag without quotes or a string

literal.

Number

Maximum string length returned to the argument. In InTouch, string (message) tags have a maximum length of 131 characters. Use 131 for this argument unless you have reduced the maximum string length of the InTouch tag. Number or integer tag.

Example

The following statement causes the **Select a Unit** dialog box to open:

```
RecipeSelectUnit("c:\recipe\recfile.csv", UnitName, 131);
```

After a Unit is selected, its name is returned to the UnitName tag.

Select individual recipes from a recipe file

Recipe Manager includes a set of functions to select an individual recipe from the recipe file. When you use these functions in a script, you can select a recipe from the file by its name or the next or previous recipe in sequence within the file.

RecipeSelectRecipe() function

The RecipeSelectRecipe() function opens the **Select a Recipe** dialog box so that the run-time user can select a recipe. The selected recipe name is returned to a message tag.

Category

Recipe

Syntax

```
RecipeSelectRecipe("Filename","RecipeName", Number);
```

Arguments

FileName

Name of the recipe template file. The FileName argument can be a string constant or a message tag containing the name of the recipe template file.

RecipeName

Message tag to which the name of the selected recipe is written. Actual message tag without quotation marks or a string literal.

Number

Maximum string length returned to the argument. InTouch message tags have a maximum length of 131 characters. Use 131 for this parameter unless you have reduced the maximum string length of the InTouch tag. Number or Integer tag.

Example

The following statement causes the Select a Recipe dialog box to open:

```
RecipeSelectRecipe("c:\recipe\recfile.csv", RecipeName, 131);
```

After a recipe is selected from the dialog box, its name is returned to the **RecipeName** tag.

RecipeSelectNextRecipe() function

The RecipeSelectNextRecipe() function selects the next recipe in the recipe template file.

Category

Recipe

Syntax

```
RecipeSelectNextRecipe("Filename", "RecipeName", Number);
```

Arguments

FileName

Name of the recipe template file. The FileName argument can be a string constant or a message tag containing the name of the recipe template file.

RecipeName

Message tag that contains the recipe name to use as a starting point (before the function is executed) and the selected recipe name (after the function is executed). Actual message tag without quotation marks or a string literal.

Number

Maximum string length returned to the argument. In InTouch, string (message) tags have a maximum length of 131 characters. Use 131 for this parameter unless you have reduced the maximum string length of the InTouch tag. Number or integer tag.

Example

The following statement reads the current value of the tag RecipeName and returns the next recipe on file. If the value of RecipeName is blank or cannot be found, the first recipe on file is returned. If RecipeName currently contains the last Recipe Name on file, it is returned unchanged. Recipes are saved in the recipe template file in the order in which they are created.

```
RecipeSelectNextRecipe("c:\recipe\recfile.csv",
RecipeName, 131);
```

RecipeSelectPreviousRecipe() function

The RecipeSelectPreviousRecipe() function selects the previous recipe defined in the recipe template file.

Category

Recipe

Syntax

```
RecipeSelectPreviousRecipe("Filename","RecipeName",Number);
```

Arguments

FileName

Name of the recipe template file. The FileName argument can be a string constant or a message tag containing the name of the recipe template file.

RecipeName

Message tag that contains the recipe name to use as a starting point (before the function is executed) and the selected recipe name (after the function is executed). Actual message tag without quotation marks or a string literal.

Number

Maximum string length returned to the parameter. In InTouch, Message tags have a maximum length of 131 characters. Use 131 for this parameter unless you have reduced the maximum string length of the InTouch tag. Number or Integer tag.

Example

The following statement causes the system to read the current value of the tag RecipeName and return the previous Recipe Name on file. This returned string will be stored in RecipeName and will overwrite the current value. If the value of RecipeName is blank or cannot be found, the last recipe on file is returned. If RecipeName currently contains the first Recipe Name on file, it is returned unchanged. Recipes are saved in the order in which they are created.

```
RecipeSelectPreviousRecipe("c:\recipe\recfile.csv", RecipeName, 131);
```

Understand error messages returned by recipe script functions

You troubleshoot recipe applications using diagnostic error codes returned by a recipe function. This section includes a list of recipe function error codes and how to use the RecipeGetMessage() function to show the message associated with an error code.

The RecipeLoad() function sets the value of the analog ErrorCode tag to 0 if it is successful. If RecipeLoad() fails, it sets the ErrorCode tag to the number of the specific error condition.

To retrieve the error code of a recipe function, it must be equated to an InTouch analog tag. The following example shows a script statement to return a recipe function error code:

```
ErrorCode = RecipeLoad(FileName, UnitName, RecipeName);
```

Display error code messages

Each recipe function returns a number that represents the error condition for the function. By using the RecipeGetMessage() function in an InTouch Data Change script, the corresponding error code can be written to an analog tag and the associated error code message can be written to a message tag.

The following code example shows a Data Change script.

```
RecipeGetMessage(ErrorCode, ErrorMessage, 131);
```

This script runs automatically whenever the value of the ErrorCode tag changes. When this script runs, the RecipeGetMessage() function reads the current numeric value of the ErrorCode tag and returns the message associated with that value to the ErrorMessage tag.

The following table lists possible error codes, their corresponding error messages, and descriptions:

Value	Error Message	Description
0	Success	The called recipe function executed successfully.
-1	No Such Recipe Template	The specified recipe template file does not exist.
-2	View Not Active	The recipe function called by another program cannot execute because WindowViewer is not running.
-3	Out of Memory	There is not enough memory to complete the current activity.
-4	Line too long in recipe template file	A line in the recipe template file exceeds the maximum allowed length.
-5	Truncated line in the recipe file	A line in the recipe template file is truncated.
-6	Not a valid recipe template file	The specified file is not a valid recipe template file.
-7	Expecting "unit" or "recipe"	A unit name or recipe name is missing from the recipe template file.
-8	No units defined in recipe template file	No units have been defined in the Units Definition template of the recipe file.

Value	Error Message	Description
-9	Recipe name not found in recipe template file	The specified recipe is not defined in the recipe template file.
-10	Unit name not found in recipe template file	The specified unit name is not defined in the unit definition template file.
-12	Expecting "Analog", "Discrete", "Message"	An incorrect type has been entered for an item in the recipe template file. Valid types are analog, discrete or message.
-13	Type of tag mismatches "Analog", "Discrete", "Message"	The specified tag does not match the item type. For example, a recipe item is defined as analog and a message tag has been defined as the unit for it.
-14	Invalid discrete value, expecting "0", "1"	An incorrect value has been entered for a discrete tag in the recipe template file. The only valid values for discrete tags are 0 or 1.
-15	Unable to open temporary file	The temporary file cannot be opened possibly because of insufficient free disk space.
-16	Write error while saving recipe template file	An error occurred while saving the recipe template file.
-17	User did not select	The user selected Cancel in the Select a Recipe dialog box instead of a recipe name.
-19	Recipe template in use by another application	The recipe template file specified is open and, therefore, cannot be accessed by WindowViewer.

RecipeGetMessage() function

The RecipeGetMessage() function takes an error number (returned by some other recipe function) and returns the plain text error message for that error number.

Category

Recipe

Syntax

```
RecipeGetMessage(Analog_Tag,Message_Tag,Number);
```

Arguments

Analog_Tag

Error number for which you want to get the error message.

Message_Tag

Actual message tag with no quotes or string literal.

Number

The *Number* argument sets the maximum length of the string returned with the *Message_Tag* argument. By default, InTouch message tags are set to the maximum length of 131 characters. Use 131 for this parameter unless you have reduced the maximum string length of *Message_Tag* in the InTouch Tagname Dictionary. The *Number* argument can be a constant or an integer tag containing a number.

Example

By using the RecipeGetMessage() function in an InTouch Data Change QuickScript, the error code is written to the ErrorCode tag and the associated error code message is written to the ErrorMessage tag:

```
Data Change Script Tagname[.field]:ErrorCode
Script body:RecipeGetMessage(ErrorCode, ErrorMessage,131);
```

This QuickScript automatically runs whenever the value of the ErrorCode tag changes. When this QuickScript runs, the RecipeGetMessage() function reads the current numeric value of the **ErrorCode** tag and returns the message associated with that value to **ErrorMessageTag**.

```
ErrorCode = RecipeLoad ("c:\App\recipe.csv","Unit1","cookies");
RecipeGetMessage(ErrorCode, ErrorMessageTag, 131);
```

Apply security to recipes

Access to recipes can be controlled by defining an Item Name in the recipe template file that sets the minimum security access level required to load, save, or delete a recipe.

In the following file sample, the SecurityLevel Item Name is defined as an analog tag. The Review unit contains the SecurityLevel analog tag for this item. Each recipe defines a value that is loaded into the SecurityLevel tag when the recipe is loaded into the Review unit.

MACHINE.CSV

	1	2	3	4	5	6	7
1	:Item Name	Item Type	Unit	Unit	Recipe	Recipe	Recipe
2	:Names		**Review**	PLC1	Setup1A	Setup2A	**Setup3A**
3	PARM1	Analog		PARM1	11	21	31
4	PARM2	Analog		PARM2	12	22	99
5	PARM3	Analog		PARM3	13	23	66
6	PARM4	Analog		PARM4	14	24	34
7	PARM5	Analog		PARM5	11	21	31
8	PARM6	Analog		PARM6	12	22	32
9	PARM7	Analog		PARM7	13	23	33
10	PARM8	Analog		PARM8	14	24	34
11	PARM9	Analog		PARM9	15	25	35
12	**SecurityLevel**	Analog	**SecurityLevel**		2000	5000	**7000**

You can create a window containing an access denied message that is shown whenever your security access level is invalid for a selected recipe. To do so, the selected recipe must be loaded into a unit that contains only an analog tag to which the selected recipe's security level value is loaded for verification.

For example:

```
RecipeSelectRecipe("c:\recipe\machine.csv",MyRecipe, "131");
```

The Select a Recipe dialog box appears. After you select a Recipe Name, it is returned to the RecipeName tag and the script continues running.

```
RecipeLoad( "c:\Recipe\Machine.csv", "Review", MyRecipe );
IF SecurityLevel <= $AccessLevel THEN
   Status =RecipeLoad( "c:\Recipe\Machine.csv", "PLC1", MyRecipe );
   ELSE
   Show "Access Denied";
ENDIF;
```

When this script runs, if your access level is equal to or greater than 7000, the selected recipe's values are loaded into PLC1 unit's tags. If not, the Access Denied window appears and the recipe is not loaded into PLC1.

Work with SQL databases from InTouch

A database stores information in tables that share a common attribute or field. Structured Query Language (SQL) is the language you use to access that information in the form of a query. SQL Access Manager allows you to access, modify, create, and delete database tables with queries.

SQL Access Manager is an optional program that can be installed with InTouch. SQL Access Manager allows you to:

- Create and run complex queries. These queries can be built dynamically or saved in external files. Additionally, these queries can contain parameters that are passed to the query at run time.
- Run SQL statements supported by your database and retrieve the results from the query. You can also use stored procedures with SQL Access Manager, although not all stored procedure functions are fully supported.

SQL functions can be automatically inserted into InTouch QuickScripts by selecting on the Add-ons button within the QuickScript editor dialog. The SQL function is automatically inserted into the script at the current cursor position.

You can use SQL Access Manager to transfer data, such as batch recipes from a SQL database to an InTouch application. SQL Access Manager can also be used to transfer run-time data, alarm status, or historical data from InTouch to a database. For example, after a machine cycle is completed, a company wants to save several sets of data, each for a different application. SQL databases provide the ability for information to be transferred between one or more third-party applications easily. SQL Access Manager allows this data to be accessed and displayed in any InTouch application.

SQL Access Manager consists of a program and a set of SQL functions. The SQL Access Manager program creates and associates database columns with InTouch tags. The process of associating database columns and InTouch database tags is called binding. Binding the InTouch database tags to database columns allows SQL Access Manager to directly manipulate InTouch data stored in a database.

SQL Access Manager saves the database field names and their associations in a comma-separated variable file named SQL.DEF. This file resides in the InTouch application folder and can be viewed or modified using SQL Access Manager or any text editor, such as Notepad. SQL Access Manager also creates Table Templates that define the structure and format of the database used with InTouch.

SQL functions can be used in scripts to automatically run based on operator input, a tag value changing, or when a particular set of conditions exist. These functions allow you to select, modify, insert, or delete records in the tables you choose to access. For example, if an alarm condition exists, the application can run a script that includes the SQLInsert() or SQLUpdate() functions that save all of the applicable data points and the state of the alarm.

Set up a data source

SQL Access Manager is an ODBC-compliant application that communicates with any database that supports an available ODBC driver or an OLE DB provider.

You can configure the connection string to the database by several methods:

- Use the Microsoft ODBC Administrator program to configure the ODBC driver for use with SQL Access Manager.
- Run the SQLConnect() function and specify an OLE DB provider with argument values. For more information, see SQL server database applications.
- Set the database connection string with an external UDL file.

Configure an ODBC driver

1. Run the Microsoft ODBC Administrator program.
2. Select a driver or data source, and then select **Add New Name**, **Set Default** or **Configure**. The **ODBC Driver Setup** dialog box appears.

Option	Description
Data Source Name	User-defined name that identifies the data source.
Description	User-defined description of the data source.
Database Directory	Identify the folder that contains database files. If none is specified, the current working folder is used.

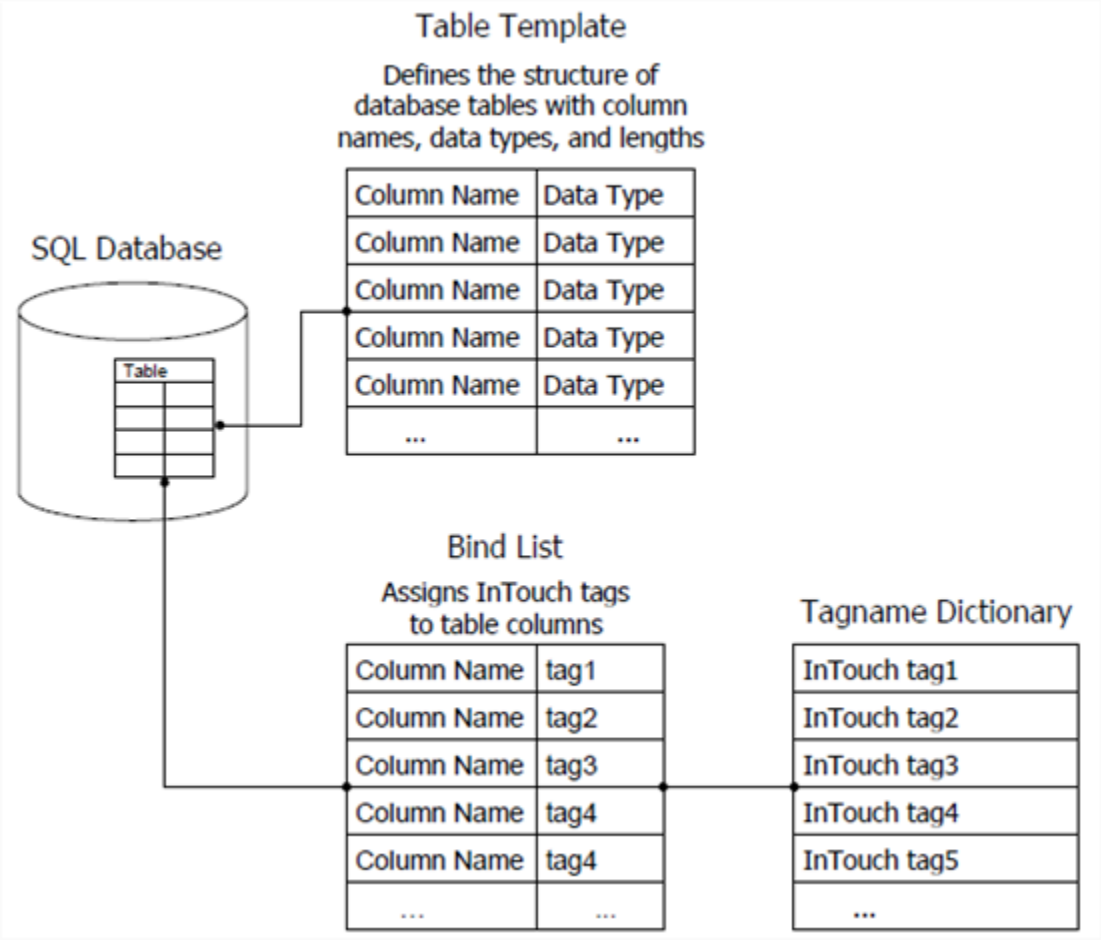
3. Select **OK**.

Note: When you create an ODBC Data Source, an ODBC.INI file is created in the Windows directory. You can manually edit the ODBC.INI file.

Map InTouch tags to database columns

You can map InTouch Tags to database columns. This is done with a Bind List. Most SQL Access functions use the Bind List to enable InTouch tags to access data in SQL database tables.

A Bind List associates database table columns to tags in the InTouch Tagname Dictionary. A Bind List also includes a Table Template that describes the format of the database tables.



When you run a script containing the SQLInsert(), SQLSelect() or SQLUpdate() functions, the Bind List is updated to specify which InTouch tags are used and which table columns are associated with these tags.

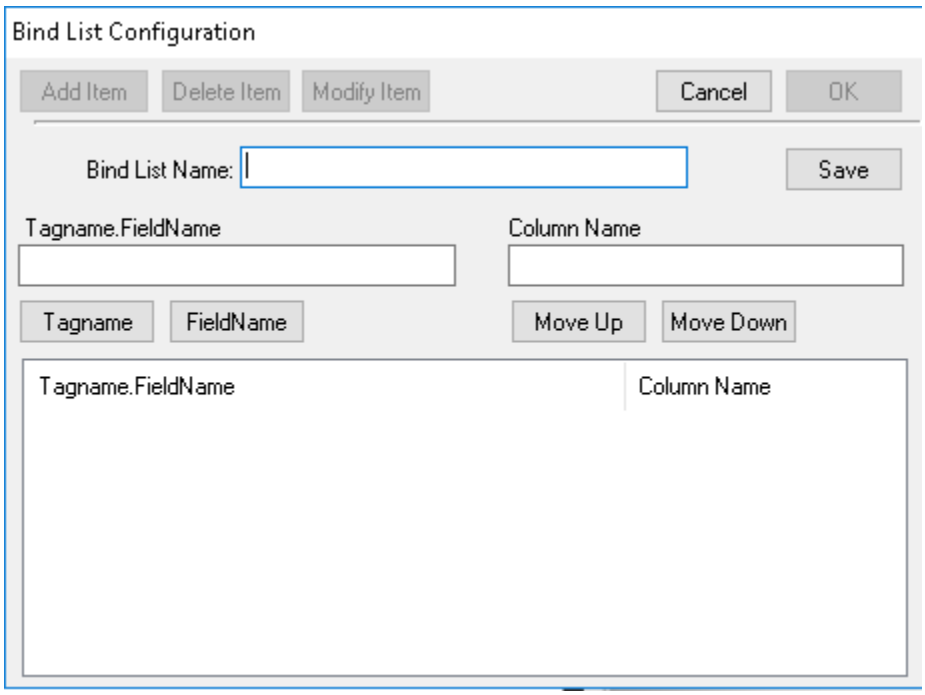
Create a new Bind List

1. In the **Tools** pane, expand **SQL Access Manager**, and select **Bind List**.

   A message to confirm the creation of the SQL.DEF file appears.
2. Select **Yes** to create the SQL.DEF file.

   The **Select a Bind List** dialog box appears.
3. Select **New**.

   The **Bind List Configuration** dialog box appears.



4. In the **Bind List Name** box, type the Bind List Name.

   A Bind List name can be up to 32 characters.
5. To define the tags for the Bind List, do one of the following:
   - In the **Tagname.FieldName** box, type an InTouch tag name. You can also add an optional tag dotfield in the form *tag_name.dotfield_name*.
   - Double-select **Tagname** to select an existing tag. The **Select Tag** dialog box appears. Select a tag from the list.

     Note: I/O type tags that are not used in your application, but are specified in a SQL Access Bind List, are activated (advised to the DAServer) as soon as WindowViewer starts.
6. Select the dotfield to append to the tag by one of the following:
   - In the **Tagname.FieldName** box, type a period and the dotfield name after the tag name
   - Select **FieldName**. The Choose field name dialog box appears. Select the dotfield that you want to append to the tag.
7. In the **Column Name** box, type the name of the column.

   A column name can be up to 30 characters in length. If the column name has a space, use square brackets

around the column name in the Bind List and when used in a script. For example:

```
WHERE EXPR= "[Valve ID] = " + text (tagname,"#");
```

8. Position the tag within the Bind List by doing one of the following:
   - Select **Move Up** to move the selected tag up one level in the list.
   - Select **Move Down** to move the selected tag down one level in the list.
9. Select **Add Item** to add your new Tagname.FieldName and Column Name to the Bind List.
10. Select **OK** to save your new Bind List configuration and close the dialog box.

Configure the SQL server string delimiter in Bind Lists

The SQLInsert() and the SQLUpdate() functions use a default format that encloses message strings within single quotation marks. Some SQL databases expect to receive message strings enclosed by another type of delimiter. For example, Oracle 8.0 expects to receive a date string surrounded by brackets. When this occurs, the Delim() function must be used.

In the Bind List Configuration dialog box Column Name field, after the column name, use the delim() function. The keyword "delim" must be entered followed by:

- a left parenthesis
- the left delimiter
- the list separator character defined in the system's regional settings
- the right delimiter
- a right parenthesis

Example for an English system: `datestring delim (',')`

Example for a German system: `datestring delim (';')`

To use the same delimiter for both left and right, specify the delimiter in parentheses without the separator, as shown in the following example:

```
datestring delim (' ')
```

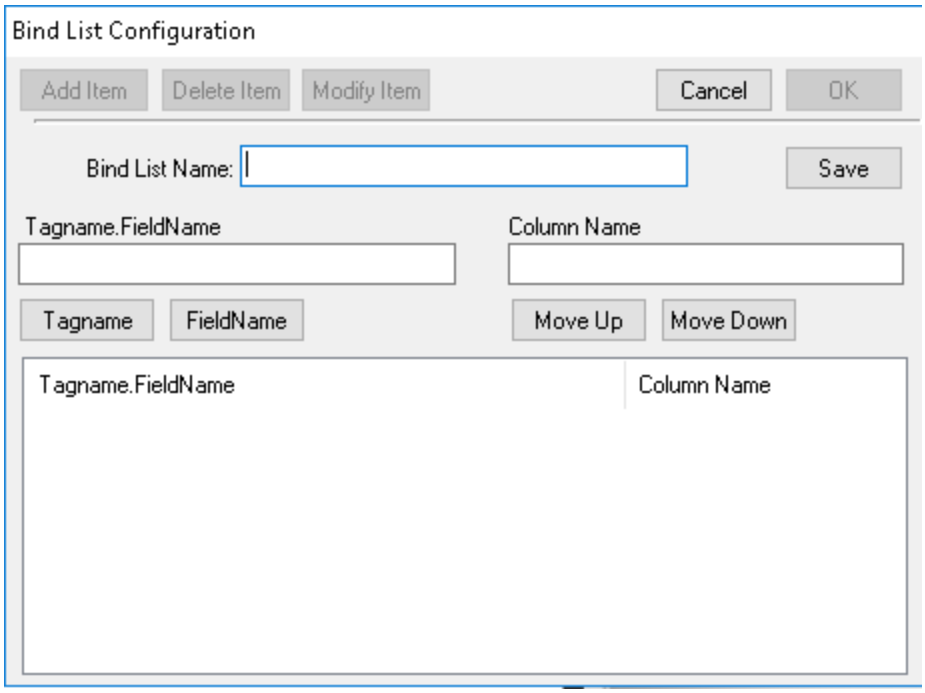
Modify a Bind List

1. In the **Tools** pane, expand **SQL Access Manager**, and select **Bind List**.

   A message to confirm the creation of the SQL.DEF file appears.
2. Select **Yes** to create the SQL.DEF file.

   The **Select a Bind List** dialog box appears.
3. Select the Bind List name that you want to change, and then select **Modify**.

   The **Bind List Configuration** dialog box appears.



4. Modify the required items.
5. Select **OK** to save your changes and close the dialog box.

Modify a Bind List with Excel

SQL Access Manager saves the configuration information for the Bind Lists and table templates to the SQL.DEF file. This file is formatted as a Comma Separated Value (CSV) file.

The SQL.DEF file can be modified with any product that supports Comma Separated Value files like Excel.

The data appears in the file as follows:

```
:BindListName, BindListName
Tagname1.FieldName,ColumnName1
Tagname2.FieldName,ColumnName2
Tagname3.FieldName,ColumnName3
:TableTemplateName, TableTemplateName
ColumnName1,ColumnType,[ColumnLength],Null,Index
ColumnName2,ColumnType,[ColumnLength],Null,Index
ColumnName3,ColumnType,[ColumnLength],Null,Index
```

Delete a Bind List

1. In the **Tools** pane, expand **SQL Access Manager**, and select **Bind List**.

   A message to confirm the creation of the SQL.DEF file appears.
2. Select **Yes** to create the SQL.DEF file.

   The **Select a Bind List** dialog box appears.
3. Select the Bind List name that you want to delete.
4. Select **Delete**. A message appears requesting confirmation to delete the Bind List.
5. Select **Yes** to delete the selected Bind List.

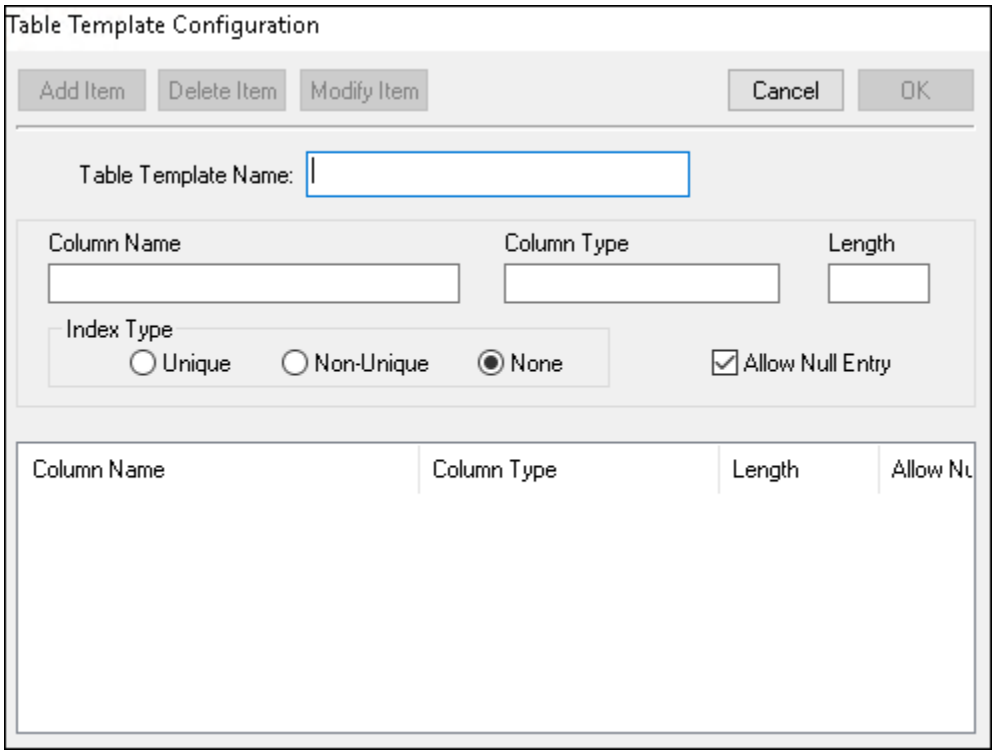
Define the structure of a new table

A Table Template defines the structure and format for new tables you create in the database. The Table Template is stored in the SQL.DEF file.

Create a new Table Template

1. In the **Tools** pane, expand **SQL Access Manager**, and select **Table Template**.
2. Select **New**.

   The **Table Template Configuration** dialog box appears.



3. In the **Table Template Name** box, type the name of the Table Template.

   A Table Template name can be up to 32 characters without an index. If you are creating an index, unique or otherwise, the Table Template name must not exceed 24 characters.
4. In the **Column Name** box, type the column name of the Table Template.

   A column name can be up to 30 characters.
5. In the **Column Type** box, type the data type for the column. Data type selections vary according to the database being used.
6. In the **Index Type** area, select one of the following options:
   - **Unique**: Each column value must be unique.
   - **Non-Unique**: Each column value is not required to be unique.
   - **None**: No index.

     Note: When you run a script containing the SQLCreateTable() function, an index file is automatically created.
7. Select **Allow Null Entry** to allow null data to be entered in this column.

Note: InTouch does not support null data.

When inserting data, if a value has not been entered for a tag, null values are assigned by the type of tag.

Data Type	Value
Discrete	0
Integer	0
Message	Strings with no characters

8. Select **Add Item** to add your new column name, column type, length, and index type to the Table Template.
9. Select **OK** to save your new Table Template configuration and close the dialog box.

Modify a Table Template

1. In the **Tools** pane, expand **SQL Access Manager**, and select **Table Template**.
   The **Select a Table Template** dialog box appears.
2. Select the Table Template name that you want to modify, and then select **Modify**.
   The **Table Template Configuration** dialog box appears.
3. Modify the required item.
4. Select **OK** to save your changes and close the dialog box.

Delete a Table Template

1. In the **Tools** pane, expand **SQL Access Manager**, and select **Table Template**.
   The **Select a Table Template** dialog box appears.
2. Select the Table Template name that you want to delete.
3. Select **Delete**. A message appears requesting confirmation to delete the Table Template.
4. Select **Yes**. The Table Template Configuration dialog box reappears.
5. Select OK to close the dialog box.

Work with database applications

SQL Access Manager supports Oracle, Microsoft SQL Server, and Microsoft Access databases. Each database's requirements are unique. This section includes separate sections that describe how to configure the connection between each database and SQL Access Manager.

SQL server database applications

You use the SQLConnect() function in an InTouch QuickScript to connect to a Microsoft SQL Server database. The SQLConnect() function logs on a user to a SQL Server database and opens a connection. The connection string used by the SQLConnect() function is formatted as follows:

(SQLConnect(*ConnectionId*,"<*attribute*>=<*value*>;
<*attribute*>=<*value*>;..."));

The *ConnectionID* argument is an integer tag containing a session number. This session number is used by almost every other SQL Access function to reference the connection to the SQL Server database. The session number increments by 1 with each SQLConnect() function call.

The following table describes the **SQLConnect()** function attributes used by Microsoft SQL Server:

Attribute	Value
Provider	SQLOLEDB
Data Source	Server name where the database is installed
Initial Catalog	Database name
User ID	Logon ID, case sensitive
Password	Password, case sensitive

```
"Provider=SQLOLEDB.1;User ID=UserIDStr; Password=PasswordStr;Initial
Catalog=DatabaseName;Data Source=ServerName;"
```

SQL Access Manager associates the four types of InTouch tags (discrete, integer, real, and message) with other SQL Server database data types.

Data Type	Length	Range	Tag Type
char	8,000 characters	1 to 131	message
int		-2,147,483,648 to 2,147,483,647	integer
float	15 digits	-1.79E+308 to 1.79E+308	real

The char data type contains fixed-length character strings. InTouch message tags require a char data type. A field length must be specified. Microsoft SQL Server databases support a char field with a maximum length of 8,000 characters. However, InTouch message tags are limited to 131 characters. If a message tag value contains more characters than the length specified for a database field, the char string is truncated when inserted into the database.

The int data type represents InTouch integer tags. If a field length is not specified, the length is set to the default value of the database. If the length is specified, it is in the form Width. The Width determines the maximum number of digits for the column.

The float data type represents InTouch real tags. The field length setting is fixed by the database. A field length for this data type is not required.

Microsoft access database applications

To communicate with Microsoft Access, you must connect to it by executing the SQLConnect() function in an InTouch QuickScript.

The SQLConnect() function is used to connect to Microsoft Access databases. Running a script containing the SQLConnect() function logs you on to the database server and opens a connection to allow other SQL functions

to be run. The connection string used by SQLConnect() is formatted as follows:

SQLConnect(*ConnectionId*,"<*attribute*>=<*value*>;
<*attribute*>=<*value*>;...");

DSN is a unique attribute used by Microsoft Access and identifies the name of the data source as configured in the Microsoft ODBC Administrator.

```
SQLConnect(ConnectionId,"DSN=MSACC");
```

The valid data types that SQL Access Manager supports depends on the version of the ODBC driver being used.

Data Type	Length	Default	Range	Tag Type
text	255 characters	--	--	Message
number	--	--	--	Integer
number	--	--	--	Real

The text data type contains fixed length character strings and are used with InTouch Message tags. A length must be specified. Microsoft Access databases support text fields with a maximum length of 255 characters. InTouch Message tags are limited to 131 characters. If a message variable contains more characters than the length specified for a database field, the string will be truncated when inserted into the database. The Microsoft Access ODBC driver supports up to 17 characters per column name. The maximum number of columns supported when using SQLSetStatement(Select Col1, Col2, ...) is 40.

Oracle database applications

To establish communication between SQL Access and an Oracle database, you must connect to it by running a script containing the SQLConnect() function.

Communicate with an Oracle 8.0 database

1. Verify the Oracle OLEDB Provider (MSDAORA.DLL) file is installed on the computer running InTouch. This file is installed by MDAC, which is installed when you install InTouch.
2. Connect to Oracle by executing the **SQLConnect()** function in an InTouch action script.

The connection string used by the **SQLConnect()** function is formatted as follows:

SQLConnect(*ConnectionId*,"<*attribute*>=<*value*>;
<*attribute*>=<*value*>;...");

The following table describes the function attributes used by Oracle:

Attribute	Value
Provider	MSDAORA
User ID	User name
Password	Password
Data Source	Oracle Server machine name

```
SQLConnect(ConnectionId, "Provider=MSDAORA; Data Source=OracleServer; User ID=SCOTT;
Password=TIGER;");
```

The following table lists the valid data types that SQL Access Manager supports for an Oracle database.

Data Type	Length	Default	Range	Tag Type
char	2,000 characters	1 character		Message
number	38 digits	38 digits		Integer

To log the date and time to an Oracle 8.0 date field, you must configure the bind list using the delim() function.

Log both date and time to an Oracle date field

1. In the Application Explorer under SQL Access Manager, double-select Bind List. The Bind List Configuration dialog box appears.
2. In the Tagname.FieldName box, type the tag that you want to use. For example, DATE_TIME_TAG.
3. In the Column Name box, type the name of the Oracle date field. If you are using Oracle 8.0, use the delim() function to specify any delimiters. The delim() function is not required if you are using Oracle 9.2 or later.
4. In your InTouch application, create a QuickScript to prepare input data from present date and time. For example:

   ```
   DATE_TIME_TAG = "TO_DATE('" + $DateString + " " + StringMid($TimeString,1,8) +
   "','mm/dd/yy hh24:mi:ss')";
   ```

   After the QuickScript runs in WindowViewer, the date appears in the following format:

   ```
   TO_DATE('08/22/06 23:32:18' ,'mm/dd/yy hh24:mi:ss')
   ```

Perform common SQL operations in InTouch

InTouch uses SQL Access functions to interact with information stored in a database. These SQL Access functions enable you to write scripts that select, modify, insert, or delete database records.

SQL actions are synchronous. When you run a database QuickScript from an InTouch application, control does not return to InTouch until the database action requested by the function is complete.

SQL Access functions adhere to punctuation standards that describe the type of arguments associated with a function. When an argument is entered in a script string surrounded by quotation marks ("Arg1") that exact string is used. If no quotation marks are used, the argument value is assumed to be a tag name and the current value of the tag is associated with the argument.

Most SQL functions return a result code. If the result code is non-zero, the function failed and other actions should be taken. The result code can be used by the SQLErrorMsg() function.

You insert SQL functions in your QuickScripts using the InTouch QuickScript editor. The general procedure to insert a SQL function into a script includes the following steps:

Add a SQL function to a script

1. Start InTouch WindowMaker.
2. Open the QuickScript with the QuickScript editor.
3. Place the cursor in the script where you want to insert the SQL function.
4. In the **Functions** area, select **Add-ons** to show the **Choose function** dialog box.

5. Select on the SQL function that you want to insert into the QuickScript. The script updates and shows the SQL function that you inserted.

The arguments associated with SQL Access functions consist of the following:

- *BindList*

  Corresponds to a Bind List name defined in the SQL.DEF file.

- *ConnectionID*

  The *ConnectionID* argument refers to the name of a memory integer tag that holds the number (ID) assigned by the **SQLConnect()** function to each database connection.

- *ConnectString*

  The ConnectString identifies the database system and any additional logon information. It is entered in the following format:

  ```
  "DSN=data source name[;attribute=value
  [;attribute=value]...]"
  ```

  Microsoft SQL Server Connection Strings

  - Microsoft OLE DB Provider for SQL Server (recommended use).

    ```
    "Provider=SQLOLEDB.1;User ID=sa; Password=;Initial Catalog=MyDB;Data
    Source=MyServer;"
    ```

    The OLE DB Provider for SQL Server is sqloledb.

  - Microsoft OLE DB Provider for SQL Server (recommended use)

    ```
    "Provider=SQLOLEDB.1;uid=sa;pwd=;Database=MyDB"
    ```

  - Microsoft OLE DB Provider for ODBC (using the default provider MSDASQL for SQL Server):

    ```
    "DSN=Pubs;UID=sa;PWD=;"
    ```

  - Microsoft OLE DB Provider for ODBC (using the default provider MSDASQL for SQL Server):

    ```
    "Data Source=Pubs;User ID=sa;Password=;"
    ```

    Oracle Connection Strings

  - Microsoft OLE DB Provider for Oracle (recommended use)

    ```
    "Provider=MSDAORA;Data Source=ServerName;User ID=UserIDStr;
    Password=PasswordStr;"
    ```

    Microsoft Access Connection Strings

    Microsoft OLE DB Provider for Microsoft Jet (recommended use). Microsoft.Jet.OLEDB.4.0 is the native OLE DB Provider for Microsoft Jet (Microsoft Access Database engine).

    ```
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:\DBName.mdb;User
    ID=UserIDStr;Password=PasswordStr;"
    ```

    Microsoft OLE DB Provider for ODBC (using the default provider MSDASQL for MS Access):

    ```
    "Provider=MSDASQL;DSN=DSNStr;UID=UserName; PWD=PasswordStr;"
    ```

- *ErrorMsg*

  Message variable containing a text description of the error.

- *FileName*

  Name of the file in which the information is contained.

- *MaxLen*

Maximum size of the column associated with a parameter. This argument determines whether the data is of varying character or long varying character type. If MaxLen is less than or equal to the largest character string allowed by the database, then the data is varying character type. If greater, then the data is long varying character type.

- *OrderByExpression*

  Defines the columns and either ascending or descending sort order. Only column names can be used to sort. The expression must be formatted:

  ColumnName [ASC|DESC]

  To sort the selected table by a column name in ascending order:

  ```
  "manager ASC"
  ```

  To sort by multi-columns, the expression is formatted:

  ColumnName [ASC|DESC],

  ColumnName [ASC|DESC]

  To sort a selected table by one column name (for example, temperature) in ascending order and another column name (for example, time) in descending order:

  ```
  "temperature ASC, time DESC"
  ```

- *ParameterNumber*

  Actual parameter number in the statement.

- *ParameterType*

  Data type of the specified parameter. Valid values are:

Type	Description
Char	Blank padded fixed length string
Var Char	Variable Length String
Decimal	BCD Number
Integer	4-byte signed integer
Small integer	2-byte signed integer
Float	4-byte floating point
Double Precision Float	8-byte floating point
DateTime	8-byte date time value
Date	4-byte date time value
Time	4-byte date time value
No Type	No data type

- *ParameterValue*

  Actual value to set.

- *Precision*

  Is the decimal value's precision, the maximum size of the character, or the length in bytes of the date-time value.
- *RecordNumber*

  Actual record number to retrieve.
- *ResultCode*

  Integer variable returned from most SQL functions. ResultCode is returned as zero (0) if the function is successful and a negative integer if it fails.
- *Scale*

  Is the decimal value's scale. This value is required only if applicable to the parameter being set to null.
- *StatementID*

  When using the advanced functionality statements, SQL returns a StatementID, which it uses internally.
- *SQLStatement*

  Actual statement, for example:

  ```
  ResultCode=SQLSetStatement(ConnectionID, "Select LotNo, LotName from LotInfo");
  ```
- *TableName*

  The *TableName* parameter contains the name of the table you want to access or create in the database.
- *TemplateName*

  The *TemplateName* parameter is the name of the template in the SQL.DEF file that defines the table.
- *WhereExpr*

  Defines a condition that can be either true or false for any row of the table. The function extracts only those rows from the table for which the condition is true. The expression must be in the following format:

  ColumnName comparison_operator expression

  Note: If the column is a character data type, the expression must be enclosed within single quotation marks.

  The following example selects all rows whose Name column contains the value EmployeeID:

  ```
  Name='EmployeeID'
  ```

  The following example selects all rows containing part numbers from 100 to 199:

  ```
  partno>=100 and partno<200
  ```

  The following example selects all records whose temperature column contains a value greater than 350:

  ```
  temperature>350
  ```

Connect and disconnect the database

Use the SQLConnect() and SQLDisconnect() functions in a script to connect to and disconnect from a SQL database.

SQLConnect() function

You use the **SQLConnect()** function in an InTouch QuickScript to connect to the database specified by the ConnectString argument.

SQLConnect() returns a value to the ConnectionID argument that is used as a parameter in all subsequent SQL functions. You must have a Bind List defined in the application folder before using the **SQLConnect** function in a script.

Category

SQL

Syntax

```
[ResultCode=]SQLConnect(ConnectionID, "ConnectString");
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

ConnectString

String that identifies the database and any additional logon information used in SQLConnect() function.

Remarks

You must have a Bind List (a SQL.DEF file) in the application folder. This function does not work without it.

If SQLTrace=1 is defined under the [InTouch] section of the win.ini file, each successful execution of **SQLConnect** logs version information for the ADO, the provider, and the database system to the Log Viewer.

Examples

The following statements connects to IBM OS/2 Database Manager and to the database named SAMPLE:

```
[ResultCode=]SQLConnect(ConnectionID,"DSN=OS2DM;
DB=SAMPLE");
```

This function returns a value to the ConnectionID variable that is used as a parameter in all subsequent SQL Functions.

```
"DSN=data source name[;attribute=value
[;attribute=value]...]"
```

SQLConnectEx () function

You use the **SQLConnectEx()** function in an InTouch QuickScript to connect to the database specified by the ConnectString argument, when using the credentials of type “Username and password” for SQL authentication.

SQLConnectEx() returns a value to the ConnectionID argument that is used as a parameter in all subsequent SQL functions. You must have a Bind List defined in the application folder before using the **SQLConnectEx** function in a script.

Category

SQL

Syntax

```
[ResultCode=]SQLConnectEx(ConnectionID, "ConnectString", "Credential Name");
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnectEx() function to each database connection.

ConnectString

String that identifies the database and any additional logon information used in SQLConnectEx() function.

Credential Name

Name of the credential stored in the Credential Manager. For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server.

Remarks

You must have a Bind List (a SQL.DEF file) in the application folder. This function does not work without it.

If SQLTrace=1 is defined under the [InTouch] section of the win.ini file, each successful execution of **SQLConnectEx** logs version information for the ADO, the provider, and the database system to the Log Viewer.

Examples

The following statements connects to IBM OS/2 Database Manager and to the database named SAMPLE:

```
[ResultCode=]SQLConnectEx(ConnectionID,"DSN=OS2DM;
DB=SAMPLE");
```

This function returns a value to the ConnectionID variable that is used as a parameter in all subsequent SQL Functions.

```
"DSN=data source name[;attribute=value
[;attribute=value]...]"
```

SQLDisconnect() function

The SQLDisconnect() function disconnects you from the database and cleans up all unreleased resources that were obtained for **SQLPrepareStatement()** and **SQLInsertPrepare()** functions.

Category

SQL

Syntax

```
[ResultCode=]SQLDisconnect(ConnectionID);
```

Argument

ConnectionId

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

See Also

SQLConnect()

Create a new table

You use the SQLCreateTable() function in an InTouch QuickScript to create a table in the database using the parameters from a specified Table Template.

SQLCreateTable() function

You use the SQLCreateTable() function in an InTouch QuickScript to create a table in the database using the parameters from a specified Table Template. Table Templates are defined in the SQL.DEF file, which includes the structure of a database table.

Category

SQL

Syntax

```
[ResultCode=]SQLCreateTable(ConnectionID, TableName,TemplateName);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

TableName

Name of the database table you want to create.

TemplateName

Name of the template definition you want to use.

Examples

The following example of the **SQLCreateTable()** function creates a table named BATCH1 with the column names and data types defined in the OutputVal template:

```
ResultCode=SQLCreateTable(ConnectionID,"BATCH1",
"OutputVal");
```

See Also

SQLConnect()

Delete a table

You use the **SQLDropTable()** function in an InTouch QuickScript to drop a table from the database.

SQLDropTable() function

You use the **SQLDropTable()** function in an InTouch QuickScript to drop a table from the database. After the QuickScript containing the **SQLDropTable()** function finishes, the table is no longer recognized and does not respond to any SQL statements.

Category

SQL

Syntax

```
[ResultCode=]SQLDropTable(ConnectionID, TableName);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

TableName

Name of the table that you want to drop from the database.

Example

The following example of the **SQLDropTable()** function drops the BATCH1 table from the database:

```
ResultCode=SQLDropTable(ConnectionID,"BATCH1");
```

See Also

SQLConnect()

Retrieve data from a table

You can use a set of SQL functions in scripts to retrieve data from a database and write the values to InTouch tags.

- The **SQLSelect()** function retrieves information from a table and places this information in the form of records into a temporary Results Table created in memory.
- The **SQLGetRecord()** function retrieves the record specified by RecordNumber from the current selection buffer.
- The **SQLNumRows()** function returns the number of table rows that met the criteria specified in a previous **SQLSelect()** function.
- The SQLFirst() function retrieves the first record of the Results Table created by the last **SQLSelect()** function.
- The SQLNext() function retrieves the next record of the Results Table created by the last **SQLSelect()** function.
- The SQLPrev() function retrieves data from the previous row of the logical table and fetch values from that row into InTouch tags.
- The SQLLast() function retrieves the last row of the logical table and fetch values from that row into InTouch tags.
- The SQLEnd() function frees memory that stores the contents of the Results Table associated with ConnectionId.

The **SQLFirst()**, **SQLPrev()**, **SQLNext()**, **SQLLast()**, and **SQLGetRecord()** functions retrieve data from specified rows of the logical table and save it as InTouch tag values. If a field is NULL, the value of the associated InTouch tag is set to zero or a zero-length string depending on whether the tag is of analog or message type.

If a string in the database is greater than 131 characters, only the first 131 characters are copied from the database to the associated InTouch message tag.

SQLSelect() function

The SQLSelect() function retrieves records from a table. When the script containing the SQLSelect() function is processed, the retrieved records are placed in a temporary Results Table in memory. These records can be browsed using the SQLFirst(), SQLLast(), SQLNext() and SQLPrev() functions.

Important: Always call the SQLEnd() function after the script containing the SQLSelect() function ends to free memory used by the Results Table.

Category

SQL

Syntax

```
[ResultCode=]SQLSelect(ConnectionID,TableName, BindList,WhereExpr,OrderByExpression);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

TableName

Name of the database table to access.

BindList

Defines which InTouch tags are used and which database.

WhereExpr

Defines a condition that can be either true or false for any row of the table. The SQLSelect() function extracts data from only those rows in which the *WhereExpr* condition is true. The expression must be in the following format:

ColumnName comparison_operator expression.

Note: If the comparison is made with a character expression, the expression must enclosed within single quotes.

The following example selects all rows whose name column contains the value EmployeeID:

```
name='EmployeeID'
```

The following example selects all rows containing part numbers from 100 to 199:

```
   partno>=100 and partno<200
```

The following example selects all rows whose temperature column contains a value greater than 350:

```
   temperature>350
```

WhereExpr - Memory message Tag

```
   OrderByExpr - Memory message Tag
   Speed_Input - Memory Real - User Input Analog
   Serial_Input - Memory Message - User Input String
```

Analog Example

```
   WhereExpr = "Speed = " + text
   (Speed_Input,"#.##");
```

Because Speed_Input is a number, it must be converted to text so it can be concatenated to the WhereExpr string.

String Example

```
   WhereExpr = "Ser_No = ‘" +
   Serial_input + "’";
```

Because Serial_Input is a string it must have single quotes around the value for example:WhereExpr = "Ser_No=’125gh’";

String Example using the like statement

```
   WhereExpr = "Ser_No like ‘-’"
```

When using the Like comparison operator the % char can be used as a wild card.

String and Analog Example using a Boolean AND operator

```
   WhereExpr = "Ser_No = ‘" + Serial_input + "’" + " and " + "Speed = " +
   text(Speed_Input,"#.##");OrderByExpr = "";
```

If the order does not matter, use a null string as shown above.

SQLSelect using WhereExpr tag

```
   ResultCode = SQLSelect(Connect_Id,TableName,
   BindList,
   WhereExpr,OrderByExpr);
   Error_msg = SQLErrorMsg( ResultCode );
```

SQLSelect WhereExpr built in function

```
   ResultCode = SQLSelect(Connect_Id,TableName,
   BindList,
   "Ser_No = ‘" + Serial_input + "’", OrderByExpr);
   Error_msg = SQLErrorMsg( ResultCode );
```

OrderByExpr

Defines the direction to sort data within a table column. Only column names can be used to sort and the expression must be in this form:

*ColumnNam*e [ASC|DESC]

The following example sorts a table in ascending order by the data from the manager column:

```
   "manager ASC"
```

You can also sort by multi-columns where the expression is in the form:

ColumnName [ASC|DESC],

ColumnName [ASC|DESC]

The next example sorts the selected table by the temperature column in ascending order and the time column in descending order:

```
   "temperature ASC,time DESC"
```

Examples

The following statement selects records from the BATCH table using a BindList named List1, whose column name type contains the value cookie. It will present the information sorted by the amount column in ascending order and the sugar column in descending order:

```
ResultCode=SQLSelect(ConnectionID,"BATCH", "List1","type='cookie'","amount ASC,sugar
DESC");
```

The following statement selects all data in the database, do not specify a value for the WhereExpr and OrderByExpr:

```
ResultCode=SQLSelect(ConnectionID,"BATCH", "List1", "","");
```

See Also

SQLFirst(), SQLConnect(), SQLLast(), SQLNext(), SQLPrev(), SQLEnd()

SQLGetRecord() function

The SQLGetRecord() function retrieves the record specified by the RecordNumber argument from the current selection buffer.

Category

SQL

Syntax

```
[ResultCode=]SQLGetRecord(ConnectionID, RecordNumber);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

RecordNumber

Actual record number to retrieve.

Example

```
ResultCode=SQLGetRecord(ConnectionID,3);
```

See Also

SQLConnect()

SQLNumRows() function

The SQLNumRows() function indicates how many rows met the criteria specified in the last SQLSelect() function. For example, if a WhereExpr argument is used to select all rows with a column name AGE, where AGE is equal to 45, the number of rows returned could be 40 or 4000. This may determine which function is processed next.

Category

SQL

Syntax

```
SQLNumRows(ConnectionID);
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Example

The following statement returns the number of rows selected to the NumRows integer tag:

```
NumRows=SQLNumRows(ConnectionID);
```

See Also

SQLConnect()

SQLFirst() function

The SQLFirst() function selects the first record of the Results Table created by the last SQLSelect() function.

Category

SQL

Syntax

```
[ResultCode=]SQLFirst(ConnectionID);
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

See Also

SQLConnect(), SQLSelect()

SQLNext() function

The SQLNext() function selects the next record in sequence of the Results Table created by the last SQLSelect() function. A SQLSelect() function must be processed before running the SQLNext() function in a script.

Category

SQL

Syntax

```
[ResultCode=]SQLNext(ConnectionID);
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Example

```
ResultCode=SQLNext(ConnectionID);
```

See Also

SQLConnect(), SQLSelect()

SQLPrev() function

The SQLPrev() function selects the previous record of the Results Table created by the last SQLSelect() function.

Category

SQL

Syntax

```
[ResultCode=]SQLPrev(ConnectionID);
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Remarks

A SQLSelect() function must be processed before using this command.

Example

```
ResultCode=SQLPrev(ConnectionID);
```

See Also

SQLConnect(), SQLSelect()

SQLLast() function

The SQLLast() function selects the last record of the Results Table created by the previous SQLSelect() function.

Category

SQL

Syntax

```
[ResultCode=]SQLLast(ConnectionID);
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Example

```
ResultCode=SQLLast(ConnectionID);
```

See Also

SQLConnect(), SQLSelect()

SQLEnd() function

The SQLEnd() function is run after the SQLSelect() function to free memory used to store the contents of the Results Table.

Category

SQL

Syntax

```
[ResultCode=]SQLEnd(ConnectionID);
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each

database connection.

See Also

SQLConnect(), SQLSelect()

Write new records to a table

You can insert new records to a database using the **SQLInsert()** function. The **SQLInsert()** function uses the current value of an InTouch tag to insert one record into a table. The **SQLInsert()** function is a one step operation that prepares, inserts, and ends the statement.

If the string associated with an InTouch message tag is longer than the defined size of the corresponding text field of the table, the number of characters used from the message tag will be the defined size of the field.

Note: InTouch tags cannot be NULL. It is impossible to update or insert NULL values into the database using these functions if the Bind List includes the field. You can insert NULL values into a field using **SQLExecute** on an INSERT statement that does not include the field, which should have been defined to allow NULL values.

SQL Access provides three other functions that separately prepare, insert, and clean up after a record insertion. Using these functions together, you can write scripts that include a single prepare and end statement and add as many record insert statements as needed. If you use individual functions to insert data instead of the **SQLInsert()** function, you can reduce resource usage on the computer.

SQLInsert() function

The SQLInsert() function inserts a new record into the referenced table using the values of the tags in the supplied BindList. The BindList parameter defines which InTouch tags are used and which database columns they are associated.

Use the SQLInsert() function to prepare, insert, and end the statement.

Category

SQL

Syntax

```
[ResultCode=]SQLInsert(ConnectionID, TableName, BindList);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

TableName

Name of the database table you want to access.

BindList

Defines which InTouch tags are used and which database columns they are associated with.

Example

The following statement inserts a new record into table ORG with the tag values specified in List1:

```
ResultCode=SQLInsert(ConnectionID,"ORG","List1");
```

SQLInsertPrepare() function

The SQLInsertPrepare() function creates and prepares an Insert statement each time the function runs. The Insert statement is not processed. The StatementID argument is an integer tag containing a value after the statement is processed.

Category

SQL

Syntax

```
[ResultCode=]SQLInsertPrepare (ConnectionID, TableName, BindList, StatementID);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

TableName

The name of the database table to access.

BindList

Defines which InTouch tags are used and which database columns they are associated with.

StatementID

Integer value returned by SQL when a SQLPrepareStatement() function is used.

See Also

SQLConnect(), SQLPrepareStatement()

SQLInsertExecute() function

The SQLInsertExecute() function runs the previously prepared insert statement specified by the SQLInsertPrepare() function.

The SQLInsertExecute() function uses the current values of InTouch tags to insert one row into the table identified by the previous SQLInsertPrepare() function. If the BindList argument includes an Identity key field for a MS SQL Server table, it is necessary to set the IDENTITY_INSERT option before running SQLInsertExecute().

The *StatementID argument contains an integer* value returned by SQL when a previous SQLInsertPrepare() function is run within the script.

Category

SQL

Syntax

```
[ResultCode=]SQLInsertExecute(ConnectionID, BindList, StatementID);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

BindList

Defines which InTouch tags are used and which database columns they are associated with.

StatementID

Integer value returned by SQL when a SQLPrepareStatement() function is used.

See Also

SQLConnect(), SQLPrepareStatement()

SQLInsertEnd() function

The **SQLInsertEnd** function cleans up resources associated with the StatementID function created by **SQLInsertPrepare**.

Category

SQL

Syntax

```
[ResultCode=]SQLInsertEnd(ConnectionID, StatementID);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

StatementID

Integer value returned by SQL when a SQLPrepareStatement() function is used.

Example

The following example shows how multiple insert functions should be specified in a script.

```
ResultCode = SQLSetStatement(ConnectionId, "SET IDENTITY_INSERT Products ON");
ResultCode = SQLExecute(ConnectionId, "", 0);
ResultCode = SQLInsertPrepare(ConnectionId, TableName, Bindlist, StatementID);
ResultCode = SQLInsertExecute(ConnectionId, Bindlist, StatementID);
ResultCode = SQLInsertEnd(ConnectionId, StatementID);
```

See Also

SQLConnect(), SQLPrepareStatement()

Update existing records in a table

SQL Access provides two functions to update table records with values from InTouch tags:

- SQLUpdate()
- SQLUpdateCurrent()

SQLUpdate() function

The SQLUpdate() function uses the current values of InTouch tags to update all rows in a table that match the condition set by the WhereExpr argument.

Category

SQL

Syntax

```
[ResultCode=]SQLUpdate(ConnectionID, TableName, BindList, WhereExpr);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

TableName

The name of the database table to access.

BindList

Defines which InTouch tags are used and which database columns they are associated with.

WhereExpr

Defines a condition that can be either true or false for any row of the table. The function updates only those rows from the table for which the condition is true. The expression must be in the following format: ColumnName comparison_operator expression.

Note: If the column is a character data type, the expression must be in single quotes.

Example

The following example selects all rows whose name column contains the value EmployeeID:

```
name='EmployeeID'
```

The following example selects all rows containing part numbers from 100 to 199:

```
partno>=100 and partno<200
```

The following example selects all rows whose temperature column contains a value that is greater than 350:

```
temperature>350
```

The following statement updates all records in the table BATCH, whose lot number is 65, to the current values of the tags specified in the BindList "List1":

```
ResultCode=SQLUpdate(ConnectionID,"BATCH", "List1","lotno=65");
```

Note: Be sure that all records are unique. If identical records exist in a table, all similar records are updated.

See Also

SQLConnect()

SQLUpdateCurrent() function

The SQLUpdateCurrent() function updates the current row of the logical table using InTouch tags mapped to the table fields by the Bind List specified in SQLSelect() or SQLExecute() function statements. If there are rows that are identical to the current row, all rows are updated.

Up to 54 identical records can be updated at once. If there are too many identical rows to be updated in SQL Access, the SQLUpdateCurrent() function returns an error. The error message is similar to, "Microsoft Cursor Engine: Key column information is insufficient or incorrect. Too many rows were affected by update."

To avoid this error, create a unique key field in the table that makes each row unique. It is strongly recommended that all tables used by SQL Access have a unique key. For a table without a key, it is recommended that a field of type AutoNumber (Access) or an integer field used as the row Identity (SQL Server) be used as the primary key so that SQLUpdateCurrent() function updates only one row at a time. This primary key field does not have to be included in a Bind List.

Category

SQL

Syntax

```
[ResultCode=]SQLUpdateCurrent(ConnectionID);
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Example

```
ResultCode=SQLUpdateCurrent(ConnectionID);
```

See Also

SQLConnect()

Delete records from a table

You can use two SQL functions to remove records from a database table.

SQL Access provides two functions to delete table records:

- SQLClearTable() deletes records from a table.
- SQLDelete() deletes records from a table that match a specified condition.

SQLClearTable() function

The **SQLClearTable()** function deletes all records from a table. It does not delete the table from the database.

Category

SQL

Syntax

```
[ResultCode=]SQLClearTable(ConnectionID, "TableName");
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

TableName

Name of the table in which all records are cleared.

Example

In the following example, the SQLClearTable() function clears all records from the BATCH1 table.

```
ResultCode=SQLClearTable(ConnectionID,"BATCH1");
```

See Also

SQLConnect(), SQLClearStatement()

SQLDelete() function

The **SQLDelete()** function removes all records from a table that match a condition specified by the *WhereExpr* argument.

Category

SQL

Syntax

```
[ResultCode=]SQLDelete(ConnectionID, TableName, WhereExpr);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

TableName

Name of the table in which records are cleared that meet the condition specified by the *WhereExpr* argument.

WhereExpr

Defines a condition that can be either true or false for any row of the table. The **SQLDelete()** function deletes only those row records in which the *WhereExpr* condition is true. The expression must be in the following format:
ColumnName comparison_operator expression

Note: The SQLDelete() function cannot contain a null *WhereExpr* argument.

Example

The following statement deletes all records in the BATCH1 table whose lot number is equal to 65:

```
ResultCode=SQLDelete(ConnectionID,"BATCH1", "lotno=65");
```

Note: If the column is a character data type, the expression must be in single quotes such as "MachineID='AG_LX7_2'".

See Also

SQLConnect()

Execute parameterized statements

Use the SQLSetStatement() and the SQLAppendStatement() functions to build dynamic queries. The SQLSetStatement() function starts a new SQL statement. This can be any valid SQL statement, including the name of a stored procedure. The SQLAppendStatement() function continues a SQL statement using the contents of string.

SQLSetStatement() function

The SQLSetStatement() function starts a SQL statement buffer using the contents of SQLStatement, on the established connection, ConnectionID. There can be one SQL Statement buffer per ConnectionID. Errors are returned in the function return.

Category

SQL

Syntax

```
[ResultCode=]SQLSetStatement(ConnectionID, SQLStatement);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

SQLStatement

Actual SQL statement, see the following examples.

Examples

```
ResultCode=SQLSetStatement(ConnectionID,"Select LotNo, LotName from LotInfo");
```

In the following example, the StatementID is set to zero so the statement does not have to call SQLPrepare(Connect_Id, StatementID) before running the statement. Because the StatementID is not created by the SQLPrepare to properly end this select, use the SQLEnd() function instead of the SQLClearStatement() function.

```
SQLSetStatement( Connect_Id, "Select Speed, Ser_No from tablename where Ser_No =’" +
Serial_input + "’");
SQLExecute(Connect_Id,0);
```

In the following example, the StatementID is created by the SQLPrepareStatement() function and used in the SQLExecute() function. To end this SELECT statement, use the SQLClearStatement() function to free resources and the handle.

```
SQLSetStatement( Connect_Id, "Select Speed, Ser_No from tablename where Ser_No =’" +
Serial_input + "’");
SQLPrepareStatement(Connect_Id,StatementID);
SQLExecute(Connect_Id,StatementID);
SQLSetStatement( Connect_Id, "Select Speed, Ser_No from tablename where Ser_No =’" +
Serial_input + "’");
SQLPrepareStatement(Connect_Id,StatementID);
SQLExecute(Connect_Id,StatementID);
```

See Also

SQLConnect()

SQLAppendStatement() function

The SQLAppendStatement() function continues a SQL statement using the contents of a string. A return value indicates if an error occurred during the function call.

InTouch tags can support character strings to a maximum of 131 characters. You typically use the SQLAppendStatement() function to concatenate additional strings to a statement.

Category

SQL

Syntax

```
[ResultCode=]SQLAppendStatement(ConnectionID, "SQLStatement");
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

SQLStatement

Actual statement to append.

Example

```
ResultCode=SQLAppendStatement(ConnectionID, "where tablename.columnname=TR-773-01");
```

See Also

SQLConnect(), SQLClearStatement()

Create a statement or loading an existing statement from a file

You can create a query with other third-party database tools, and then use SQL Access to run the query. First, you must load the SQL statement from an .SQL query file created by the third-party database tool.

```
ResultCode = SQLLoadStatement (ConnectionID, "c:\myappdir\lotquery.sql");
```

You load the SQL query using the SQLoadStatement() function. The statement is now ready to run.

SQLLoadStatement() function

The SQLLoadStatement() function reads a SQL statement from a file.

There can be only one statement per file. However, SQLAppendStatement() function can be used to append something to the statement if SQLPrepareStatement() function or SQLExecute() function has not been called.

Category

SQL

Syntax

```
[ResultCode=]SQLLoadStatement(ConnectionID, FileName);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

FileName

Name of the file containing the SQL statement.

Remarks

After you load the statement and get the statement handle, use the SQLPrepareStatement() function to prepare the statement for execution.

Example

The SQL.txt file contains the following SQL statement:

```
Select ColumnName from TableName where ColumnName>100;
```

The SQLLoadStatement() function loads the statement from the file.

```
ResultCode=SQLLoadStatement(ConnectionID,
"C:\SQL.txt")
```

See Also

SQLConnect(), SQLAppendStatement(), SQLExecute(), SQLPrepareStatement

Preparing a statement

Using the following functions, you can create any parameterized statement you want, and then dynamically fill in the parameters one by one. For example, you could save a generic statement in a file, load it using the SQLLoadStatement() function, prepare it using the SQLPrepareStatement() funtion to get a statement ID, and then fill in the statement parameters using the following functions:

- **SQLPrepareStatement()**
- **SQLSetParamChar()**
- **SQLSetParamDate()**
- **SQLSetParamDateTime()**
- **SQLSetParamDecimal()**
- **SQLSetParamFloat()**
- **SQLSetParamInt()**
- **SQLSetParamLong()**
- **SQLSetParamNull()**
- **SQLSetParamTime()**
- **SQLClearParam()**
- **SQLClearStatement()**

To perform parameter substitution on a SQL statement, place a "?" in the SQL statement where you want to specify a subsequent parameter. The statement is prepared, parameters are set into the statement, and then the statement is run.

SQLPrepareStatement() function

The SQLPrepareStatement() function prepares the SQL statement to be run. It does not run the statement, it just makes the statement active so you can set parameter values.

Category

SQL

Syntax

```
SQLPrepareStatement(ConnectionId, StatementID)
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Remarks

Prepare the default statement and return a StatementID (1, 2, 3, and so on). This preparation is useful for statements with parameters that need to be set using the SQLSetParam{Type} functions.

Set Statement parameters

SQL Access Manager provides a set of functions to modify the value assigned to a parameter included in a SQL statement.

SQLSetParamChar() function

The SQLSetParmChar() function can be used in a script to set the value of the specified parameter to the specified string. The function can be called multiple times before executing, resulting in the parameter value being set to the concatenation of all values sent. Lengths of 0 (zero) are ignored.

Category

SQL

Syntax

```
SQLSetParamChar(StatementID, ParameterNumber, ParameterValue, MaxLength);
```

Arguments

StatementID

Integer value returned by SQL when a SQLPrepareStatement() function is used.

ParameterNumber

Parameter number in the statement.

ParameterValue

Value to set as the parameter value.

MaxLength

Maximum width of the column with which this parameter is associated. This setting determines whether the parameter is of varying character or long varying character type. If *MaxLength* is less than or equal to the largest character string allowed by the database, then the parameter is varying character type. If greater, long varying character type.

See Also

SQLPrepareStatement()

SQLSetParamDate() function

The SQLSetParamDate() function sets the value of a parameter to a specified date.

Category

SQL

Syntax

```
SQLSetParamDate(StatementID, ParameterNumber, "Value");
```

Arguments

StatementID

Integer value that identifies a SQL statement within a query.

ParameterNumber

Integer value that identifies the parameter in the SQL statement identified by the *StatementID* argument.

Value

Date assigned to the parameter as a literal enclosed with double quotation marks or the name of a tag whose value is a date. The time assigned to the date is 12:00:00 AM.

Example

This example sets the second parameter of the third statement to the date associated with the NewDate tag.

```
SQLSetParamDate(3, 2, NewDate);
```

See Also

SQLPrepareStatement()

SQLSetParamDateTime() function

The SQLSetParamDateTime() function sets the value of a parameter to a specified date and time.

Category

SQL

Syntax

```
SQLSetParamDateTime(StatementID, ParameterNumber, Value, Precision);
```

Arguments

StatementID
Integer value that identifies a SQL statement within a query.
ParameterNumber
Integer value that identifies the parameter in the SQL statement identified by the *StatementID* argument.
Value
Date and time to assign to the parameter identified by the ParameterNumber argument.
Precision
Integer that specifies the number of characters of the date-time value assigned as the value of the parameter.

See Also

SQLPrepareStatement()

SQLSetParamDecimal() function

The SQLSetParamDecimal() function sets the value of a parameter to a decimal number.

Category

SQL

Syntax

```
SQLSetParamDecimal(StatementID, ParameterNumber, Value, Precision, Scale);
```

Arguments

StatementID

Integer value that identifies a SQL statement within a query.

ParameterNumber

Integer value that identifies the parameter in the SQL statement identified by the *StatementID* argument.

Value

Value can be either a string or an InTouch message tag that represents a decimal number (123.456) or an InTouch memory real tag.
It is recommended that a message tag is used instead of a real tag to guarantee the precision of the parameter. However, if Value must be a floating point number (for example, a real value received from an DAServer), the function continues to work. But, high precision may not be guaranteed because of the limitation of floating point representation.

Precision

Integer that specifies the total number of digits in the number.

Scale

Integer that specifies the number of digits to the right of the decimal point.

Example

This example sets the second parameter of the third SQL statement to 123.456. The precision is six digits and the scale is three digits to the right of the decimal point.

```
SQLSetParamFloat(3, 2, 123.456, 6, 3);
```

See Also

SQLPrepareStatement()

SQLSetParamFloat() function

The SQLSetParamFloat() function sets the value of a parameter to a 64-bit, signed, floating-point value.

Category

SQL

Syntax

```
SQLSetParamFloat(StatementID, ParameterNumber, Value);
```

Arguments

StatementID

Integer value that identifies a SQL statement within a query.

ParameterNumber

Integer value that identifies the parameter in the SQL statement identified by the *StatementID* argument.

Value

64-bit, signed, floating-point number to assign as the value of the specified parameter.

Example

This example sets the second parameter of the third SQL statement to -5.

```
SQLSetParamFloat(3, 2, -5);
```

See Also

SQLPrepareStatement()

SQLSetParamInt() function

The SQLSetParamInt() function sets the value of a parameter to a 16-bit signed integer.

Category

SQL

Syntax

```
SQLSetParamInt(StatementID, ParameterNumber, Value);
```

Arguments

StatementID

Integer value that identifies a SQL statement within a query.

ParameterNumber

Integer value that identifies the parameter in the SQL statement identified by the *StatementID* argument.

Value

16-bit, signed, integer to assign as the value of the specified parameter.

Example

This example sets the second parameter of the third SQL statement to -5.

```
SQLSetParamInt(3, 2, -5);
```

See Also

SQLPrepareStatement()

SQLSetParamLong() function

The SQLSetParamLong() function sets the value of a parameter to a 32-bit signed analog number.

Category

SQL

Syntax

```
SQLSetParamLong(StatementID, ParameterNumber, Value);
```

Arguments

StatementID
Integer value that identifies a SQL statement within a query.
ParameterNumber
Integer value that identifies the parameter in the SQL statement identified by the *StatementID* argument.
Value
32-bit signed analog number to assign as the value of the specified parameter.

Example

This example sets the third parameter of the first statement to 2.1e9.

```
SQLSetParamLong(1, 3, 2.1e9);
```

See Also

SQLPrepareStatement()

SQLSetParamNull() function

The SQLSetParamNull() function sets a specified parameter within a SQL statement to NULL.

Category

SQL

Syntax

```
SQLSetParamNull(StatementID, ParameterNumber, ParameterType, Precision, Scale)
```

Arguments

StatementID

Integer value that identifies a SQL statement within a query.

ParameterNumber

Integer value that identifies the parameter in the SQL statement identified by the *StatementID* argument.

ParameterType

Integer value that specifies the type of data associated with the parameter specified by the *ParameterNumber* argument. The *ParameterType* argument can be assigned the following values:
0: String
1: Date/time
2: Integer
3: Floating point number
4: Decimal number

Precision

Precision of the data associated with the parameter data type.

Scale

Decimal value's scale. This value is required only if applicable to the parameter being set to null.

Remarks

Comparison with the NULL value is controlled by the ANSI_NULLS option in SQL Server. In SQL Server 7.0, this option is resolved at object creation time, not at query execution time. When a stored procedure is created in SQL Server 7.0, this option is ON by default and thus a clause such as "WHERE MyField = NULL" always returns NULL (FALSE) and no row is returned from a SELECT statement using this clause.

In order for the comparison = or <> to return TRUE or FALSE, it is necessary to set the option to OFF when creating the stored procedure. If the ANSI_NULLS is not set to OFF, then SQLSetParamNull() does not work as expected. In this case, comparison against NULL value should use the syntax "WHERE MyField IS NULL" or "WHERE MyField IS NOT NULL".

Example

This transaction set returns all rows of the Products table where the ProductName is not NULL.

```
SET ANSI_NULLS OFF
GO
CREATE PROCEDURE sp_TestNotNull @ProductParam varchar(255)
AS SELECT * FROM Products WHERE ProductName <> @ProductParam
```

```
GO
SET ANSI_NULLS ON
GO
```

InTouch can run the following SQL Access scripts.

```
ResultCode = SQLSetStatement(ConnectionId, "sp_TestNotNull");
ResultCode = SQLPrepareStatement(ConnectionId, StatementID);
ResultCode = SQLSetParamNull(StatementID, 1, 0, 0, 0);
ResultCode = SQLExecute(ConnectionId, BindList, StatementID);
ResultCode = SQLFirst(ConnectionId);
ResultCode = SQLClearStatement(ConnectionId, StatementID);
```

See Also

SQLPrepareStatement()

SQLSetParamTime() function

The SQLSetParamTime() function sets the value of the specified time parameter to a specified string.

Category

SQL

Syntax

```
SQLSetParamTime(StatementID, ParameterNumber, Value)
```

Arguments

StatementID

Integer value that identifies a SQL statement within a query.

ParameterNumber

Actual parameter number in the SQL statement identified by the *StatementID* argument.

Value

Actual value to set. Set the parameter specified by the ParameterNumber argument to a time value. The current date from the computer running the function is included with the specified time.

Example

This example sets the second parameter from the fourth SQL statement to 10:00 AM.

```
ResultCode=SQLSetParamTime(1, 3, "10:00:00 AM");
```

See Also

SQLPrepareStatement()

Clear statement parameters

The **SQLClearParam()** function clears the value of the specified parameter.

SQLClearParam() function

The **SQLClearParam()** function clears the value of the specified parameter. One of the **SQLSetParam***xxx***()** functions must be called again to reload parameters before calling the **SQLExecute()** function to run the query.

Category

SQL

Syntax

```
[ResultCode=]SQLClearParam(StatementID,ParameterNumber);
```

Arguments

StatementID

Integer value returned when a SQLPrepareStatement() function runs.

ParameterNumber

The ParameterNumber argument identifies the actual argument within the SQL statement to modify. Set the value of ParameterNumber associated with *StatementID* to zero or a zero-length string, depending on whether the argument is numeric or a string.

See Also

SQLPrepareStatement(), SQLExecute()

Execute a statement

The SQLExecute() function can be used within an InTouch script to run a SQL query during run time.

SQLExecute() function

The SQLExecute function runs a SQL query within a script. If the statement includes a SELECT, the BindList argument designates the name of the Bind List to use for binding the database columns with InTouch tags. If the

Bind List is NULL, no tag associations are made.

Category

SQL

Syntax

```
SQLExecute(ConnectionID,BindList,StatementID);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

BindList

The BindList argument can be a zero-length string. If *StatementID* is associated with a row-returning query, then the logical table is updated with the result of **SQLExecute()**. If a real Bind List is specified, then the result is associated with the BindList argument. A zero-length Bind List is useful when it is known in advance that the StatementID is not associated with a row-returning query.

StatementID

Integer value returned by SQL when a SQLPrepareStatement() function is used.

Remarks

Errors are returned in the function return. If the statement has been prepared, the statement handle returned from the prepare should be passed. If the statement has not been prepared, the statement handle should be zero.

Note: The SQLExecute() function can be called only once for a statement that has not been prepared. If the statement has been prepared, it can be called multiple times.

A default statement is associated with a connection ID. It can be a textual SQL statement (SELECT, INSERT, DELETE, or UPDATE), the name of a query in MS Access (with or without parameters), or the name of a stored procedure in MS SQL Server (with or without parameters).

The default statement is modified by the **SQLLoadStatement()**, **SQLSetStatement(),** and **SQLAppendStatement()** functions. The default statement is used by **SQLExecute()** whenever StatementID = 0 is specified.

Examples

This example loads the SQL statements from the lotquery.sql file and places the results of the SELECT statement to InTouch tags specified by the Bind List.

```
ResultCode = SQLLoadStatement (ConnectionID, "c:\myappdir\lotquery.sql");
ResultCode = SQLExecute (ConnectionID, "BindList", 0);
ResultCode = SQLNext (ConnectionID);
```

This **SQLSetStatement()** function must be used for complex queries and string expressions greater than 131 characters. When the string expression exceeds 131 characters use the **SQLAppend()** function.

```
SQLSetStatement(ConnectionID, "Select Speed, Ser_No from tablename where Ser_No =’" +
Serial_input + "’");
SQLExecute(ConnectionID, "BindList", 0);
```

In the previous example, the StatementID argument is set to zero so the statement does not have to call SQLPrepareStatement(Connection_Id, StatementID) before the execute statement.

Because the StatementID is not created by the SQLPrepare statement to properly end this SELECT, use the SQLEnd() function instead of the **SQLClearStatement() function.**

```
SQLSetStatement(Connection_Id, "Select Speed, Ser_No from tablename where Ser_No =’" +
Serial_input + "’");
SQLPrepareStatement(Connection_Id, StatementID);
SQLExecute(Connection_Id, StatementID);
```

In the above example, the StatementID is created by a SQLPrepareStatement function call and used by the SQLExecute function. To end this SELECT statement, use a SQLClearStatement() function call within a script to free resources and the StatementID.

The SQLExecute() function supports some stored procedures. For example, suppose you create a stored procedure on the database server named "LotInfoProc," that contains the following select statement: "Select LotNo, LotName from LotInfo".

You write the InTouch QuickScript to run the stored procedure based upon the type of database that you are using. The following example shows script statements to run a stored procedure for a SQL Server database.

```
ResultCode = SQLSetStatement (ConnectionID,"LotInfoProc");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext (ConnectionID);
{Get results of Select}
```

The following example shows script statements to run a stored procedure for an Oracle database.

```
ResultCode = SQLSetStatement (ConnectionID, "{CALL LotInfoProc}");
ResultCode = SQLExecute(ConnectionID, "BindList", 0);
ResultCode = SQLNext (ConnectionID);
{Get results of Select}
```

See Also

SQLConnect(), SQLPrepareStatement()

Releasing occupied resources

The **SQLClearStatement** function releases database resources associated with the statement specified by the *StatementID*.

SQLClearStatement() function

The **SQLClearStatement()** function releases database resources associated with the statement specified by the *StatementID* argument.

Category

SQL

Syntax

```
[ResultCode=]SQLClearStatement(ConnectionID, StatementID);
```

Arguments

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

StatementID

Integer value returned by SQL when a **SQLPrepareStatement()** function is used.

See Also

SQLConnect(), SQLPrepareStatement()

Work with transaction sets

SQL Access includes a set of transaction functions to change insert, update, or delete records from a database. Generally, these transactions are grouped within a script in the form of a transaction set. A transaction set is committed at one time.

SQLTransact() function

The **SQLTransact()** function defines the beginning of a group of SQL statements called a transaction set. A transaction set is handled like a single transaction. After the **SQLTransact()** function runs, all subsequent operations are not committed to the database until the **SQLCommit()** function runs successfully.

Category

SQL

Syntax

```
[ResultCode=]SQLTransact(ConnectionID)
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Example

This example transaction set includes three insert statements.

```
ResultCode = SQLTransact(ConnectionID);
ResultCode = SQLInsertPrepare(ConnectionID, TableName, BindList, StatementID);
ResultCode = SQLInsertExecute(ConnectionID, BindList, StatementID);
ResultCode = SQLInsertExecute(ConnectionID, BindList, StatementID);
ResultCode = SQLInsertExecute(ConnectionID, BindList, StatementID);
ResultCode = SQLInsertEnd(ConnectionID, StatementID);
ResultCode = SQLCommit(ConnectionID);
```

See Also

SQLCommit(), SQLRollback()

SQLCommit() function

The **SQLCommit()** function defines the end of a transaction set. After the **SQLTransact(**) function runs, all subsequent all SQL statements within the transaction set are not committed to the database until the **SQLCommit()** function runs successfully.

Note: Use caution when writing QuickScripts that include the SQLCommit() function. Processing time increases with the number of SQL statements in a transaction set.

Category

SQL

Syntax

```
[ResultCode=]SQLCommit(ConnectionID)
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Example

This example script includes a transaction set that makes three database inserts.

```
ResultCode = SQLTransact(ConnectionID);
ResultCode = SQLInsertPrepare(ConnectionID, TableName, BindList, StatementID);
ResultCode = SQLInsertExecute(ConnectionID, BindList, StatementID);
ResultCode = SQLInsertExecute(ConnectionID, BindList, StatementID);
ResultCode = SQLInsertExecute(ConnectionID, BindList, StatementID);
ResultCode = SQLInsertEnd(ConnectionID, StatementID);
```

```
ResultCode = SQLCommit(ConnectionID);
```

See Also

SQLRollback(), SQLTransact(), SQLCommit()

SQLRollback() function

The SQLRollback() function reverses or rolls back the most recent transaction set. A transaction set is a group of commands issued between the **SQLTransact()** and the **SQLCommit()** functions.

A transaction set is handled like a single transaction. After the SQLTransact() function runs, all subsequent operations are not committed to the database. Query changes to the database occur after the SQLCommit() function runs. The SQLRollback() function rolls back the transaction set if it runs before the SQLCommit() function.

Category

SQL

Syntax

```
[ResultCode=]SQLRollback(ConnectionID)
```

Argument

ConnectionID

Name of a memory integer tag that holds the number (ID) assigned by the SQLConnect() function to each database connection.

Example

This example rolls back the database values prior to the **SQLTransact** function within the script.

```
ResultCode =SQLTransact(ConnectionID);
ResultCode = SQLInsertPrepare(ConnectionID, TableName, BindList, StatementID);
ResultCode = SQLInsertExecute(ConnectionID, BindList, StatementID);
ResultCode = SQLInsertEnd(ConnectionID, StatementID);
ResultCode =SQLRollback(ConnectionID);
```

See Also

SQLCommit(), SQLTransact()

Open the ODBC Administrator dialog box at runtime

Use the **SQLManageDSN()** function to run the Microsoft ODBC Manager while an InTouch application is running.

SQLManageDSN() function

The **SQLManageDSN** function runs the Microsoft ODBC Manager setup program while an InTouch application is running. **SQLManageDSN()** can be used within a script to add, delete, and modify existing data source names of a SQL Server or Access database.

Category

SQL

Syntax

```
SQLManageDSN(ConnectionId)
```

Argument

ConnectionId

ConnectionId is not used. It is retained for backward compatibility with older versions of SQL Access. Therefore, any number can be passed to the function. No database connection needs to be established before running the function to open Microsoft ODBC Manager.

Example

```
SQLManageDSN(0);
```

Understand SQL error messages

This section explains how to troubleshoot InTouch applications that use SQL Access functions. The first section describes the SQLErrorMsg() function and includes a table of SQL result codes with their corresponding error messages. The second section includes tables with specific database error messages.

SQLErrorMsg() function

All SQL functions return a result code that can be used for troubleshooting. The SQLErrorMsg() function returns the error message associated with the result code and assigns it as the value of an InTouch message tag.

Category

SQL

Syntax

```
ErrorMsg = SQLErrorMsg(ResultCode);
```

Argument

Result Code

Integer value returned by a previous SQL function. The SQLErrorMsg() function sets the value of an InTouch message tag to the message associated with the result code. For more information about error messages associated with result codes, see Understanding SQL Error Messages.

Remarks

Refer to your database documentation for undocumented result codes. Also, browse the Log Viewer for any additional error messages.

The SQLTrace=1 flag defined under the [InTouch] section of the win.ini file is useful for debugging SQL Access scripts.

Example

This example assigns the error message associated with the SQL Access Manager result code to the ErrorMsg message tag.

```
ErrorMsg = SQLErrorMsg(ResultCode)
```

See Also

SQLConnect()

SQL Access Manager result codes and messages

The following table lists some common SQL Access result codes, their corresponding error messages, and descriptions:

Result Code	Error Message	Description
0	No errors occurred	The SQL function ran successfully without errors.
-1	<Message from the database provider>	A specific error message from the vendor database.
-2	An empty statement cannot be executed	A SQLExecute(ConnectionId, BindList, 0) is run without previously calling SQLSetStatement or SQLLoadStatement with a non-empty statement.

Result Code	Error Message	Description
-4	Value returned was Null	An integer or real value read from the database is null. This is only a warning message sent to the Log Viewer.
-5	No more rows to fetch	The last record in the table has been reached.
-7	Invalid parameter ID	The SQLSetParamChar(), SQLSetParamDate(), SQLSetParamDateTime(), SQLSetParamDecimal(), SQLSetParamFloat(), SQLSetParamInt(), SQLSetParamLong(), SQLSetParamNull(), or SQLSetParamTime() function is called with an invalid parameter ID.
-8	Invalid parameter list	Example of an invalid parameter list: 1, 2, 3, 5 (Missing 4).
-9	Invalid type for NULL parameter	The SQLSetParamNull function is called with an invalid *Type* argument value.
-10	Changing data type of parameter is not allowed	The SQLSetParamChar(), SQLSetParamDate(), SQLSetParamDateTime(), SQLSetParamDecimal(), SQLSetParamFloat(), SQLSetParamInt(), SQLSetParamLong(), SQLSetParamNull(), or SQLSetParamTime() function is called with a different type for an existing parameter.
-11	Adding parameter after the statement has been executed successfully is not allowed.	The SQLSetParamChar(), SQLSetParamDate(), SQLSetParamDateTime(), SQLSetParamDecimal(), SQLSetParamFloat(), SQLSetParamInt(), SQLSetParamLong(), SQLSetParamNull(), or SQLSetParamTime() function is

Result Code	Error Message	Description
		called for a new parameter ID after the statement has been run successfully.
-12	Invalid date time format	An invalid date time format is encountered, for example, when executing SQLSetParamTime(), SQLInsertExecute(), or SQLUpdateCurrent().
-13	Invalid decimal format	An invalid decimal format is encountered, for example, when executing SQLSetParamDecimal(), SQLInsertExecute(), or SQLUpdateCurrent().
-14	Invalid currency format	An invalid currency format is encountered, for example, when executing SQLInsertExecute() or SQLUpdateCurrent().
-15	Invalid statement type for this operation	SQLInsertEnd is called for a statement ID created by SQLPrepareStatement() or SQLClearStatement() is called for a statement ID created by SQLInsertPrepare().
-1001	Out of memory	There is insufficient memory to run this function.
-1002	Invalid connection	The value passed to the *ConnectionId* argument is not valid.
-1003	No Bind List found	The specified Bind List name does not exist.
-1004	No template found	The specified Table Template name does not exist.
-1005	Internal Error	An internal error occurred. Call Technical Support.

Result Code	Error Message	Description
-1006	String is null	Warning - the string read from the database is null. This is only a warning message sent to the Log Viewer.
-1007	String is truncated	Warning - the string read from the database is longer than 131 characters and is truncated when selected. The warning is sent to the Log Viewer.
-1008	No Where clause	There is no Where clause on Delete.
-1009	Connection failed	Check the Log Viewer for more information about the failed connection to the database.
-1010	The database specified on the DB= portion of the connect string does not exist	The specified database does not exist.
-1011	No rows were selected	A SQLNumRows(), SQLFirst(), SQLNext(), SQLLast(), or SQLPrev() function is called without running a SQLSelect() or SQLExecute() function first.
-1013	Unable to find file to load	The SQLLoadStatement() function is called with a file name that cannot be found.

Error messages from a vendor database return a ResultCode of -1. The SQL Access function ResultCode is always -1, but the message is copied exactly from the database provider.

For error messages that occur when using an Oracle database, refer to Oracle Server documentation for specific error messages and solutions.

The following table lists common error messages that can occur when using a Microsoft SQL Server or Access database.

Error Message	Solution
You cannot have more than one statement active at a time	You are trying to run a SQL command after calling the SQLSelect() function. Run SQLEnd() to free system resources from the SQLSelect() or use a separate *ConnectionId* for the second statement.
There is not enough memory available to process the command	Try rebooting the client workstation.
Invalid object name table name	The table name does not exist in the database you are using. Try DB=database name.

Check your Microsoft SQL Server documentation for specific error messages and solutions.

Reserved word list

This section lists keywords that are excluded from use with the SQL Access Bind List, the Table Template, and the ODBC interface.

If a reserved keyword is used as the Column Name in a Bind List or Table Template, an error message appears in the Log Viewer. The type of error depends upon the ODBC driver being used and the location where the keyword is found. For example, one of the most common errors is using DATE and TIME for Column Names in a Bind List or Table Template. To avoid this error, use a slightly different name, for example, "aDATE" and "aTIME".

Reserved keywords define the Structured Query Language (SQL) used by InTouch SQL Access. The keywords are also recognized by the specific ODBC driver being used. If the SQL command cannot be interpreted correctly, SQL Access Manager generates an error message that can be viewed from the Log Viewer.

The following alphabetical list shows the reserved keywords for SQL Access and ODBC:

ABSOLUTE	ADA	ADD
ALL	ALLOCATE	ALTER
AND	ANY	ARE
AS	ASC	ASSERTION
AT	AUTHORIZATION	AVG
BEGIN	BETWEEN	BIT
BIT_LENGTH	BY	CASCADE
CASCADED	CASE	CAST
CATALOG	CHAR	CHAR_LENGTH
CHARACTER	CHARACTER_LENGTH	CHECK

CLOSE COALESCE	COBOL	COLLATE
COLLATION	COLUMN	COMMIT
CONNECT	CONNECTION	CONSTRAINT
CONSTRAINTS	CONTINUE	CONVERT
CORRESPONDING	COUNT	CREATE
CURRENT	CURRENT_DATE	CURRENT_TIME
CURRENT_TIMESTAMP	CURSOR	DATE
DAY	DEALLOCATE	DEC
DECIMAL	DECLARE	DEFERRABLE
DEFERRED	DELETE	DESC
DESCRIBE	DESCRIPTOR	DIAGNOSTICS
DICTIONARY	DISCONNECT	DISPLACEMENT
DISTINCT	DOMAIN	DOUBLE
DROP	ELSE	END
ESCAPE	EXCEPT	EXCEPTION
EXEC	EXECUTE	EXISTS
EXTERNAL	EXTRACT	FALSE
FETCH	FIRST	FLOAT
FOR FOREIGN	FORTRAN	FOUND
FROM FULL	GET	GLOBAL
GO	GOTO	GRANT
GROUP	HAVING	HOUR
IDENTITY	IGNORE	IMMEDIATE
IN	INCLUDE	INDEX
INDICATOR	INITIALLY	INNER
INPUT	INSENSITIVE	INSERT
INTEGER	INTERSECT	INTERVALL

INTO	IS	ISOLATION
JOIN	KEY	LANGUAGE
LAST	LEFT	LEVEL
LIKE	LOCAL	LOWER
MATCH	MAX	MIN
MINUTE	MODULE	MONTH
MUMPS	NAMES	NATIONAL
NCHAR	NEXT	NONE
NOT	NULL	NULLIF
NUMERIC	OCTET_LENGTH	OF
OFF	ON	ONLY
OPEN	OPRN	OPTION
OR	ORDER	OUTER
OUTPUT	OVERLAPS	PARTIAL
PASCAL	PLI	POSITION
PRECISION	PREPARE	PRESERVE
PRIMARY	PRIOR	PRIVILEGES
PROCEDURE	PUBLIC	RESTRICT
REVOKE	RIGHT	ROLLBACK
ROWS	SCHEMA	SCROLL
SECOND	SECTION	SELECT
SEQUENCE	SET	SIZE
SMALLINT	SOME	SQL
SQLCA	SQLCODE	SQLERROR
SQLSTATE	SQLWARNING	SUBSTRING
SUM	SYSTEM	TABLE
TEMPORARY	THEN	TIME

TIMESTAMP	TIMEZONE_HOUR	TIMEZONE_MINU
TO	TRANSACTION	TRANSLATE
TRANSLATION	TRUE	UNION
UNIQUE	UNKNOWN	UPDATE
UPPER	USAGE	USING
VALUE	VALUES	VARCHAR
VARING	VIEW	WHEN
WHENEVER	WHERE	WITH
WORK	YEAR	

Use the 16-Pen trend wizard

You can use an InTouch wizard to create real-time and historical trends capable of displaying data from up to 16 tags. The 16-Pen Trend is a supplementary component that you can install during an InTouch installation.

The 16-Pen Trend Wizard can be configured much like other InTouch chart wizards. The 16-Pen Trend wizard allows you to configure the following trend properties:

- Tag assigned to each trend pen
- Trend line width and color
- Starting and ending dates and times for historical trends
- Update rate and time span for real-time trends
- Minimum and maximum engineering units assigned to a trend tag
- Major and minor trend time divisions
- Major and minor trend value divisions

Create a 16-Pen Trend

You can create a trend by selecting the 16-Pen Trend Wizard from WindowMaker.

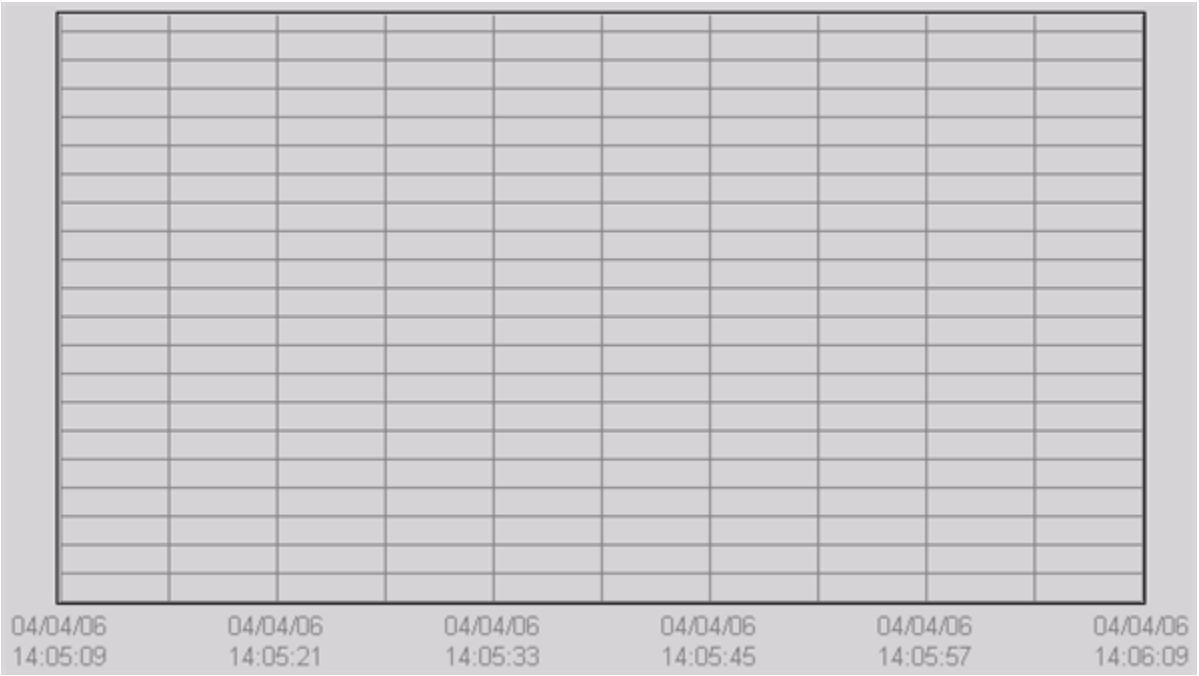
Create a 16-Pen real-time or historical trend

1. Open a window from WindowMaker to place the 16-Pen Trend.
2. On the **Draw** menu, in the **Insert** group, select **Wizards**.
   The **Wizard Selection** dialog box appears.
3. Select **Trends** from the list of wizards.
   The right pane of the **Wizard Selection** dialog box shows a set of trend wizard icons.
4. Select the **16-Pen Trend** wizard and select **OK**.

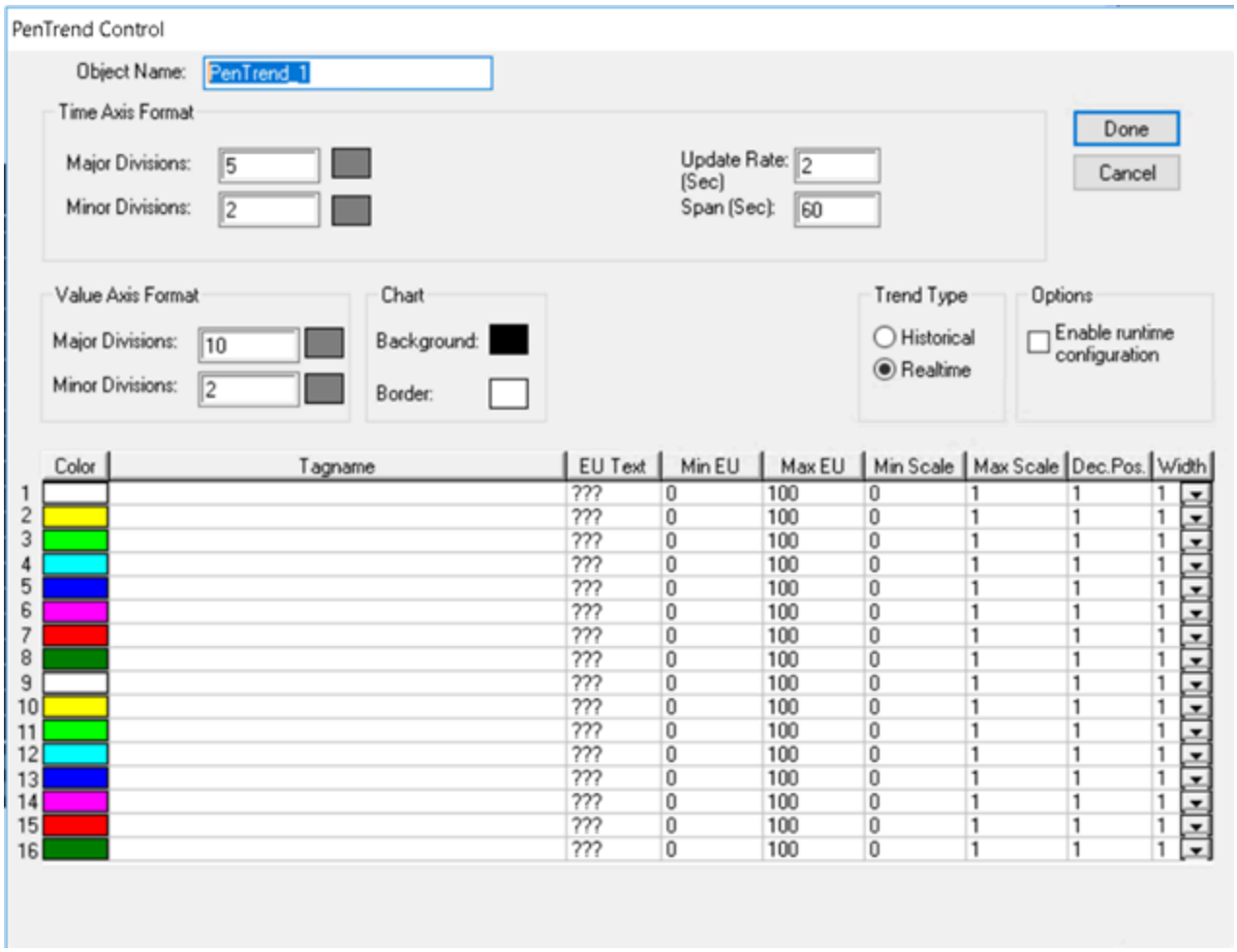
The **Wizard Selection** dialog box closes and your window reappears.

5. Select in the window to place the 16-Pen Trend.

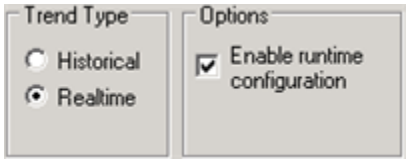
   The wizard places a 16-Pen Trend template in the window.



6. Double-click the 16-Pen Trend template to open the **PenTrend Control** dialog box.



7. In the **Trend Type** area, select **Historical** or **Realtime** as the type of trend you want to create.



   The **PenTrend Control** dialog box automatically shows the appropriate time and update options based upon the type of trend you select.

8. In the **Options** area, select or clear the **Enable runtime configuration** option.

   Selecting this option allows operators to modify some properties of the 16-Pen Trend while it is running.

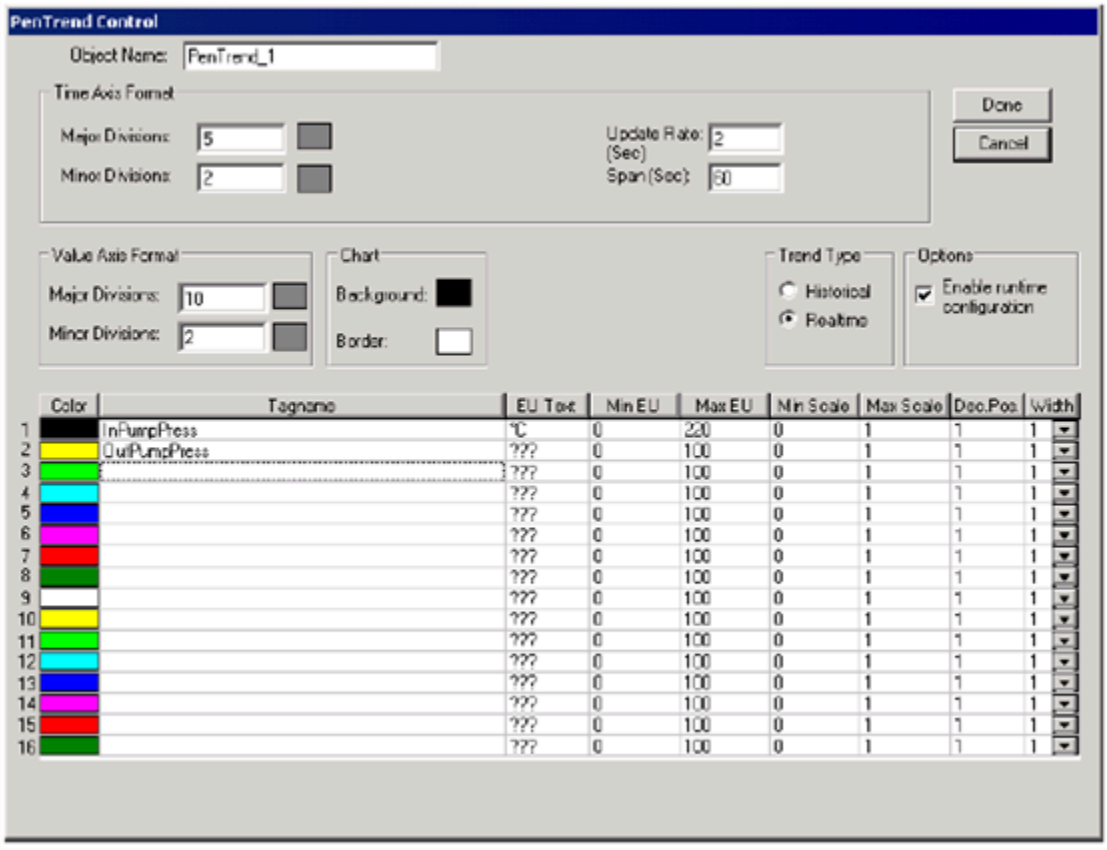
Configure which tags to display on the Trend graph

You can use the 16-Pen Trend Wizard to assign tags to trend pens. The 16-Pen Trend Wizard includes a set of columns that specify tag properties shown on the trend. These columns use the default property values assigned to the tag from the Tagname Dictionary. You can override these assigned tag values by specifying other values when you configure the trend.

Configure 16-Pen Trend tags

1. Open the window containing the 16-Pen Trend template.

2. Double-click the 16-Pen Trend. The **PenTrend Control** dialog box appears with a grid area near the bottom to specify the tags associated with trend pens.



3. In the **Object Name** box, assign a name to the 16-Pen Trend.

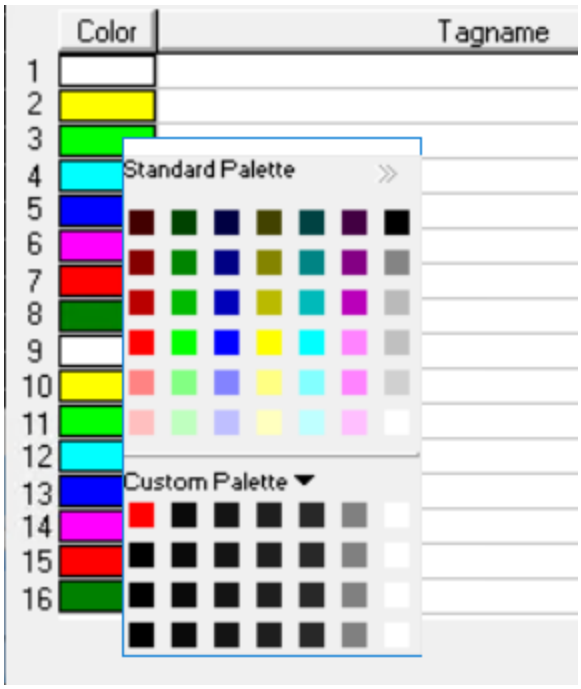
   The default name is PenTrend_1, which increments the number in the name as you create each new trend.

4. In the **Tagname** box, enter the name of the tag to associate with the pen number listed at the left of the grid.

   Double-clicking within a cell beneath **Tagname** shows the **Select Tag** dialog box. You can assign a tag to a pen by selecting the tag from the **Select Tag** dialog box.

   Note: You remove a tag by selecting a Tagname box containing a tag name and pressing your keyboard space bar.

5. In the **Color** column, click each color box to open a color palette. Select a color for the pen.



6. In the **EU Text** column, enter the text that you want to initially use in run time as the header text for the pen axis for each respective pen.

   This text is the axis text when a pen is set to active. The **EU Text** column is initially assigned the tag's engineering units from the Tagname Dictionary. You can override the these default engineering units for the 16-Pen Trend.

7. In the **Min EU** column, enter the minimum engineering units value assigned to the pen.

   The **Min EU** column initially shows the tag's minimum engineering units value from the Tagname Dictionary. You can assign another minimum engineering units value that applies only to a 16-Pen Trend.

8. In the **Max EU** column, enter the maximum engineering units assigned to the pen.

   The **Max EU** column initially shows the tag's maximum engineering units value from the Tagname Dictionary. You can assign another maximum engineering units value that applies only to a 16-Pen Trend.

   Note: The Min/Max engineering units are very important for showing historical trend data. The historical trend shows from 0-100 percent of engineering units scale.

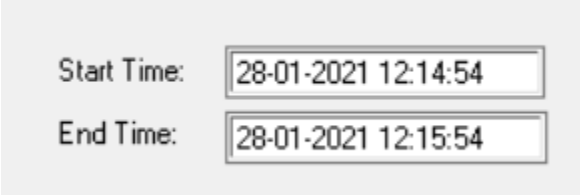
9. In the **Min Scale** column, enter the percentage that you want to use initially in run time to calculate the minimum pen axis grid for the respective EU scale.
10. In the **Max Scale** column, enter the percentage that you want to use initially in run time to calculate the maximum pen axis grid for the respective EU scale.
11. In the **Dec.Pos** column, enter the number of decimal points that you want to use initially in run time when labeling the pen axis grid.
12. In the **Width** column, select the pen line width in pixels to plot data values shown on the trend.
13. Continue with the next procedure to update the trend time and update rate of a 16-Pen Trend.

Configure the Trend time span and update rate

The **Pen Trend Control** dialog box shows different options based upon whether you are creating a real-time or historical 16-Pen Trend. You can set the time span for a historical trend and the update rate for a real-time trend.

Configure the time span of a 16-Pen historical trend

1. Double-click a 16-Pen historical trend within a window. The **PenTrend Control** dialog box appears with options to set the starting and ending dates and time of a trend.



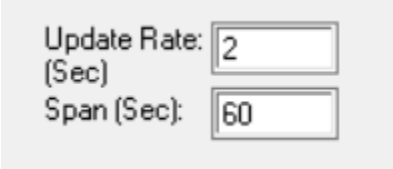
2. Set the starting and ending date and time of the historical trend.

   Use the following format for both the starting and ending dates and times:

   DD-MM-YYYY HH:MM:SS AM/PM

Configure the update rate of a 16-Pen real-time trend

1. Double-click a 16-Pen real-time trend object within a window. The **PenTrend Control** dialog box appears with options to set update rate and span of a real-time trend.



2. In the **Update Rate** box, type the number of seconds between each refresh interval of the historical trend.
3. In the **Span** box, type the number of seconds of the real-time interval shown in the trend.

Configure the Trend display options

You can configure the visual appearance of a trend with the 16-Pen Trend Wizard.

Configure the display options of a 16-Pen Trend

1. Double-click the **16-Pen Trend** in WindowMaker.

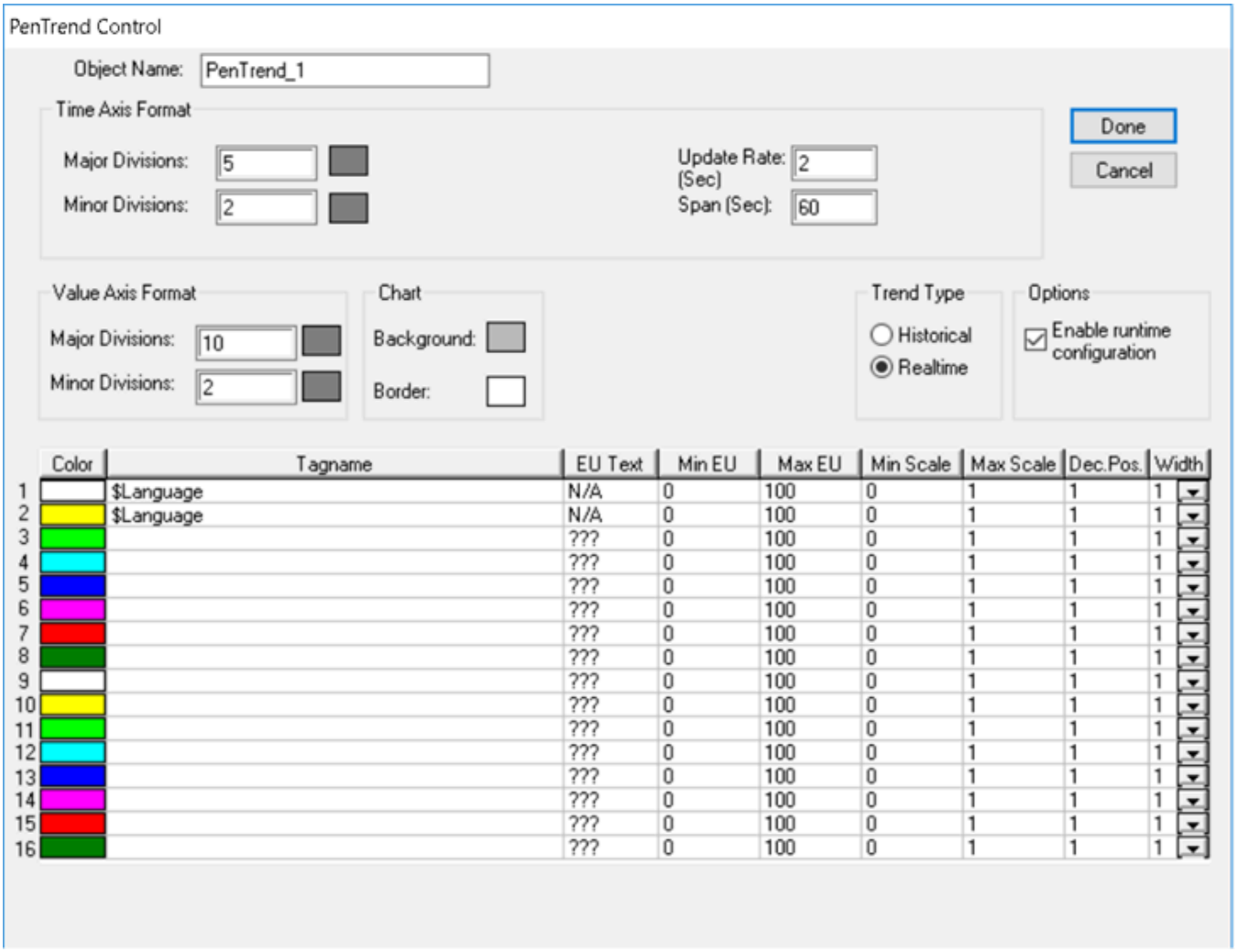
   The **PenTrend Control** dialog box appears.
2. In the **Time Axis Format** area, enter the number of major time divisions in **Major Divisions**. This option sets the number of major time divisions on the horizontal axis of the trend.
3. Select the color box to the right of **Major Divisions** to open the color palette and select a color if you want to assign another color to the major time division lines. Otherwise, skip this step and accept the default color assigned to major time division lines.
4. In the **Minor Divisions** box, enter the number of minor time divisions shown on the horizontal axis of a trend.
5. Select a color for the minor time division lines.
6. In the **Value Axis Format** area, enter the number of major divisions in **Major Divisions**. This option sets the number of major divisions shown on the vertical value axis.
7. Set the color of the major value divisions.

8. In the **Minor Divisions** box, enter the number of minor value divisions shown on the vertical axis of the trend.
9. Set the color of the minor value divisions.
10. In the **Chart** area, select the background and border colors of the trend.
11. Select **Done** to save the configuration changes to the 16-Pen Trend.

Change the Trend configuration at runtime

If the **Enable runtime configuration** option is selected from the **PenTrend Control** dialog box, operators can change some characteristics of a 16-Pen Trend while the application is running.

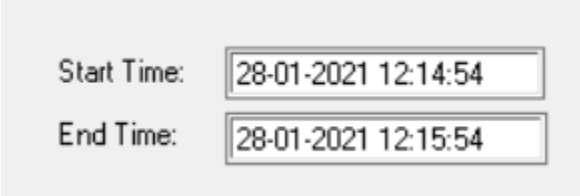
Run-time changes to the 16-Pen Trend are not permanent. If operators close WindowViewer and then start the application window again, the 16-Pen Trend retains the configuration originally defined from WindowMaker. The following figure shows the **PenTrend Control** dialog box that appears if you select on a 16-Pen Trend while it is running.



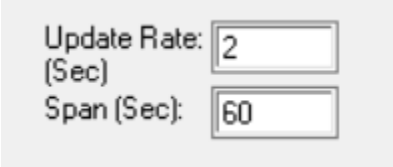
You can change the following during run time:

- Tags or expressions assigned to trend pens
- Characteristics of trend tags or expressions

- Type of trend (Historical or Realtime)
- Date and time range of a historical trend



- Update rate and frequency of a real-time trend.



After clicking **Done**, the trend retains the configuration changes for the duration of the current WindowViewer application session.

Control a 16-Pen Trend Wizard using scripts

You can use a set of functions in QuickScripts to control a 16-Pen Trend object in run time. For example, you can connect pens to the chart, add new events to the chart, remove or replot the grid, and remove or replot the scooters.

ptGetTrendType() function

The ptGetTrendType() function can be used in a script to return a value that indicates whether the current mode of a 16-Pen Trend shows historical or real-time data.

Category

Pen trend

Syntax

```
ptGetTrendType(TrendName);
```

Argument

TrendName

Name of the trend. *TrendName* must be either a string constant or message tag.

Return Value

Returns the trend type:

0 = Historical trend

1 = Real-time noscroll

2 = Real-time trend

Example

The following example returns a value that indicates whether the PumpPress trend shows historical or real-time data.

```
ptGetTrendType("PumpPress");
```

ptLoadTrendCfg() function

The ptLoadTrendCfg() function can be used in a script to load trend configuration values from a file.

Category

Pen trend

Syntax

```
ptGetTrendCfg(TrendName,FileName);
```

Arguments

TrendName

The name of the trend object. The value assigned to *TrendName* must be either a string constant or a message tag.

FileName

Name of the configuration file. The folder path to the configuration file must be included with the *FileName* argument.

Example

The TankFarm trend is configured with values from the C:\TrendCfg.txt file.

```
ptLoadTrendCfg("TankFarm","C:\TrendCfg.txt");
```

ptPanCurrentPen() function

The ptPanCurrentPen() function can be used in a script to scroll a 16-Pen Trend's pen upward or downward on the vertical value axis. Vertical scrolling is determined by the number of major and minor trend units specified as argument values.

Category

Pen trend

Syntax

```
ptPanCurrentPen(TrendName,MajorUnits, MinorUnits);
```

Arguments

TrendName

The name of the trend object. *TrendName* must be either a string constant or message tag.

MajorUnits

Multiplier to scroll by the number of units defined by the major division lines. A negative number indicates a downward scroll of the vertical axis.

MinorUnits

Multiplier for additional scrolling by number of units defined by the minor division lines. A negative number indicates a downward scroll of the vertical axis.

Examples

This example scrolls the pen upward one major division line.

```
ptPanCurrentPen("TrendName", 1, 0);
```

This example scrolls the pen upward half a minor trend division.

```
ptPanCurrentPen("TrendName", 0, 0.5);
```

This example scrolls the pen downward by 2 major division lines and half of a minor division line.

```
ptPanCurrentPen("TrendName", -2, -0.5);
```

This example scrolls one major division line up 1 and downward by 2 minor division lines.

```
ptPanCurrentPen("TrendName", 1, -2);
```

ptPanTime() function

The ptPanTime() function can be used in a script to scroll a 16-Pen Trend's pen left or right on the horizontal time axis based on the number of specified major or minor trend units.

Category

Pen trend

Syntax

```
ptPanTime(TrendName, MajorUnits, MinorUnits);
```

Arguments

TrendName

The name of the trend object. *TrendName* must be either a string constant or message tag.

MajorUnits

Multiplier for scrolling by the number of horizontal major division lines. A negative number indicates panning left on the trend.

MinorUnits

Multiplier for additional scrolling by number of units defined by the minor division lines. A negative number indicates panning left on the trend.

Remarks

The settings for Major Division and Minor Division specified in the **PenTrend Control** dialog box during development are the basis from which the amount to scroll by is calculated. A trend with a time span of 120 seconds, a major division value of 10 and a minor division value of 2, results in a trend with a major division line every 12 seconds and a minor division line every 6 seconds. The function ptPanTime("TrendName",1,0.5) scrolls the time axis by 1*12 + 0.5*6 = 15 seconds.

Examples

This example scrolls the pen 1 major division to the right on the horizontal trend axis.

```
ptPanTime("TrendName", 1, 0);
```

This example scrolls the pen to the right on the horizontal axis of the trend by 0.5 minor division.

```
ptPanTime("TrendName", 0, 0.5);
```

This example scrolls the pen 2.5 major divisions to the left on the horizontal axis of the trend.

```
ptPanTime("TrendName", -2, -0.5);
```

This example scrolls the pen 1 major division to the right and 2 minor divisions to the left.

```
ptPanTime("TrendName", 1, -2);
```

ptPauseTrend() function

The ptPauseTrend() function can be used in a script to temporarily stop a 16-Pen Trend from updating the graph. The trend remains stopped until you call ptPauseTrend again with a value of 0.

Category

Pen trend

Syntax

```
ptPauseTrend(TrendName, Value);
```

Arguments

TrendName

The name of the trend object. *TrendName* must be either a string constant or message tag.

Value

A value of 1 pauses trend updates. A value of 0 resumes trend updating.

Example

This example pauses any further updates to the 16-Pen Trend while the *Value* argument is 1.

```
ptPauseTrend ("TrendName",1);
```

ptSaveTrendCfg() function

The ptSaveTrendCfg() function can be used in a script to save a trend's current configuration values to a file.

Category

Pen trend

Syntax

```
ptSaveTrendCfg(TrendName,FileName);
```

Arguments

TrendName

The name of the trend object. Must be either a string constant or message tag.

FileName

Name of the file to save the trend's configuration values. The folder path to the configuration file can be specified with the FileName argument.

Example

The **ptSaveTrendCfg()** function saves the values from the PumpTrend 16-Pen Trend to the C:\Config.txt file.

```
ptSaveTrendCfg ("PumpTrend", "C:\Config.txt")
```

ptSetCurrentPen() function

The ptSetCurrentPen() function can be used in a script to select a pen by its assigned number to control the pen axis.

Category

Pen trend

Syntax

```
ptSetCurrentPen(TrendName, PenNum);
```

Arguments

TrendName

Name of the trend. Must be either a string constant or message tag.

PenNum

Number of the pen (1-16) assigned as the current trend pen.

Example

The **ptSetCurrentPen()** function assigns pen 2 as the current pen of the PumpPress trend.

```
ptSetCurrentPen("PumpPress",2);
```

ptSetPen() function

The ptSetPen() function can be used in a script to assign a tag to a trend pen.

Category

Pen trend

Syntax

```
ptSetPen(TrendName,PenNum,TagName);
```

Arguments

TrendName

The name of the trend object. Must be a string constant or a message tag.

PenNum

Number of the pen assigned as the current trend pen.

TagName

Name of the tag assigned to the trend pen.

Example

The **ptSetPen()** function assigns the **PumpInP** tag to pen 2 of the PumpPress trend.

```
ptSetPen ("PumpPress",2,"PumpInP");
```

ptSetPenEx() function

The ptSetPenEx() function can be used in a script to assign a tag to a specific trend pen and override the tag's configuration values specified in the Tagname Dictionary.

Category

Pen trend

Syntax

```
ptSetPenEx(TrendName, PenNum, TagName, minEu, maxEU, minPercent, maxPercent, Decimal, EU);
```

Arguments

TrendName

The name of the trend object. Must be either a string constant or message tag.

PenNum

Number of the pen assigned as the current trend pen.

TagName

Name of the tag assigned to the trend pen.

minEU

The minimum engineering units value for the specified tag.

maxEU

The maximum engineering units value for the specified tag.

minPercent

The percentage to use initially in run time to calculate the minimum pen axis grid for the respective EU scale.

maxPercent

The percentage to use initially in run time to calculate the maximum pen axis grid for the respective EU scale.

Decimal

Decimal precision of a tag's value in the trend.

EU

The label for the tag's engineering units.

Example

The **ptSetPenEx()** function assigns the **PumpInP** tag to pen 2 of the PumpPress trend. The tag's engineering units

range is set between 0 to 1500 and its units are PSI. The percentages for the grid are 0 to 1, and the decimal precision is set to 2.

```
ptSetPenEx ("PumpPress", 2, "PumpInP", 0, 1500, 0, 1, 2, "PSI");
```

ptSetTimeAxis() function

The ptSetTimeAxis() function can be used in a script to set the trend's starting date and time and ending date and time.

Category

Pen trend

Syntax

```
ptSetTimeAxis(TrendName,StartDateTime,EndDateTime);
```

Arguments

TrendName

The name of the trend object. Must be either a string constant or message tag.

StartDateTime

The date and time when the trend begins. The format for the starting date and time is: dd/mm/yyyy hh:mm:ss AM/PM.

EndDateTime

The date and time when the trend ends. The format for the ending date and time is: dd/mm/yyyy hh:mm:ss AM/PM.

Example

The **ptSetTimeAxis()** function sets the starting and ending dates and times of a trend for a 25 hour period starting at 8:30 on May 22, 2007.

```
ptSetTimeAxis ("PumpPress", "05/22/2007 08:30:00 AM", "05/23/2007 09:30:00 AM");
```

ptSetTimeAxisToCurrent() function

The ptSetTimeAxisToCurrent() function can be used in a script to calculate the current chart span and the chart's ending time.

Category

Pen trend

Syntax

```
ptSetTimeAxisToCurrent(TrendName);
```

Argument

TrendName

The name of the trend object. *TrendName* must be a string constant or a message tag.

Example

The **ptSetTimeAxisToCurrent()** function sets the ending date and time of the PumpPress trend to the current date and time.

```
ptSetTimeAxisToCurrent("PumpPress");
```

ptSetTrend() function

The ptSetTrend() function can be used in a script to pause or restart updates to a 16-Pen Trend.

Category

Pen trend

Syntax

```
ptSetTrend(TrendName, EnableUpdates);
```

Arguments

TrendName

The name of the trend object. Must be either a string constant or message tag.

EnableUpdates

The value 1 starts updates to the trend. The value 0 stops trend updates.

Example

The **ptSetTrend()** function updates the PumpPress trend.

```
ptSetTrend("PumpPress",1);
```

ptSetTrendType() function

The ptSetTrendType() function can be used in a script to specify whether the trend shows historical or real-time data.

Category

Pen trend

Syntax

```
ptSetTrendType(TrendName, TrendType);
```

Arguments

TrendName

The name of the trend object. Must be either a string constant or Message tag.

TrendType

The value 1 indicates a historical trend. The value 2 specifies a real-time trend.

Example

The **ptSetTrendtype()** function specifies the PumpPress trend shows real-time data.

```
ptSetTrendType("PumpPress",2);
```

ptZoomCurrentPen() function

The ptZoomCurrentPen() function can be used in a script to change the value range shown on a trend's Y-axis. The range of the trend's vertical value axis can be increased or decreased by a specified zoom ratio.

Category

Pen trend

Syntax

```
ptZoomCurrentPen(TrendName,ZoomFactor);
```

Arguments

TrendName

The name of the trend object. Must be either a string constant or message tag.

ZoomFactor

Assigning a number larger than 1.0 increases the value range of the trend by multiplying the current range limits by the zoom factor. Assigning a zoom factor less than 1.0 decreases the value range shown in the vertical axis of the trend.

Remarks

The zoom ratio is applied to the existing span of the current pen's Y-axis range. For example, if the trend starts with a Y-axis range of –50 to 50 and then you zoom by a ratio of 2.0, the new range is –100 to 100. If you zoom by 2.0 again, then the new range is –200 to 200. The zoom ratio applies to the range currently in effect, not the original Y-axis range.

The zoom ratio persists during run time for each of the trend's pens. As you switch from one pen to another using the **ptSetCurrentPen()** function, the Y-axis value range reflects the current scaling for the selected pen.

Example

The **ptZoomCurrentPen** function doubles the Y-axis range of the current tag in the trend named "PumpPress".

```
ptZoomCurrentPen("PumpPress",2);
```

ptZoomTime() function

The ptZoomTime() function can be used in a script to change the time range shown on the trend's horizontal axis.

Category

Pen trend

Syntax

```
ptZoomTime(TrendName,Zoom);
```

Arguments

TrendName

The name of the trend object. Must be either a string constant or message tag.

Zoom

Assigning a number larger than 1.0 increases the time period shown on the trend's horizontal axis. Assigning a number less than 1.0 decreases the time period shown on the horizontal axis.

Examples

The **ptZoomTime()** function increases the time period shown on the trend's horizontal axis by 17 percent.

```
ptZoomTime("PenTrend_1",1.17);
```

The **ptZoomTime()** function decreases the time period shown on the trend's horizontal axis by 50 percent. For example, the **ptZoomTime()** function reduces the trend's time period to 30 minutes if the original time range was set to 1 hour.

```
ptZoomTime("PenTrend_1", 0.5);
```

Symbol Factory

Symbol Factory includes a collection of over 4,000 industrial symbols that can be used as visual elements in your InTouch application windows. Symbol Factory is a supplementary component that you can install during the InTouch HMI installation.

Note: Use the Industrial Graphic Editor to create visual elements for InTouch applications that interact with Application Server. You can also use the Graphic Editor to create intelligent visual elements for applications independent of the InTouch HMI. For more information about the Industrial Graphic Editor, see the Application Server documentation.

AVEVA provides no warranty of any kind for any of this product. You can report problems to Global Technical Support. We highly recommend that you always back up your application and data before you install or use any new utility or application.

Symbol types

Symbol Factory includes four types of wizards:

- Picture Wizards
- Bitmap Wizards
- Texture Wizards
- InTouch Objects

Picture wizards

Symbol Factory picture wizards are vector-based images of equipment or flow diagrams. As you are creating your application, you can modify picture wizard images by doing the following:

- Assign an animation to an image
- Flip an image horizontally or vertically
- Change the horizontal and vertical perspective of an image
- Rotate an image on its axis
- Change the fill color and pattern of an image
- Change the size, pattern, and color of image lines

Bitmap wizards

Bitmap wizards are bitmap images, such as windows icons, or a block of text. As you are creating your application, you can modify picture bitmap wizard images by doing the following:

- Assign an animation to a bitmap
- Flip an image horizontally or vertically
- Change the horizontal and vertical length of a bitmap
- Place a border or a shadow around the bitmap border
- Rotate the bitmap image on its axis in 90 degree increments

- Define a transparent color
- Replace up to three colors in the bitmap with other colors

Texture wizards

A texture wizard is similar to a bitmap wizard, except that it can be resized to form a continuous pattern. Texture wizards are typically used to create backgrounds for windows or as a fill for a graphic object. You select texture wizards from the Symbol Factory Textures category.

InTouch object

An InTouch object is an InTouch cell or wizard that is stored "as is" in the Symbol Factory.

After you paste an InTouch object from the Symbol Factory into WindowMaker, you cannot edit it in Symbol Factory.

When you double-select the object in WindowMaker, the **Substitute Tagnames** dialog box appears if the object is a cell, or the animation link selection dialog box appears for an individual graphic object.

Use Symbol Factory

Using the Symbol Factory wizard is very similar to using other wizards. Select a wizard, place it in a window, and configure it.

Get started quickly

If you are familiar with Symbol Factory, review these tips for getting started quickly:

- To configure options for a wizard, double-click it in the window and then select **Options** in the **Symbol Factory by Reichard Software** dialog box.
- To copy a wizard to another category, drag its thumbnail image into the **Categories** window and drop it on another category. To move, hold down the SHIFT key.
- To edit a wizard's description, right-click the wizard thumbnail. To edit a category's description, right-click the category.
- To delete a wizard, right-click the wizard thumbnail with and then select **Delete Symbol**.
- The Symbol Factory can be configured so that a group of developers can use and contribute to the same library of wizards across a network.
- All of the wizards in a particular category are stored in a file with the .cat file name extension. Symbol Factory category files are normally placed in the c:\program files\wonderware\intouch\symfac folder. You can copy the file into the c:\program files\wonderware\intouch\symfac folder on another computer.

Place a Symbol Factory wizard in a window

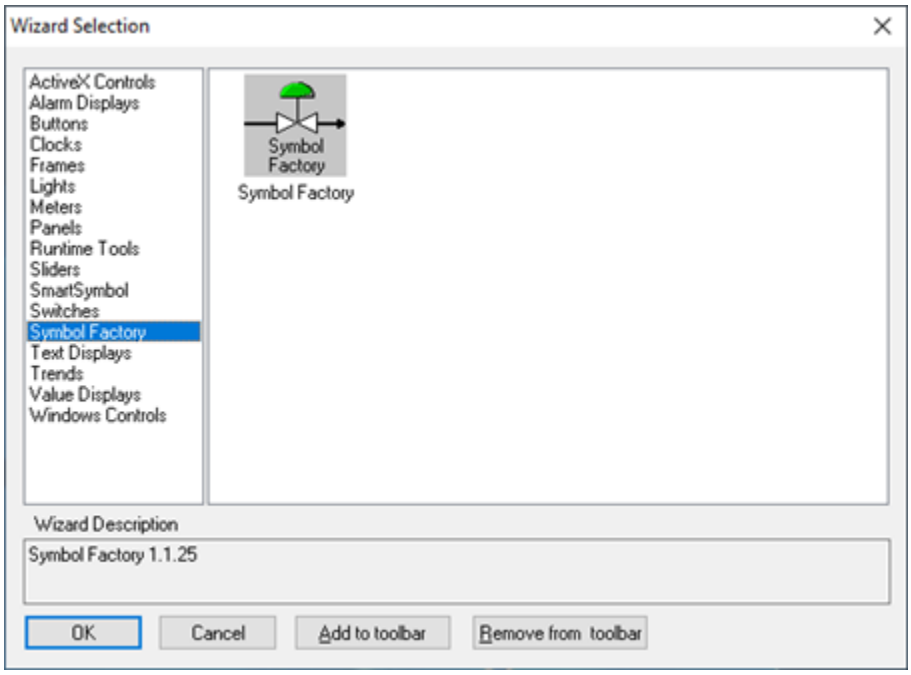
You place a Symbol Factory wizard in a window similar to placing other wizards.

Place a Symbol Factory wizard into a window

1. Open an application in WindowMaker.

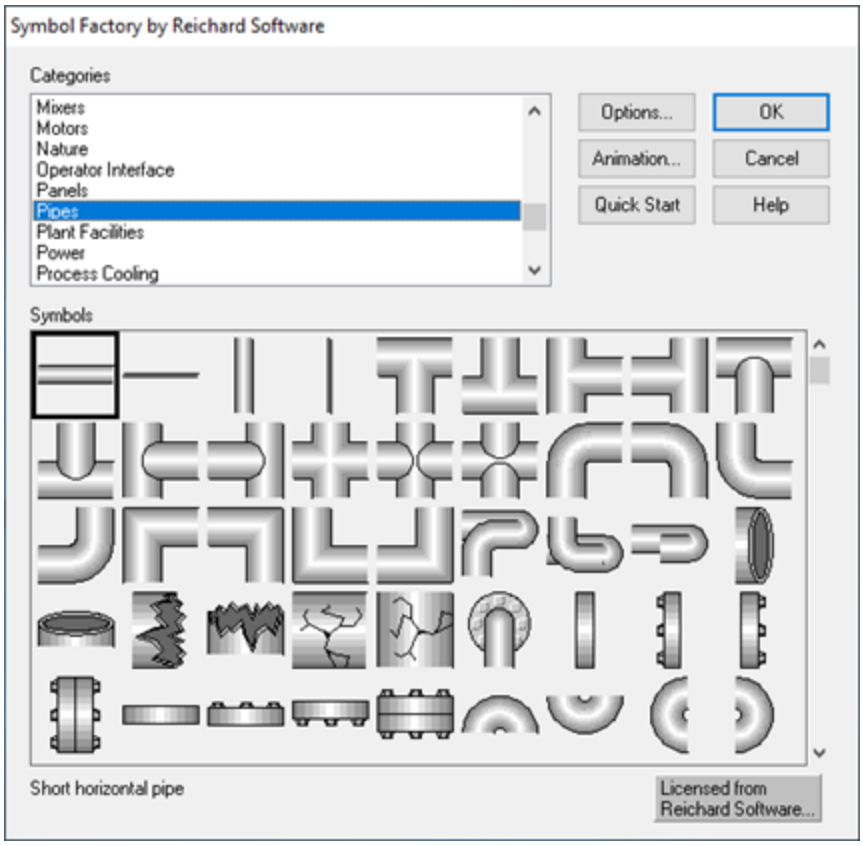
2. On the **Draw** menu, in the **Insert** group, select **Wizards**.

   The **Wizard Selection** dialog box appears.



3. In the list of wizards shown in the left pane, select **Symbol Factory.**
4. Select the Symbol Factory wizard in the display area and then select **OK**.
5. Select in the window to place the wizard.

   The **Symbol Factory by Reichard Software** dialog box appears.



6. In the **Categories** list, select a category. The **Symbol** window shows the wizards for the category you selected.
7. Select the wizard to place and then select **OK**.

Configure symbol options

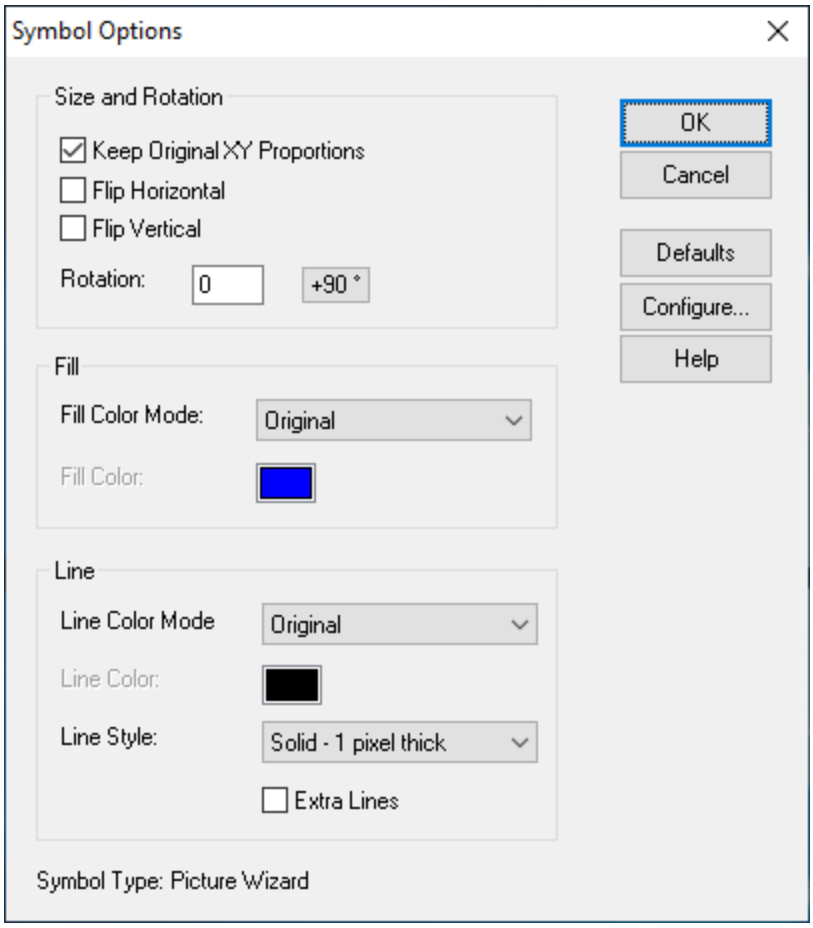
Wizard options vary for different wizards. Changing colors will impact the drawing speed of the bitmaps and textures since each pixel must be scanned and possibly changed.

Configure wizard options

1. Open a window containing a Symbol Factory wizard.
2. Double-click the wizard.

   The **Symbol Factory by Reichard Software** dialog box appears.
3. Keep the wizard selected and select **Options**.

   The **Symbol Options** dialog box appears. The image properties shown on the **Symbol Options** dialog box vary by the type of wizard you selected to edit. The following example shows the options that are available when you selected a picture wizard symbol.



Tip: If the **Enable alternatives to right mouse button** option is selected in the **Configure** Symbol Factory dialog box, the Edit Symbol button is shown and you can select it to configure the selected wizard.

4. In the **Size and Rotation** area, do any of the following:
   - Select the **Keep Original XY Proportions** check box to retain the original aspect ratio of the wizard.
   - Select the **Flip Horizontal** check box to flip the wizard horizontally.
   - Select the **Flip Vertical** check box to flip the wizard vertically.
   - In the **Rotation Type** box, type the number of degrees to rotate a wizard. Picture wizards can be rotated to any angle. Bitmap and texture wizards can only be rotated in 90 degree increments (0, 90, 180, or 270). Select the button to automatically increment the rotation angle by 90 degrees.
5. If you are configuring a picture wizard, in the **Line** and **Fill** areas, do any the following:
   - In the **Fill Color Mode** list, select a fill type. Double-click the **Fill Color** box to open the color palette.
   - In the **Line Color Mode** list, select a line color. Double-click the **Line Color** box to access the color palette.
   - In the **Line Style** list, select a line style.
   - Select the **Extra Lines** check box to add lines at the borders of gradients within the wizard.
6. If you are configuring a bitmap or texture wizard, in the **Effects** and **Change Colors** areas, do any the following:
   - Select the **Include Border** check box to create a black border around a wizard.
   - Select the **Include Shadow** check box to create a dark gray shadow behind the wizard.
   - Select each color box to open the color palette to change the colors in the wizard.

7. Select **OK**.

Animate a wizard

You can animate any Symbol Factory wizard. The Symbol Factory provides you with access to the most common animation links.

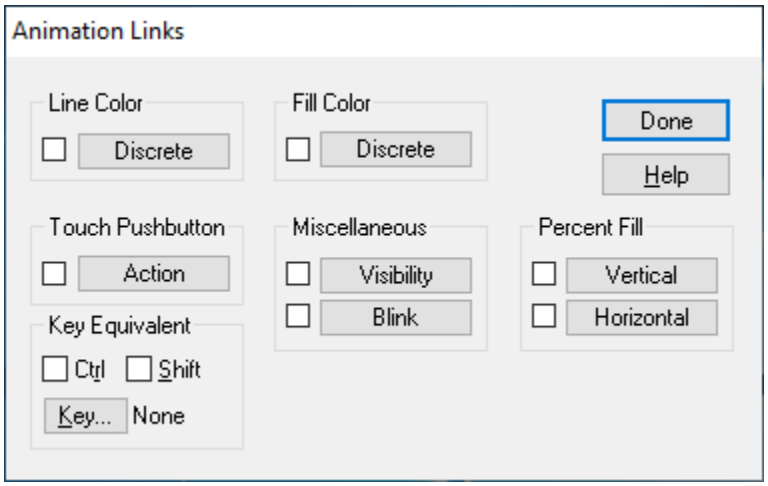
If you need another type of animation link, you must break the wizard and then animate it using the standard InTouch animation links.

Animate a wizard

1. Select a wizard in the Symbol Factory, or double-click the wizard if you have already pasted it into your window.

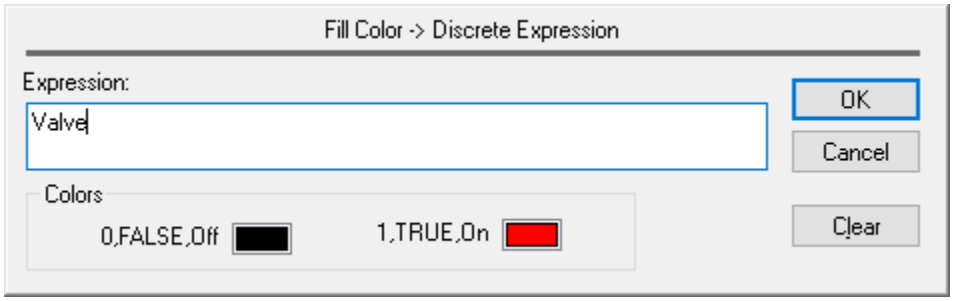
   The **Symbol Factory by Reichard Software** dialog box appears.
2. Select **Animation**.

   The **Animation Links** dialog box appears.



3. Select the button for each type of animation link to apply to the selected wizard.

   An expression dialog box appears.



4. In the **Expression** window, type the expression.

   Double-click in the window to open the **Select Tag** dialog box. If you use an existing tag, you can double-click the tag in your expression to open the **Tagname Dictionary** and see the tag definition.

   If you use an undefined tag, you are prompted to define it when you close the expression dialog box.
5. Configure the details for the type of animation link.
6. Select **OK**.

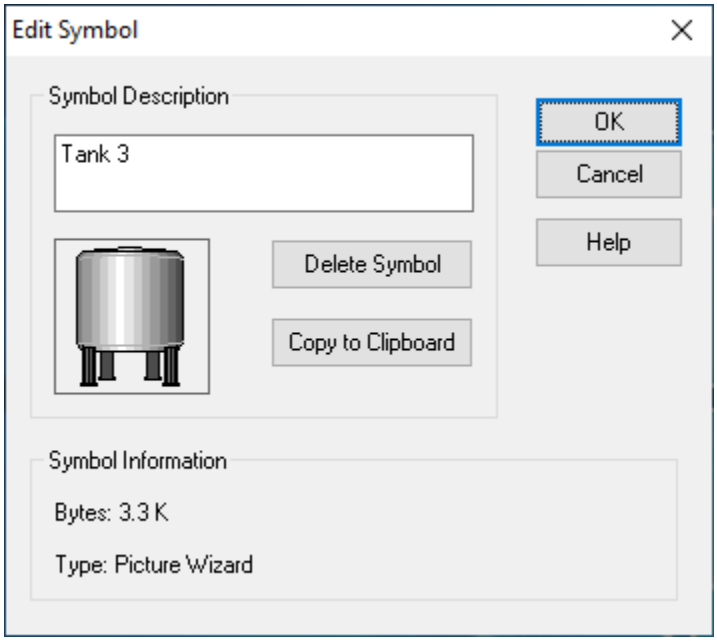
Edit a symbol

Yo can change the description in a wizard tool tip, delete a wizard, or copy a wizard to the Windows clipboard.

Edit a wizard

1. In the **Symbol Factory by Reichard Software** dialog box, select the category containing the wizard.
2. Right-click the wizard.

   The **Edit Symbol** dialog box appears.



3. Edit the wizard. Do any of the following:
   - In the **Symbol Description** box, type the tool tip text. The maximum description is 80 characters.
   - Select **Delete Symbol** to delete the wizard.
   - Select **Copy to Clipboard** to copy the wizard to the Windows clipboard. If the wizard is a picture wizard, it will be copied as a Windows metafile. If the wizard is a bitmap wizard or texture wizard, it will be copied as a Windows bitmap.
4. Select **OK**.

Break a wizard for editing

You can break a Symbol Factory wizard for individual editing. However, after you break a wizard, it loses its wizard properties. If you accidentally break a wizard, you can reassemble it by using the Undo tool.

Break a wizard

1. Open the WindowMaker.
2. On the **Animation** menu, in the **Cell** group, select **Break**.

Share a category of symbols on a network

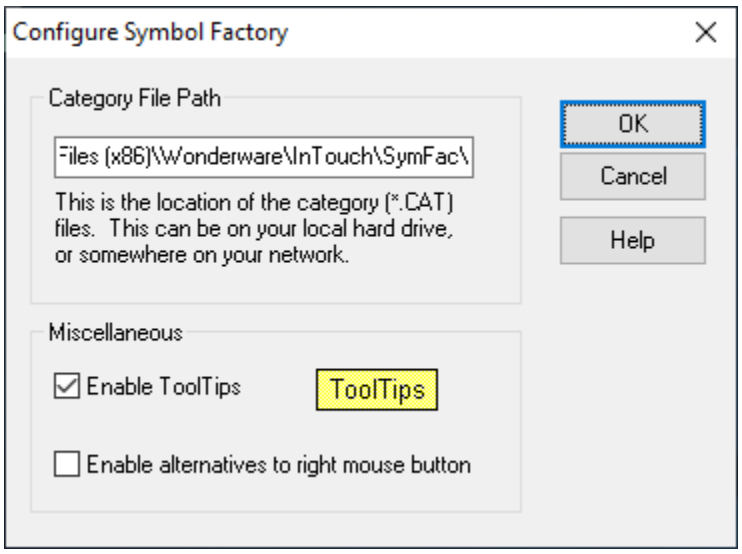
You can configure the Symbol Factory to allow multiple developers across a network to use and contribute to the category file of wizards.

Move the category file to a network folder

1. In the Symbol Factory dialog box, select **Options**.

   The **Symbol Options** dialog box appears.
2. Select **Configure**.

   The **Configure Symbol Factory** dialog box appears.



3. In the **Category File Path** box, type the full path to the network folder where the category file is saved.
4. Select **OK**.

Make a category read-only

When storing a wizard file on a network folder, you may want to make the category read-only to prevent other users from moving or renaming wizards.

Make a category read-only

- Set the file as read-only in Windows Explorer.

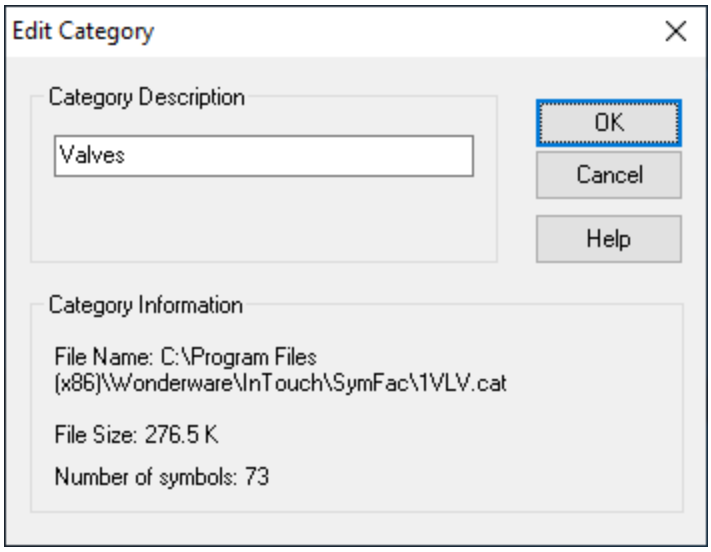
View category properties

You can view the category path, file size, and number of wizards.

View properties of a category

1. In the **Symbol Factory** dialog box, right-click the category in the **Categories** list.

   The **Edit Category** dialog box appears.



2. In the **Category Description** box, type the new description for the category, and select **OK**. The maximum length of the description is 40 characters.
3. In the **Category Information** area, view the properties.

Category Information	Description
File Name	The category (.cat) file path. By default, this path is C:\Program Files (x86)\Wonderware\InTouch\SymFac.
File Size	Size of the category file, in kilobytes.
Number of Symbols	Total number of wizards contained in the category. Maximum is 32,767.

4. Select **OK**.

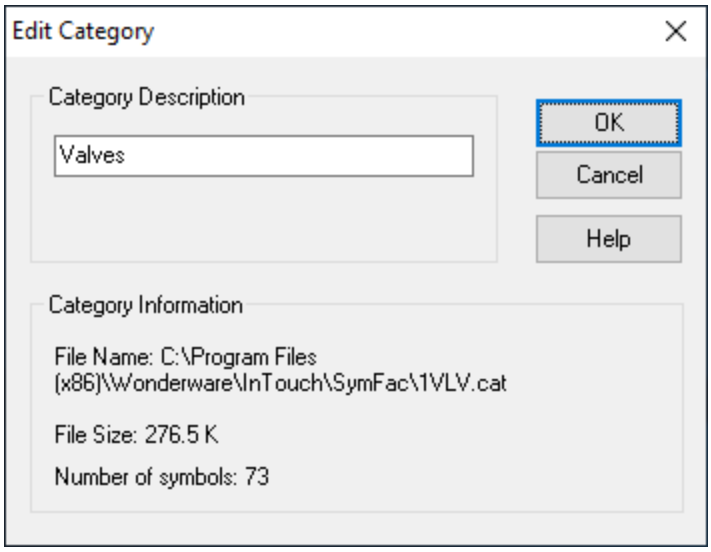
Edit an existing category

You can only edit the category name.

Edit an existing category

1. In the Symbol Factory dialog box, right-click the category in the **Categories** list.

   The **Edit Category** dialog box appears.



2. In the **Category Description** box, type the new description for the category, and select **OK**. The maximum length of the description is 40 characters.
3. Select **OK**.

Delete a category

Use Windows Explorer and delete the category (.cat) file by specifying the filename for the category.

Tip: You can verify the category filename in the Edit Category dialog box.

Configure Symbol Factory

When you configure the Symbol Factory, you can specify:

- Whether tool tips are shown when you select a wizard.
- Whether additional options should appear in the Symbol Factory **by Reichard Software** dialog box. By default, to edit category and wizard descriptions, you must right-click the item.
- The location for your category (.cat) files. All data for all wizards in each category is stored in one category file. For performance reasons, this path should contain only .cat files. To share wizards with other developers, set this path to a network folder. See Share a category of symbols on a network.

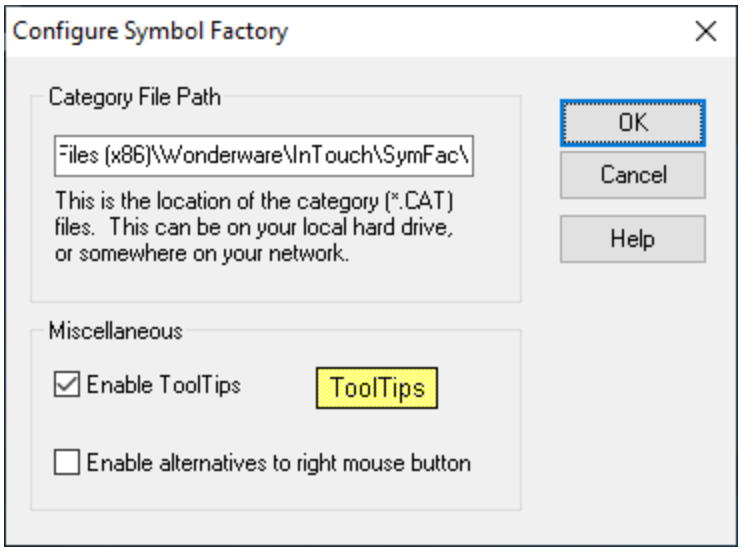
  Caution: Do not place category files in your local InTouch application folder. Instead, save category files to: C:\Program Files\Wonderware\intouch\symfac.

Configure Symbol Factory

1. In the Symbol Factory dialog box, select **Options**.

   The **Symbol Options** dialog box appears.
2. Select **Configure**.

   The **Configure Symbol Factory** dialog box appears.



3. Configure Symbol Factory. Do the following:
   - In the **Category File Path** box, type the location where you want to save your Symbol Factory category (.cat) files.
   - Select the **Enable ToolTips** check box if you want tool tips to be shown for wizards in the **Symbol Factory by Reichard Software** dialog box.
   - Select the **Enable alternatives to right mouse button** check box if you want buttons added to the **Symbol Factory by Reichard Software** dialog box. You can use these buttons instead of the right mouse button for editing categories and wizards:

     Edit Category

     Shows the Edit Category dialog box for a selected category.

     Edit Symbol

     Shows the Edit Symbol dialog box for a selected wizard.
4. Select **OK**.

Troubleshoot

If you accidentally uninstall the Symbol Factory wizard, you need to install it again. For more information on installing wizards, see Wizards in *AVEVA™ InTouch HMI Application Development Guide*.

If you accidentally delete a wizard and you want it back, you must retrieve it.

Retrieve a deleted wizard from a category

1. Rename the file ~cat.bak to temp.cat.
2. Run Symbol Factory and see if the deleted wizard is back. Move it to its original category, then delete the temp.cat file.
3. If the above step did not work, hold down the CTRL key while you right-click the category with the deleted wizard. This compacts the category file and creates a fresh backup ~cat.bak.

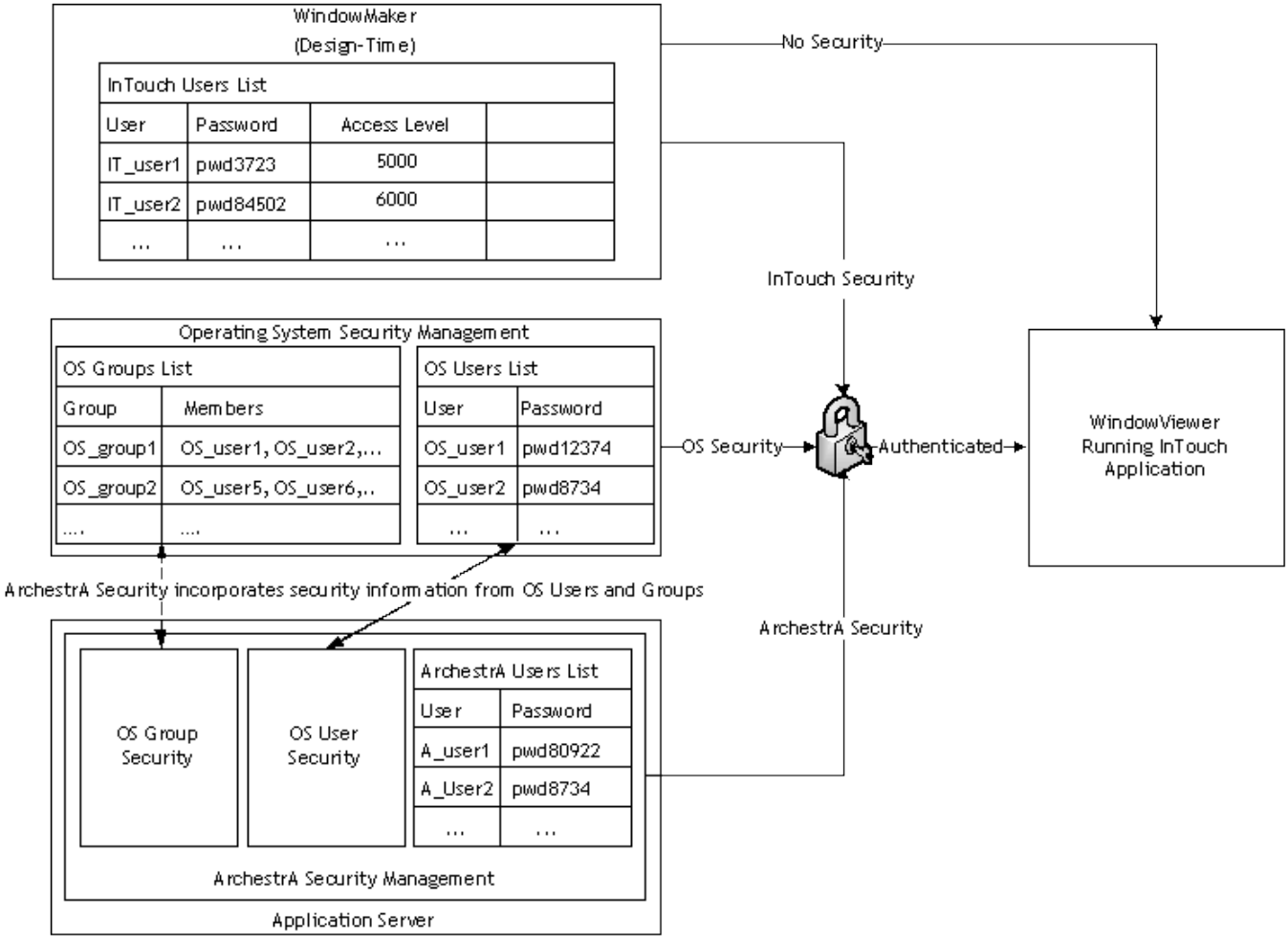
Perform the previous steps until you find your deleted wizard.

Secure InTouch

You can protect your InTouch applications using:

- Traditional InTouch-based security
- Operating system-based security
- ArchestrA-based security

The following figure shows the relationship between the three types of security.



InTouch Security Features

To protect your InTouch application while it is running, you can:

- Set an inactivity time-out period
- Lock keys
- Hide menus

Configure an inactivity time-out

You can configure WindowViewer to automatically log off an inactive operator from an InTouch application. An operator must log on again after being logged off for inactivity. Setting an automatic inactivity log off period prevents unauthorized access to your InTouch application when operators leave their workstations unattended.

A timer measures the period the operator has not interacted with the running InTouch application. The timer resets each time the operator uses a mouse or any other input device to enter data. If the timer expires, the user is automatically logged off.

Note: The inactivity timer does not reset for Active-X controls, and OLE Automation controls.

Automatically logging off an operator is a two-step process:

1. WindowViewer sets the $InactivityWarning system tag to 1 when the operator's inactivity period exceeds a specified warning period. You can use the $InactivityWarning tag in a condition QuickScript to show a window that warns the operator about the pending log off for inactivity. The operator stays logged on by responding before the specified time-out period occurs. When the operator takes some action, the $InactivityWarning tag and inactivity timer are reset to zero.
2. If the operator fails to respond after the inactivity warning, the $InactivityTimeout system tag is set to 1 when the time-out period has been reached. When $InactivityTimeout is 1, WindowViewer equates the logged on operator name to the reserved name None and sets the $AccessLevel security tag to 0. The user is automatically logged off.

You can use the time-out feature independently of the warning feature.

To configure an inactivity time-out

1. Open WindowMaker.
2. On the **File** menu, point to **Configure**, and then select **WindowViewer**.

   The **WindowViewer configuration** screen appears.
3. In the **Inactivity in secs** area, configure the warning and time-out values. Do the following:
   - In the **Warning** box, type the number of seconds that can elapse before the $InactivityWarning tag is set to 1.
   - In the Timeout box, type the number of seconds that can elapse before the $InactivityTimeout tag is set to 1 and the user is automatically logged off.

4. Select **OK**.
5. To show a window named "Warning - Logoff Pending" after the inactivity warning time elapses, create a condition script with "$InactivityWarning" as the condition and the following script body:

   ```
   Show "Logoff Pending";
   ```
6. To show a window named "Logged Off" after the inactivity timeout elapses, create a condition script with "$InactivityTimeout" as the condition and the following script body:

   ```
   Show "Logged Off";
   ```

$InactivityTimeout system tag

Indicates that the time configured for inactivity elapsed.

Category

security

Usage

$InactivityTimeout

Remarks

Set to 1 when the inactivity timer elapses. For more information on setting the log off time, see Configure an inactivity time-out.

Note: The inactivity timer does not reset for ActiveX controls, OLE and automation controls.

Data Type

Discrete (read only)

See Also

$InactivityWarning

Example(s)

The following example is an "on true" condition script:

```
If $InactivityTimeout == 1 THEN
   Show "Logged Off";
ENDIF
```

See Also

$InactivityWarning

$InactivityWarning system tag

Indicates that the time configured for warning the user that log off is about to occur elapsed.

Category

security

Usage

$InactivityWarning

Remarks

Set to 1 when the inactivity warning time elapses. The inactivity timer is reset only by mouse or keyboard activities. For more information on setting the log off warning, see Configure an inactivity time-out.

Note: The inactivity timer does not reset for ActiveX controls, OLE automation controls, and SPC wizards.

Data Type

Discrete (read only)

Example(s)

The following example is an "on true" condition script.

```
If $InactivityWarning == 1 THEN
   Show "Logoff Pending";
ENDIF;
```

See Also

$InactivityTimeOut

Lock system keys

You can restrict operator access to standard Windows functions by disabling system keys on the computer running an InTouch application. For example, you can prevent an operator from using the Windows CTRL+ALT+DEL key combination to show the **Task Manager** dialog box. Disabling system keys prevents operators from switching from the InTouch HMI to another Windows application.

WindowViewer has key filter options that set the default state of system keys when an InTouch application starts. A key filter disables a system key when it is active.

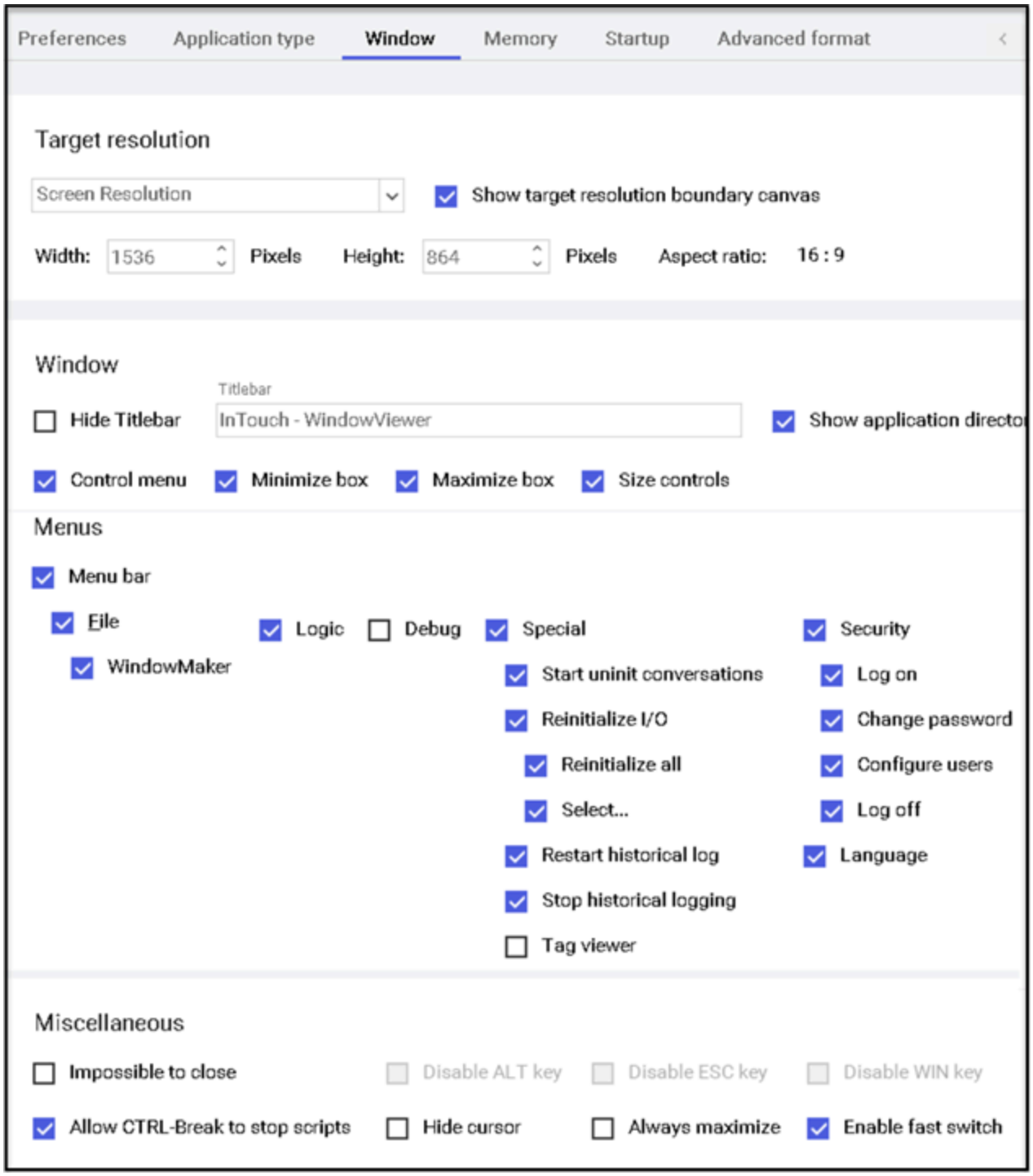
Disable system keys based on what tasks you expect your various InTouch users to complete. Most function keys should be disabled for operators. Administrators still need function keys for their InTouch tasks.

You can write a script that enables or disables system keys based on the access level of the person logging on to WindowViewer. Use the EnableDisableKeys() function in a script to selectively enable or disable Windows function keys.

Enable key filters

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.
3. Select the **Window** tab.



4. In the **Miscellaneous** area, disable WindowViewer system keys. Do the following:
   - Clear the **Enable Fast Switch** check box to remove the **Development** button from WindowViewer that switches the user to WindowMaker.
   - Select the **Disable ALT key** check box to disable the ALT key on the computer running the InTouch application.
   - Select the **Disable WIN key** check box to disable the WIN key on the computer running the InTouch application.

- Select the **Disable ESC key** check box to disable the ESC key on the computer running the InTouch application.

5. Select **OK**.
6. Write a script that runs when WindowViewer starts running the InTouch application.

   The script should include statements to dynamically lock or unlock key based on the access level of the person who logged on to WindowViewer.

   Include the EnableDisableKeys() function within the script to enable/disable the ALT, ESC, and WIN keys. The EnableDisableKeys() function enables or disables system keys based on the discrete values of its arguments:

   ```
   EnableDisableKeys(AltKey,EscKey,WinKey);
   ```

   An argument value of 1 enables the key filter to disable the key.

EnableDisableKeys() function

Enables/disables key filters for the Alt, Escape, and Windows keys.

Category

View

Syntax

```
EnableDisableKeys(AltKey, EscKey, WinKey);
```

Parameters

AltKey

Integer to enable or disable key filters for the Alt key:

1 = enable filter (disable Alt key)

0 = disable filter (enable Alt key)

EscKey

Integer to enable or disable key filters for the Escape key:

1 = enable filter (disable Esc key)

0 = disable filter (enable Esc key)

WinKey

Integer to enable or disable key filters for the Windows key:

1 = enable filter (disable Win key)

0 = disable filter (enable Win key)

Remarks

Disabling the Alt key also disables the Win+L key combination (for locking the Windows desktop). Win+L is the shortcut for another combination of keys that involves the Alt key. Thus, disabling the Alt key also disables the shortcut for locking the Windows desktop.

Disabling the Esc key disables it for all actions.

Example(s)

```
EnableDisableKeys(0,0,0); // enable all three keys
EnableDisableKeys(1,1,1); // disable all three keys
EnableDisableKeys(0,0,1); // enable Alt and Escape keys, disable Windows key.
```

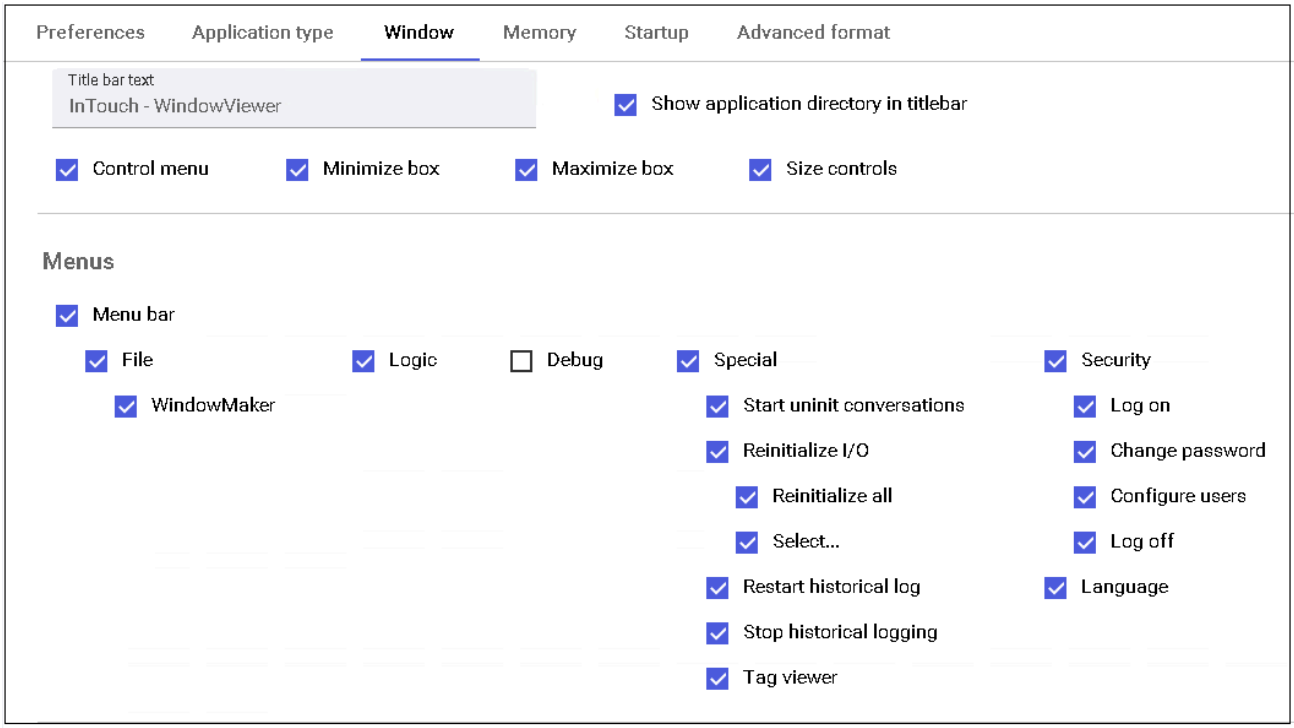
Hide menu items at runtime

You restrict operator access to WindowViewer menus and commands by hiding them while an InTouch application is running.

Hide menu items at runtime

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **WindowViewer**.

   The WindowViewer configuration screen appears.
3. Select the **Window** tab.



4. In the **Menus** area, select the WindowViewer menus and commands that you want to be visible to an operator. Do the following:
   - Clear the WindowMaker check box to make the WindowMaker command unavailable from the WindowViewer **File** menu. Clearing this option does not affect the fast switch to WindowMaker.
   - Clear the Logic check box to hide the WindowViewer **Logic** menu that contains commands to start and stop QuickScripts.

     Note: You can use the $LogicRunning system tag to enable the operator to start and stop all QuickScripts.

If you select the **Allow CTRL-Break to stop scripts** option, the operator can stop all QuickScripts from running regardless of whether the **Logic** menu appears or not.
Currently executing asynchronous QuickFunctions cannot be stopped. However, you can prevent operators from starting new asynchronous QuickFunctions.

- Select the Debug check box if you are testing your application. Otherwise, clear the **Debug** check box to hide the **Debug** menu during run time.
- Clear the **Special** menu items to prevent operators from stopping ongoing InTouch functions like logging and I/O connections.
- Clear the **Security** check box to prevent operators from changing security related options.

5. In the **Window** area, select the window controls that you want to make available to an operator from WindowViewer. These options affect the window that is running the InTouch application. Do the following:
   - Clear the **Control Menu** check box to hide the controls that close, minimize, maximize, and resize the window.
   - Clear the **Minimize Box** check box to prevent an operator from minimizing the window.
   - Clear the **Maximize Box** check box to prevent an operator from maximizing the window.
   - Clear the **Size Controls** check box to prevent an operator from resizing the window.
6. In the **Title** bar area, configure the title bar of the window running the InTouch application. Do the following:
   - In the **Title Bar Text** box, type a title to be shown in the WindowViewer title bar.
   - Select the **Show Application Directory** check box to include the path to the InTouch application's folder in the title bar.
   - Select the **Hide Title Bar** check box to hide the window's title bar.
7. In the **Miscellaneous** area, do the following:
   - Select the **Impossible to Close** check box to prevent an operator from closing the WindowViewer window running the InTouch application. Selecting this option disables the window's **Close** button.

     If you want to hide the **Close** button, clear the **Control Menu** check box in the **Window** area.
   - Clear the **Allow CTRL-Break to stop scripts** check box to disable the CTRL + BREAK key combination that enables operators to stop QuickScripts.

     Note: Currently executing asynchronous QuickFunctions cannot be stopped. However, you can prevent new asynchronous QuickFunctions from executing.
   - Select the Hide Cursor check box to hide the mouse pointer during run time. This is useful if you are designing the application for a touch-screen.
   - Select the Always Maximize check box to keep the window running the InTouch application fully maximized on the operator's screen.
8. Select **OK**.
9. Restart WindowViewer to apply your changes.

Authentication and authorization based security

InTouch security is a two-step process of first determining if the person attempting to use an application is recognized as a valid user. The second step determines what InTouch privileges are granted to an authenticated user.

Compare authentication and authorization

Authentication is the process of verifying the identity of the user. Typically, operators enter a user name and password to authenticate themselves before using an InTouch application. All three types of security verify the user's credentials during the logon process as part of the authentication process.

Authorization is the process of determining if an authenticated user has access to the requested resources. Typically, access to InTouch functions is granted based upon the user's membership in a group or assigned access level.

Different authentication security modes

All types of InTouch security authenticate users during the logon process with a user name and password combination. Each type of security provides a different mechanism to verify the user name and password during the authentication process.

Use InTouch-based security

Applying security to your application is optional. By default, an InTouch application is not secured. However, you can restrict which functions an operator is allowed to perform by linking those functions to internal tags. In addition, when you establish security on your application, audit trails can be created that associate alarms and events to the operator logged on to the InTouch HMI.

Set the Security Type to InTouch

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **Security**.

   The **Security** configuration screen appears.
3. Select the Security type as **InTouch**.

   Configure-Secuirty-InTouch-Warning

When you add the Security type as InTouch, the **Configure users** section is enabled. Security is based on operators authenticating themselves by entering a user name and password to log on to an InTouch application. You must assign user name, password, and access level for each operator.

To configure users

1. Select the Add icon (Alt+A).

   The **Create user** dialog appears.

   Embedded Image (65% Scaling) (LIVE)
2. In the **User name** box, type the name that you want to assign to the operator.
3. In the **Password** box, type an operator password up to a maximum of 29 characters.
4. In the **Confirm password** box, re-enter the same password.
5. In the **Access level** box, type the operator's access level (lowest = 0 to highest = 9999).

After you add a new user name to the security list and restart WindowMaker or WindowViewer, the default user name is automatically reset to None with an access level of 0, which prevents access to the Configure Users

command in both WindowMaker and WindowViewer. However, the Administrator account and password remain and can still be used.

After an operator logs on to the application, access to any protected function is granted upon verification of the operator's password and access level against the value specified for the internal security tag linked to the function.

For Standalone applications, only users with Administrator privileges are allowed to open and edit applications in InTouch WindowMaker. If a user without administrator privileges attempts to launch InTouch WindowMaker, an error dialog box appears, informing users that they need administrative privileges to proceed. Users without administrative privileges can launch WindowMaker via IDE for Managed applications.

Use Operating System-based security

An operating system-based authentication method inherits enforcement of some account policies from the Windows operating system, while other policies are enforced from the InTouch HMI. Password policies such as maximum and minimum password age and minimum password length are enforced by the operating system.

User names used during installation act as a part of the operating system. The Windows domain must be set up with the desired account policies to enforce these standards. The InTouch HMI enforces the inactivity time-out period.

In the operating system-based authentication method, user names can be chosen from the list of users associated with a Windows Network Domain or Workgroup. Each user name has an assigned access level that determines the user's authorization for a given activity. Because the operating system manages passwords internally, the InTouch HMI does not store passwords on the node hosting the application.

Operating system-based security uses the InTouch AddPermission() script function to define and maintain a list of users and their corresponding access levels. This list, created after the execution of the AddPermission() call, is written to disk. The file containing the authentication details of users is not copied to NAD client nodes.

The operator can log on to the application by executing the **Log on** menu command under **Security** in the WindowViewer **Special** menu (if the **Special** menu is shown), or you can create a custom log on window with touch-sensitive input objects that are linked to internal security tags.

The commands used to establish security on an application are located under **Security** in both WindowMaker and WindowViewer. The security commands are used to log on and off the application, change passwords, and to configure the list of valid user names, passwords, and access levels.

For example, you can control access to a window, the visibility of an object, and so on, by specifying the logged on operator's access level must be greater than 2000.

Use ArchestrA-based security

When you configure a node to use ArchestrA security, the InTouch HMI uses methods and dialog boxes from Application Server for logon and logoff operations. Users are configured on the Application Server Galaxy Repository node. For more information, see the Application Server documentation.

ArchestrA security enables you to easily define users and assign the operations they are allowed to perform. Define security permissions in terms of the operations the users can perform using automation objects. The basic approach consists of the following steps:

1. Define the security model.
2. Organize the automation objects according to the security model for protection.

3. Define the users according to the security model.

The system administrator defines the system users by creating corresponding user profiles. The system administrator then assigns one or more roles to each user by selecting from a list of user roles predefined in the security model.

If you are using InTouch with ArchestrA-based security, the maximum number of characters for a password is 31.

InTouchView users are normally authenticated by means of a password-based log-on.

Use Smart Cards for authentication

A Smart Card is a pocket-sized card with embedded integrated circuits. The card has secure storage for data, including private keys and public key certificates. The card holder is authenticated through a Personal Identification Number (PIN) and can be authorized to access only particular data on the card.

You can configure an InTouch application to support Smart Cards for user authentication. Instead of the application requiring a username, password, and domain to be provided, the Smart Card certificate and associated PIN number can be used for authentication. You can also choose to log on with your name, password, and domain instead of the Smart Card.

Operations that require user authentication, such as logging on or secured/verified writes, can also take advantage of Smart Card authentication. For more information, see Use Secured and Verified Writes.

Set up Smart Card authentication

You must do the following to set up Smart Card authentication:

- Configure the InTouch application to use either InTouch OS or ArchestrA OS security. The ArchestrA security can be either user-based or group-based. You configure ArchestrA security using the System Platform IDE. For more information, see the IDE documentation.
- Join the WindowViewer computer to the correct domain for your network.
- Within WindowMaker, enable Smart Card authentication for the InTouch application. For more information, see Enable Smart Card authentication in WindowMaker.
- Configure the Smart Cards for the domain where you will use them.
- Install card drivers on the WindowViewer computer. Smart Card and their drivers are hardware-specific. For information on installing and setting up your Smart Card reader, refer to the documentation for your specific reader.
- Connect the Smart Card reader to the appropriate port of the WindowViewer computer. For instructions, see the documentation that comes with the Smart Card.
  - More than one Smart Card reader is required to perform verified write functions, which involve more than one user.
  - To use Smart Card with Terminal server and RDP clients, a Smart Card reader must be attached to the client systems to enable Smart Card authentication. To connect a Smart Card reader to a Terminal Server using RDP, you need to make sure that the RDP client connection settings have the **Smart Card** option enabled under **Local Devices and Resources**.

Enable Smart Card authentication in WindowMaker

You must enable Smart Card authentication in WindowMaker before you can use the Smart Card reader for authentication.

Configure the Smart Card Option

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **Security**.
3. Select a security type from the available options: **None**, **InTouch**, **OS**, and **ArchestrA**.
   - If you select **ArchestrA**, be sure that you have configured ArchestrA OS security (OS user-based or OS group-based) using the IDE.
   - If you select the **OS** or **ArchestrA** security types, the **Smart card authentication** check box is enabled. Select the check box to enable smart card authentication.

Log on with your Smart Card

You can use a smart card to log on to InTouch WindowViewer. You must have an application with smart card authentication enabled to use it to log on to the InTouch application.

Your smart card must contain at least one certificate that is configured in your domain. A smart card reader must be attached to the computer running WindowViewer. You will be required to enter the PIN of the smart card you are using.

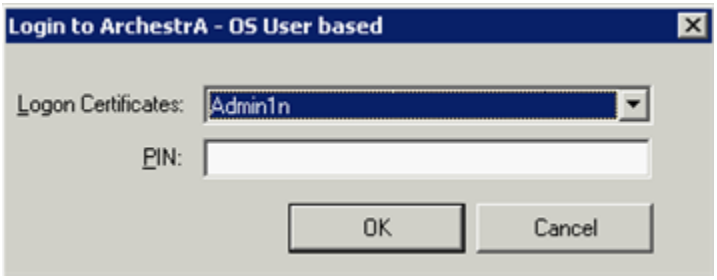
If a smart card is not detected in the reader when you try to log on, you are prompted to authenticate using your user name and password instead.

You can use smart card for authentication for secured and verified data writes. For more information, see Use Secured and Verified Writes.

Log on with your smart card

1. Run WindowViewer.
2. Insert your smart card if not already inserted.
3. On the **Special** menu, point to **Security**, and then select **Log On**.

   The Login dialog box appears.



   If you have inserted your smart card, your log on certificate—the domain and the user name—is shown in the dialog box.

   The smart card log on dialog box also appears if the LogonCurrentUser() or PostLogonDialog() functions execute from scripts in WindowViewer. These functions are available only in InTouch scripting, not in ArchestrA client scripting.

4. In the **PIN** box, enter the PIN for the smart card being used

   If a smart card is not available, the system will prompt you to log on with your user ID and password.
5. Select **OK**. You are logged on to WindowViewer.

Note: After you log on as a smart card user, you must keep the card in the smart card reader. If you remove it, the system logs you off.

Use Secured and Verified Writes

You can configure an InTouch application so that operators can write data to Galaxy attributes that are configured with certain security classifications:

- A "secured write" classification requires the run-time operator to enter his or her credentials to complete the write operation.
- A "verified write" classification requires two signatures. An operator can write data if the appropriate credentials are provided, but authorization is also required by an additional verifier to complete the write operation.

Secured and verified writes require the following:

- Security must be enabled for the Galaxy.
- ArchestrA security must be enabled for the InTouch application.
- Run-time operators must have the appropriate operational permissions configured within the Galaxy:
  - An operator must have the "Can Modify Operate Attributes" operational permission to perform either a Secured Write or a Verified Write.
  - A verifier must have the "Can Verify Writes" operational permission to confirm the verified write.

Regardless of who is currently logged on as the run-time user for the InTouch application, Secured or Verified Writes always require user authentication. You can modify attributes configured with Secured or Verified Write security classification even if you are not the logged-on user. This does not affect the session of the logged-on user.

Important: For Galaxies that have security enabled and are migrated to Application Server version 3.5, the "Can Modify Operate Attributes" operational permission setting will be copied to the "Can Verify Writes" attribute. Starting with Application Server 3.5, Galaxies have the "Can Verify Writes" operational permission setting disabled by default.

Within InTouch Tag Viewer, a run-time user can only write to an indirect tag with a reference to an Application Server attribute.

You can use smart cards for authentication for secured and verified data writes. For more information, see Use Secured and Verified Writes.

Performe a Secured Write

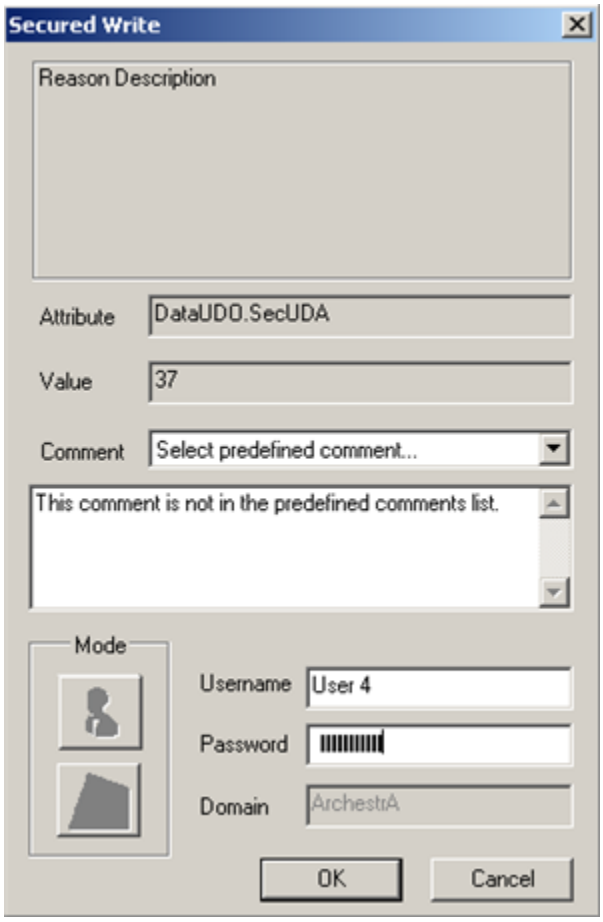
If you attempt to modify the value of an Application Server Galaxy attribute that has been configured with the Secured Write security classification, you must authenticate yourself using either a valid security account (domain name, username, and password) or a smart card. The smart card option is only available to you if a smart card reader is attached to the WindowViewer computer.

You must have the "Can Modify Operate Attributes" operational permission within the Galaxy to perform a secured write.

Your authentication for a secured write does not affect the session of the currently logged-on user. If you previously logged on with your smart card, you must re-authenticate yourself.

Perform a Secured Write

1. Attempt to modify the value of an attribute configured with the **Secured Write** security classification. The **Secured Write** dialog box appears. The **Mode** buttons are disabled if no smart card is available.



2. Add a comment for the write action by selecting from the predefined **Comment** list or by entering a comment in the **Comment** text box. The comment is limited to 200 characters.

   You can predefine a list of comments using the SignedWrite() script function, or you can enter a new comment in the **Comment** text box. The predefined comments list is only accessible when using the SignedWrite() script function.

3. If you are authenticating using a network user account, the user account options are shown.

   Do the following.

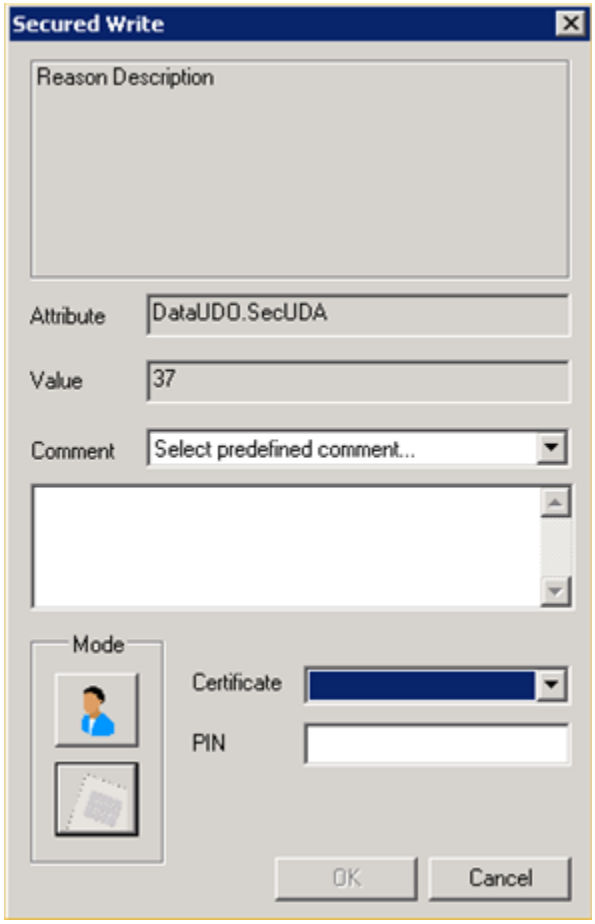
   a. In the **Username** box, type your user name. The name of the currently logged-on user is shown by default. If no user is currently logged on, the box is blank.

   b. In the **Password** box, type the password associated with the user name.

   c. In the **Domain** box, type the domain name.

   d. Select **OK**.

4. If you are authenticating using a smart card, the smart card options appear.



Do the following to authenticate using a smart card.

a. In the **Certificate** list, select your smart card certificate. The certificate list appears as domain_name/user_name. For a certificate to appear in the list, smart card must be currently inserted into a reader attached to the computer. The certificate of the currently logged-on user is shown by default. If you insert or remove a card while the **Secured Write** dialog box is open, the certificate list automatically updates.

b. In the **PIN** box, the PIN for the smart card being used.

c. Select **OK**.

When a smart card is present, if you want to authenticate using your name, password, and domain instead, select the Mode button. Go to Step 3.

Perform a Verified Write

If you attempt to modify the value for an Application Server Galaxy attribute that has been configured with the Verified Write security classification, you must authenticate yourself using either a valid security account (domain name, username, and password) or a smart card. The write must also be verified by another person.

- The operator must have the "Can Modify Operate Attributes" operational permission to perform the Verified

Write.

- The verifier must also have the "Can Verify Writes" operational permission to confirm the Verified Write.

Your authentication for a verified write does not affect the session of the currently logged-on user.

The smart card option is only available if a smart card reader is attached to the WindowViewer computer. You can use smart cards for logging in as either an operator, a verifier, or both, but the operator and the verifier must be two different people. If you previously logged on with your smart card, you must re-authenticate yourself.

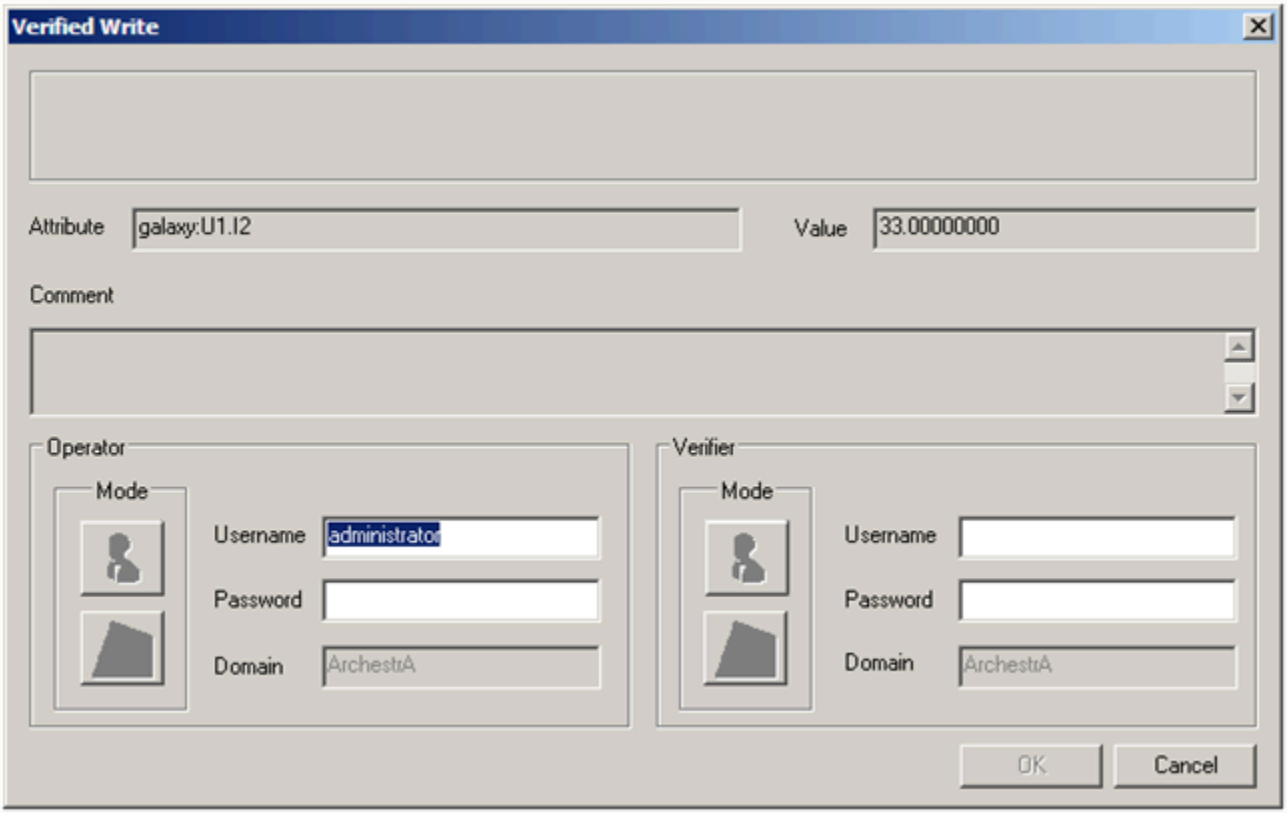
You have the following options:

- You can use two smart card readers and two smart cards.
- If you have only one smart card reader available, you can use one smart card reader with one smart card for the Operator or for the Verifier. When the Operator logs on using certificate number and PIN, the Verifier needs to log on using the username and password or vice versa.
- You can use username and password authentication for both the operator and the verifier.

Perform a Verified Write

1. Attempt to Modify the value of an attribute configured with the **Verified Write** security classification.

   The **Verified Write** dialog box appears. The **Mode** buttons are disabled if no smart card is available.



2. Add a comment for the write action by selecting from the predefined **Comment** list or by entering your own comment in the **Comment** box. The comment is limited to 200 characters.

   The predefined comments list is only accessible when using the SignedWrite() script function.

3. If you are authenticating using a network user account, the user account options are shown.

Do the following.

a. In the **Username** box, type your user name. The name of the currently logged-on user is shown by default. If no user is currently logged on, the box is blank.

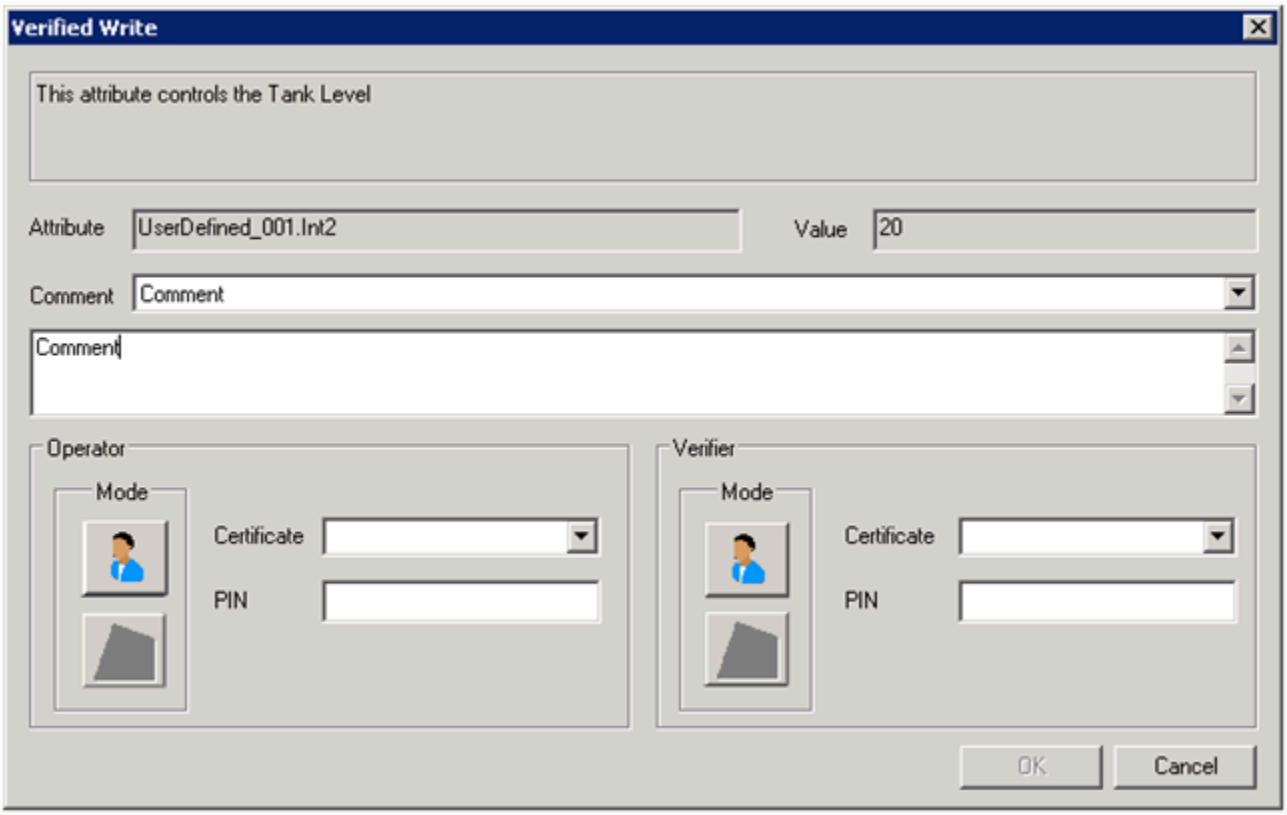
b. In the **Password** box, type the password associated with the user name.

c. In the **Domain** box, type the domain name.

d. Select **OK**.

e. To Authenticate using a smart card instead, select the **Certificate** button. Go to Step 4.

4. If you are authenticating using a smart card, the smart card options are shown.



Do the following to authenticate using a smart card.

a. In the **Certificate** list, select your smart card certificate. The certificate list appears as domain_name/user_name. For a certificate to appear in the list, smart card must be currently inserted into a reader attached to the computer. The certificate of the currently logged-on user is shown by default, if the user logged on using a smart card. If you insert or remove a card while the **Secured Write** dialog box is open, the certificate list automatically updates.

b. In the **PIN** box, type the PIN for the smart card being used.

c. Select **OK**.

d. When a smart card is present, if you want to authenticate using your name, password, and domain instead, select the Mode button. Go to step 3.

Customize the Secured/Verified Write dialog box

You can use the SignedWrite() script function to configure the following in the **Secured Write** or **Verified Write** dialog box:

- Show a reason message
- Populate the predefined **Comment** list

- Allow editing in the **Comment** text box

For information about the SignedWrite() function and how to use it, including syntax, parameters, and detailed examples, see the Industrial Graphic Editor User Guide.

Work with the SignedWrite() function at runtime

It is possible to use the SignedWrite() function to directly assign a value to an attribute that requires a Secured or Verified Write signature.

When you configure a value with Secured or Verified Write security classifications and modify it, the **Secured or Verified Writes** dialog box appears. Depending on how the value is modified, the content appearing in the **Secured or Verified Writes** dialog box differs.

- If the value is modified using the SignedWrite() function, then the **Secured or Verified Writes** dialog box shows options based on the parameter settings from the function.
- If the value is modified by a user operation, then the reason message area shows the field attribute description, if there is one. If the attribute is not a field attribute or does not have a description, then the reason message area shows the description of the ApplicationObject to which the attribute belongs. The predefined **Comment** list is not available.

You can view the reason message in the **Secured Write** or **Verified Write** dialog box when you try to modify the value of the attribute in InTouch WIndowViewer. The dialog box displays the name of the attribute and the new value that is written to the attribute.

Note: The reason description and the predefined **Comment** list and box are shown in the **Secured Write** or **Verified Write** dialog box only in InTouch WindowViewer and not in Tag Viewer.

Manage users and setting their authorization levels

To implement security for the group of users who need to use the InTouch HMI, you must:

- Assign user name and password authentication credentials to each user.
- Assign an InTouch authorization level (access level) to each user.

Configure InTouch security authentication and authorization

For each of your operators, you need to assign a user name, password, and access level.

The **None** and **Administrator** names are reserved and only the password of the Administrator can be changed (the default is `wonderware`). After you configure user names for your application, change the Administrator password. The Administrator default access level (9999) is the highest and allows access to all InTouch functions including the **Configure Users** command.

You can also link a User Input - Discrete button to the $ConfigureUsers tag to allow an authorized operator with an access level of equal to or greater than 9000 to access the **Configure Users** dialog box to edit the security user name list. When the operator selects the button, the value of the $ConfigureUsers tag is set to 1 and the **Configure Users** dialog box appears. When the operator closes the dialog box, the system resets the value to 0. This is a system discrete tag intended for write operation only.

Note: The $ConfigureUsers tag only works if the security type is set to InTouch. It does not work for ArchestrA-based and operating system based security.

Configure security for operators of your application

1. Open WindowMaker.
2. On the **File** menu, select **Configure**, and then select **Security**.

   The Security configuration screen appears.
3. Select the Security type as **InTouch**.

   The Logon dialog appears.
4. Enter the User name and password.

   Note: Only users with administrator accounts or with access level greater than 9000 can configure operators for InTouch.

   The **Configure users** section is enabled.
5. Select the Add icon.

   The **Create user** dialog appears.

   Embedded Image (65% Scaling) (LIVE)
6. To add a security account, do the following:
   a. In the **User Name** box, type the name that you want to assign to the operator.
   b. In the **Password** box, type an operator password up to a maximum of 29 characters.
   c. In the **Access Level** box, type the operator's access level (lowest = 0 to highest = 9999).
   d. Select **Add** to add the user name to the InTouch security list.
7. To change a user name, select the entry from the Configure users grid.
8. Make the required changes, and then select **Update**.
9. To delete a user name, select the entry from the Configure users grid. and then select **Delete**.
10. Select **OK**.

Change an InTouch operator password

Operators can change their passwords from the WindowMaker or the WindowViewer.

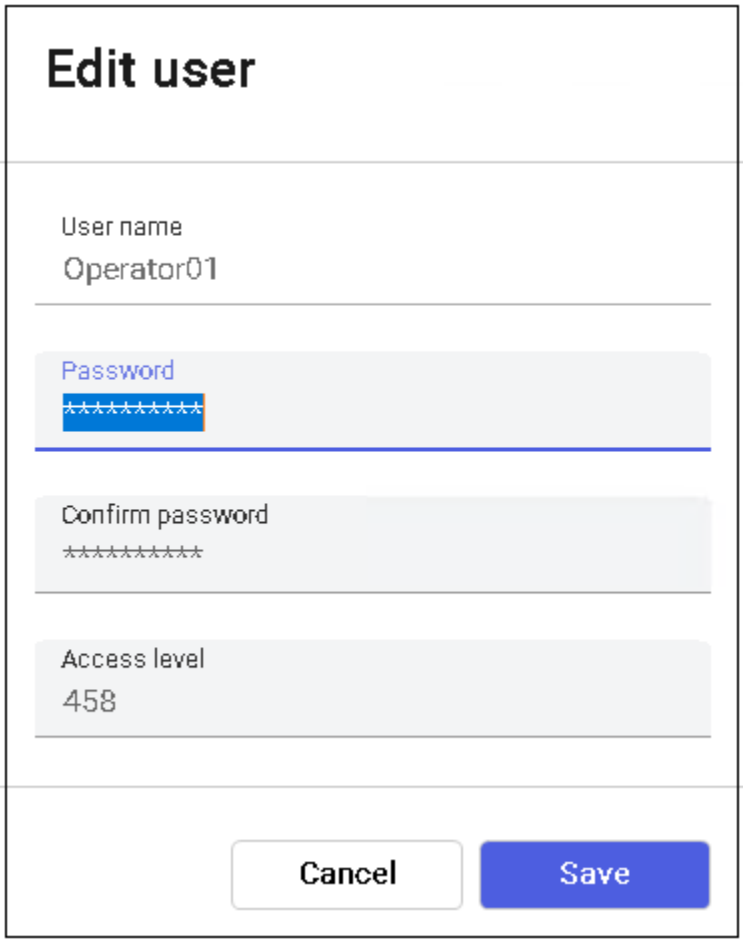
Change an operator password in WindowMaker

1. On the **File** menu, select **Configure**, and then select **Security.**

   The **Security** configuration screen appears.
2. Select the relevant Security type of the user.

   You may be asked to Sign-in. This is to ensure that only administrators or user accounts with access level greater than 9000 can change the password.
3. Enter your User Name and Password.
4. From the **Configure users** list, select the user account that needs password change, and select **Edit**.

   The **Edit user** screen appears.

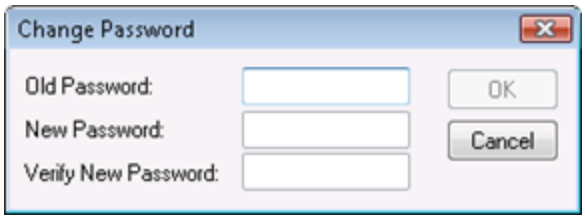


5. In the **Password** field, enter the password.
6. In the **Confirm password** field, enter the password again to confirm.
7. Select **Save**.

Change an operator password in WindowViewer

1. Launch WindowViewer.
2. On the **Special** menu, select **Security**, and then select **Change Password**.

   The **Change Password** dialog box appears.



3. Configure the password. Do the following:
   - In the **Old Password** box, type the old password.
   - In the **New Password** box, type the new password.
   - In the **Verify Password** box, type the new password again.
4. Select **OK**.

Change an operator password using Discrete Button

If you do not plan on showing the **Special** menu in WindowViewer, you can create a discrete button and link it to the $ChangePassword internal tag. When the value of $ChangePassword tag is set to 1, the **Change Password** dialog box appears. Operators can then change their passwords. When the operator closes the dialog box, the system resets the $ChangePassword value to 0. This is a system discrete tag intended for write operation only.

Set up Operating System-based authentication and authorization

Operating system-based security authenticates InTouch users from a list of authorized Windows user groups. You create Windows user groups either on the local computer or an Active Directory server. You must associate Windows users to groups by adding them to specific groups. For more information on creating user groups, see your Windows operating system documentation.

You then assign InTouch access levels to the Windows groups using the AddPermission() function in a script. The AddPermission() function is typically called on application start-up so WindowViewer recognizes all the authorized user groups when a user is ready to log on.

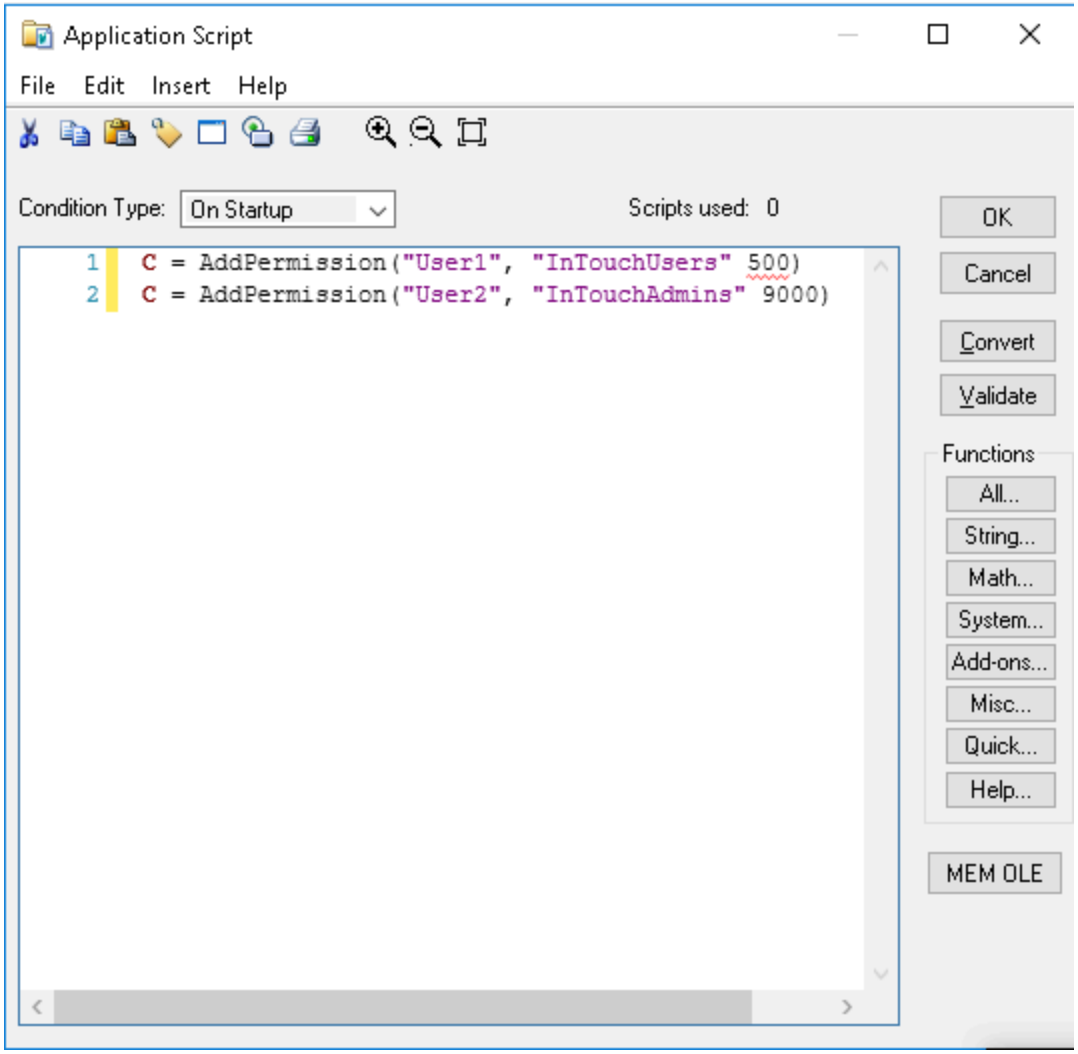
You typically specify operating system-based security immediately after you create an InTouch application.

After you configure the InTouch application to use the operating system authentication and internal InTouch authorization, the Change Password, LogOn, Configure Users and LogOff commands on the Special...Security menu are unavailable.

Set operating system-based security and configure access levels

1. On the **Scripts** pane, select **Application**.

   The **Application Script** dialog box appears.



2. In the **Condition Type** list, select **On Startup**.
3. Use the AddPermission() function to specify the group names and corresponding access levels. The arguments for AddPermission() are operating system (or domain), group name, and access level.
4. Select **OK**.

Set up ArchestrA-based security

The ArchestrA security system is a global function that applies to every object in the Galaxy database. It is a relationship-based system between users and the objects and functions of the Galaxy. This system is based on security roles (configuration, system administration, and run-time permissions) and security groups, which determine a particular security role's run-time permissions on an object-level basis. Configuration of the security system is done in the Integrated Development Environment (IDE) and applied to every object through its own editor.

After you configure the InTouch application to use ArchestrA authentication, the **Change Password**, **LogOn**, **Configure Users** and **LogOff** commands are unavailable in WindowMaker.

Set ArchestrA-based security

1. Open a window in WindowMaker.

2. On the **File** menu, select **Configure**, and then select **Security**.

   The Security configuration screen appears.

3. From the Security type options, select **ArchestrA**.

AddPermission() function

Assigns a certain InTouch access level to a given user group on the local system or on the domain. When a user belonging to that group logs on to the InTouch HMI after the AddPermission() function is called, he or she receives the specified access level.

Category

security

Syntax

```
DiscreteTag=AddPermission( "Domain", "Group", AccessLevel);
```

Arguments

Domain

Name of the domain or local computer in which the group is located.

Group

Windows user group.

AccessLevel

InTouch access level that you want to associate with the given group.

Remarks

Valid for operating system security only. When this function is called, it checks for the presence of the specified group in the specified domain or workgroup. If successful, TRUE is returned, and the specified Access Level is associated with the group for subsequent user log ons. In all other cases, (that is, if an invalid value is specified for any of the arguments) FALSE is returned.

This function is typically configured to run on application startup. It does not affect users that are currently logged on. Only users that log on after AddPermission() is successfully called receive the access level associated with their group.

Examples

```
DiscreteTag=AddPermission( "corporate_hq", "InTouchAdmins", 9000);
DiscreteTag=AddPermission( "johns01", "InTouchUsers", 5000);
```

Operations Control connected experience

The AddPermission() method accepts only two parameters in Operations Control connected experience:

- AVEVA Connect Group
- Access Level

Script function for AddPermission() in Operations Control connected experience:

```
DiscreteTag=AddPermission("", "AVEVA Connect group", AccessLevel);
```

If a runtime user is a member of multiple groups from CONNECT, the access level will be determined by the group with the highest access level.

See also

AttemptInvisibleLogon() function
InvisibleVerifyCredentials() function
IsAssignedRole() function
PostLogonDialog() function
QueryGroupMembership() function

ChangePassword() function

Shows the **Change Password** dialog box, allowing the logged on operator to change his/her password.

Category

security

Syntax

```
[Result=]ChangePassword();
```

Return Value

Returns one of the following integer values:

0 = Cancel was pressed.

1 = OK was pressed.

Remarks

If the operator uses a touch screen, the operator can use the alphanumeric keyboard to enter the new password.

Example

The following script can be placed on a button or called from a condition script or data change script.

```
Errmsg=ChangePassword();
```

$AccessLevel system tag

Defines the access level of the currently logged-in user.

Category

security

Usage

$AccessLevel

Remarks

The value for this tag is determined by the access level assigned to the currently logged-in user's security profile within the InTouch HMI. This profile can be accessed using the **Configure Users** menu in WindowViewer.

The actual numeric value of $AccessLevel has no meaning to WindowViewer, except that a value of 9000 or greater indicates administrative privileges and enables the **Security** menu within WindowViewer. Use the $AccessLevel system tag to further customize security within the system.

Data Type

Integer (read only)

Valid Values

0 through 9999

Example(s)

The following statement is used for the visibility link to make an object, such as a button, visible based on the logged on user's access level:

```
$AccessLevel >= 2000;
{Objects can have a "disable" link associated with them, with the expression based on
$AccessLevel.}
$AccessLevel < 5411;
IF $AccessLevel <=500 THEN
Show "Access Denied"; {popup window denying access}
ELSE
Show "Access Granted"; {popup window granting access}
ENDIF;
```

See Also

$Operator, $OperatorEntered, $PasswordEntered; $ConfigureUsers

$ChangePassword system tag

Shows the **Change Password** dialog box.

Category

security

Usage

```
$ChangePassword
```

Remarks

Set this value to 1 to show the **Change Password** dialog box. The value of the $ChangePassword system tag resets to 0 when the dialog box closes. If you set this system tag to a value other than 1, the results are undefined.

Data Type

Discrete (write only)

Valid Values

1

Example(s)

You can create a discrete push button that opens the **Change Password** dialog box. Assign a single discrete push button link, with the Set option selected, to the push button. When the button is pressed, the $ChangePassword system tag is set to 1, and the **Change Password** dialog box opens.

See Also

$AccessLevel, $OperatorEntered, $PasswordEntered, $Operator, $ConfigureUsers

$ConfigureUsers system tag

Shows the **Configure Users** dialog box.

Category

security

Usage

```
$ConfigureUsers
```

Remarks

This function only works with InTouch security.

Set the value to 1 to open **Configure Users** dialog box.

The value of this system tag resets to 0 when the dialog box closes. If you set this system tag to a value other than 1, the results are undefined.

The user must have an $AccessLevel of >9000 to show this dialog box.

Data Type

Discrete (write only)

Valid Values

1

Example(s)

You can create a discrete push button that opens the **Configure Users** dialog box. Assign a single discrete push button link, with the Set option selected, to the push button. When the button is pressed, the $ConfigureUsers system tag is set to 1, and the **Configure Users** dialog box opens.

See Also

$Operator, $OperatorEntered, $ChangePassword, $PasswordEntered, $AccessLevel

Log On and Off

Logging on to and logging off from an InTouch application varies by the type of security used to protect an application.

Log on to an InTouch-secured application

If the logon information is entered incorrectly or is invalid, a message indicates the log on attempt failed.

If the log on is successful, the $AccessLevel system tag is set to the predefined value associated with the user in the InTouch security user list.

Note: You can also show the **Log On** dialog box using the PostLogonDialog() function. For more information, see PostLogonDialog() function.

Log on to an application

1. On the **File** menu, select **Configure**, and then select **Security**.
2. Select the relevant Security type.

   A logon dialog appears.
3. In the **Name** box, type your user name.
4. In the **Password** box, type your password.
5. Select **Sign in**.

Log on to an Operating System-secured application

When a user logs on to an InTouch application, a dialog box appears requiring the following:

- User name
- Password
- Domain or local computer name

The domain/user name combination is passed to the operating system to authenticate the user's credentials. An attempt is made to log on with or without enabling the operating system cache. If the user cannot be logged on without the cache (due to a network outage, for example), but the user was previously authenticated with the cache enabled, then the user's full name and access level is obtained from the local InTouch cache.

If all of the security checks are cleared successfully, the user is considered to be logged on to the InTouch HMI and the relevant data structures (for example, $Operator) are updated. Otherwise, an error message is shown.

If the operator has never logged on successfully before and the domain is unavailable, the logon attempt fails. The InTouch HMI logs a system event to the error log.

If the password is expired, an error message is shown. After the operator selects **OK**, the **Change Expired Password** dialog box appears, so that the operator can change the password and attempt to log on again with the new password.

Log on to an ArchestrA-secured application

Users typically log on and log off from an ArchestrA-secured InTouch application by entering a valid user name and password.

If your InTouch application has been configured for the ArchestrA security "None", the log on credentials of the default user are used and the operator is not prompted to log on. The following procedure assumes your system has been configured for ArchestrA authentication modes, such as "Galaxy", "OS User based", or "OS Group based".

Log on to an ArchestrA-secured application

1. Start the ArchestrA-secured InTouch application. A log in dialog box appears.
2. Type a valid user name and password. If the system cannot authenticate you, you are prompted again to log on.

After the system authenticates your logon credentials, access to all future operations is granted based on your associated roles/permissions in the security model.

Log off from an InTouch application

Operators log off from an InTouch application after completing their work. You can also configure an application to automatically log off an operator after a specified amount of time has elapsed without any activity by the operator. For more information, see Configure an inactivity time-out.

Log off from an application

1. On the **File** menu, select **Credentials**.
2. In the InTouch HMI WindowMaker User section, select **Sign-out**.

Note: The **Credentials** option is not available if you have selected **Operations Control connected experience** in **Licensing mode** tab of the Configurator.

Create a custom logon window

If the **Special** menu is not shown in WindowViewer, you can create a custom logon window for operator to log on to the application.

Create a custom log on window

- Link the $OperatorEntered, $PasswordEntered and $OperatorDomainEntered system tags to user input objects or use them in a script to set the user name, password, and domain name. These tags are internal message type tags that are intended for write operation only.

The $OperatorDomainEntered tag is required only if the security mode is operating system-based. Otherwise, this tag is ignored. If the security mode is operating system-based and the $OperatorDomainEntered is null, it is treated as pointing to the local computer.

When a value is written to the $PasswordEntered system tag, a logon attempt occurs using the $OperatorEntered, $PasswordEntered, and $OperatorDomainEntered system tag values. No logon occurs if values are written to only the $OperatorEntered or $OperatorDomainEntered system tags.

If the entries are valid, the $AccessLevel and $Operator internal tags are set to their predefined values (configured in the security user list).

You can also link a User Input - Discrete button to the $ConfigureUsers tag to allow an authorized operator with an access level of equal to or greater than 9000 to access the **Configure Users** dialog box to edit the security user name list. When the operator selects the button, the value of the $ConfigureUsers tag is set to 1 and the **Configure Users** dialog box appears. When the operator closes the dialog box, the system resets the value to 0. (This is a system discrete tag intended for write operation only.)

Note: The $ConfigureUsers tag only works if the security type is set to InTouch. It does not work for ArchestrA-based security.

PostLogonDialog() function

Shows the InTouch **Logon** dialog box and returns TRUE.

Category

security

Syntax

```
DiscreteTag=PostLogonDialog();
```

Examples

```
DiscreteTag=PostLogonDialog();
```

See Also

InvisibleVerifyCredentials(), AttemptInvisibleLogon(), IsAssignedRole(), QueryGroupMembership(), AddPermission()

LogonCurrentUser() function

Logs on to InTouch with a user account that is currently logged on to the Windows operating system.

- InTouch configured with OS security: the user is logged on to WindowViewer.
- InTouch configured with ArchestrA security: the user must be a member of ArchestrA OS user-based or OS group-based security.
- InTouch configured with ArchestrA OS user-based or OS group-based security and the user account is configured with smart card credentials: user is logged on using the smart card credentials. The user is logged off if the smart card is removed from the reader.

Category

security

Syntax

```
IntegerResult = LogonCurrentUser();
```

Return value

Returns -1 and no change to the values assigned to $Operator, $OperatorName, $OperatorDomain, and $AccessLevel if the logon fails.

Remarks

This function is available only in InTouch scripting, not in ArchestrA client scripting.

Example

```
IntegerResult = LogonCurrentUser();
```

See also

AddPermission() function
AttemptInvisibleLogon() function
InvisibleVerifyCredentials() function
IsAssignedRole() function
PostLogonDialog() function
QueryGroupMembership() function

Logoff() function

Logs the user off from an InTouch application.

Category

security (write only)

Syntax

```
DiscreteTag = LogOff();
```

Remarks

Logs off the currently logged on user and sets the current user status to the default none operator.

Example

```
DiscreteTag = LogOff();
```

See Also

PostLogonDialog(), InvisibleVerifyCredentials(), IsAssignedRole(), AttemptInvisibleLogon(), QueryGroupMembership(), AddPermission()

AttemptInvisibleLogon() function

The AttemptInvisibleLogon() function can be used in a script to log on a user to InTouch using the supplied credentials. The user is not required to enter a password or user ID.

Category

security

Syntax

```
DiscreteTag=AttemptInvisibleLogon( "UserId", "Password", "Domain" );
```

Arguments

UserId

A valid user account name.

Password

Password of the user.

Domain

Name of the local computer, workgroup, or domain to which the user belongs. This argument applies only if the current security type is operating system-based.

Return value

Returns TRUE if authentication is successful. Otherwise, it returns FALSE.

Remarks

An attempt is made to log on to the InTouch HMI using the supplied credentials.

- If the logon attempt succeeds, then TRUE is returned and the $OperatorDomain, $OperatorName, $AccessLevel, and $Operator system tags are updated accordingly.
- If the log on attempt fails, then FALSE is returned, and the currently logged on user (if any) continues to be the current user.

The *Domain* argument is only valid for operating system-based security. If ArchestrA security mode is in use and if ArchestrA security is in turn using operating system-based security, the *UserId* argument should contain the fully qualified user name with domain name or computer name.

Examples

When security is operating system-based:

```
DiscreteTag=AttemptInvisibleLogon("UserId", "Password", "Domain" );
```

When security is either InTouch-based or ArchestrA-based:

```
DiscreteTag=AttemptInvisibleLogon("UserId", "Password", "" );
```

See also

AddPermission() function
InvisibleVerifyCredentials() function
IsAssignedRole() function
PostLogonDialog() function
QueryGroupMembership() function

AttemptInvisibleLogonEx() function

The AttemptInvisibleLogonEx() function can be used in a script to log on a user to InTouch using the credentials stored in the Credential Manager. The user is not required to enter a password or user ID.

Category

security

Syntax

```
DiscreteTag=AttemptInvisibleLogonEx("Credential Name");
```

Arguments

Credential Name

Name of the credential stored in the Credential Manager. For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server.

Return Value

Returns TRUE if authentication is successful. Otherwise, it returns FALSE.

Remarks

An attempt is made to log on to the InTouch HMI using the credentials saved in the Credential Manager. For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server.

- If the logon attempt succeeds, then TRUE is returned and the $OperatorDomain, $OperatorName, $AccessLevel, and $Operator system tags are updated accordingly.
- If the log on attempt fails, then FALSE is returned, and the currently logged on user (if any) continues to be the current user.

The *Domain* argument is only valid for operating system-based security. If ArchestrA security mode is in use and if ArchestrA security is in turn using operating system-based security, the *UserId* argument should contain the

fully qualified user name with domain name or computer name.

Examples

When security is operating system-based:

```
DiscreteTag=AttemptInvisibleLogonEx("TestCredentialName01");
```

See Also

PostLogonDialog(), InvisibleVerifyCredentials(), IsAssignedRole(), QueryGroupMembership(), AddPermission()

$OperatorEntered system tag

Used to enter a valid user name.

Category

security

Usage

```
$OperatorEntered
```

Remarks

You can use this tagname to create a custom log-on window. You can link touch-sensitive input objects and/or QuickScripts to this tag to set the user name for the logon.

Note: When the $OperatorEntered system tag is valid, $AccessLevel and $Operator system tags are set to their pre-defined values.

Data Type

Message (write only)

See Also

$AccessLevel, $Operator, $PasswordEntered, $ChangePassword, $ConfigureUsers

$PasswordEntered system tag

Used to enter a valid password.

Category

security

Usage

```
$PasswordEntered
```

Remarks

The $PasswordEntered system tag always reads as an empty string. Display links tied to this system tag are always blank. Because the tag always returns an empty string, data change scripts never trigger off of this tag. You can use this tagname to create a custom log-on window. You can link touch-sensitive input objects and/or scripts to this tag to set the password for the user.

Note: When the $PasswordEntered is valid, the $AccessLevel and $Operator system tags are set to their pre-defined values.

Data Type

Message (write only)

See Also

$AccessLevel, $Operator, $OperatorEntered, $ChangePassword, $ConfigureUsers

$OperatorDomainEntered system tag

The domain name as entered by the operator.

Category

Security

Remarks

Whenever the $PasswordEntered tag changes, a log on is attempted without showing a dialog box. The log on attempt uses the $*Entered tags as input user name and the string value of $OperatorDomainEntered as the domain name (used only if the current mode is operating system-based security). If the mode is not operating system-based, this tag is ignored.

Data Type

String

Examples

```
$OperatorEntered == "john";
$OperatorDomainEntered == "Corporate_HQ";
$PasswordEntered == "password";
```

See Also

$Operator

Enable and Disable functionality based upon operator or access levels

After you implement security for your application, you can use the $AccessLevel and $Operator security tags on buttons, in animation link expressions, or in QuickScripts to control whether or not the logged on operator is allowed to perform specific application functions.

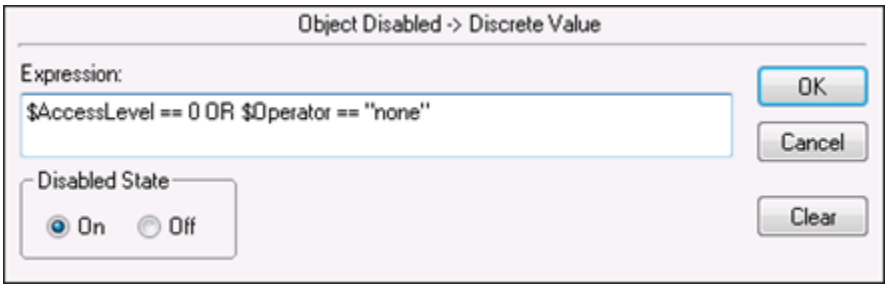
For example, to make an object become visible based on the access level of the logged on user, use the following statement in a visibility animation link expression:

```
$AccessLevel >= 2000;
```

Or, a script can be bounded by an IF statement:

```
IF $Operator == "DayShift" THEN
   Show "Control Panel Window";
   {and other lines that only execute for the DayShift Operator}
ENDIF;
```

You can also control an object's touch functionality based upon the value of an internal security tag by using the Disable animation link. For example:



By using this expression, the object or button is secured from tampering if no one is logged on.

InvisibleVerifyCredentials() function

The InvisibleVerifyCredentials() function can be used in a synchronous QuickScript to verify the credentials of the given user without logging the user on to the InTouch HMI.

Category

security

Syntax

```
AnalogTag=InvisibleVerifyCredentials( "UserId", "Password", "Domain" );
```

Arguments

UserId

Windows operating system user account name that is part of local computer, workgroup, or domain.

Password

Password for the account.

Domain

The Windows domain for the account.

Remarks

If the supplied combination of user, password, and domain are valid then the corresponding access level associated with the user is returned as an integer. Otherwise, -1 is returned.

Note: The InvisibleVerifyCredentials() function must be run from a synchronous QuickScript. The function always returns -1 if run from an asynchronous QuickScript.

This function does not change the currently logged on user. The Domain argument is only valid for operating system-based security. If ArchestrA security is in use and if ArchestrA security is in turn using operating system-based security, the UserId argument should contain the fully qualified user name with domain name or computer name.

Example

```
AnalogTag=InvisibleVerifyCredentials( "john", "Password", "corporate_hq" );
```

See also

AddPermission() function

AttemptInvisibleLogon() function

IsAssignedRole() function

PostLogonDialog() function

QueryGroupMembership() function

InvisibleVerifyCredentialsEx() function

The InvisibleVerifyCredentialsEx() function can be used in a synchronous QuickScript to verify the credentials of the given user without logging the user on to the InTouch HMI..

Category

security

Syntax

```
AnalogTag=InvisibleVerifyCredentialsEx("Credential Name");
```

Arguments

Credential Name

Name of the credential stored in the Credential Manager. For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server.

Remarks

If the supplied combination of user, password, and domain are valid then the corresponding access level associated with the user is returned as an integer. Otherwise, -1 is returned.

Note: The InvisibleVerifyCredentialsEx() function must be run from a synchronous QuickScript. The function always returns -1 if run from an asynchronous QuickScript.

This function does not change the currently logged on user. The Domain argument is only valid for operating system-based security. If ArchestrA security is in use and if ArchestrA security is in turn using operating system-based security, the UserId argument should contain the fully qualified user name with domain name or computer name.

Example

```
AnalogTag=InvisibleVerifyCredentialsEx("john", "Password", "corporate_hq");
```

See Also

PostLogonDialog(), AttemptInvisibleLogon(), IsAssignedRole(), QueryGroupMembership(), AddPermission()

Retrieving information about the currently logged-on operator

Auditing is a primary function of any security system. You can use a set of security system tags to identify the users who logged on to an InTouch application, the domain from which the user logged on, and when the attempt was made.

GetAccountStatus() function

Returns the number of days until the user's password expires.

Category

security

Syntax

```
Result=GetAccountStatus(Domain, UserID);
```

Arguments

Domain

Name of the domain or local computer in which the user account is located.

UserID

Windows user account name that is part of the local computer, workgroup, or domain.

Return value

This function also returns the following values:

Result	Description
-1	Account password expired
-2	Account password never expires
-3	Account locked out
-4	Account disabled
-5	Account info failed

Remarks

Use this script function with operating system-based security. Do not use this function with the ArchestrA security mode.

If the GetAccountStatus() function is used with ArchestrA security, the script attempts to retrieve the account information directly from the domain controller. This works as long as the ArchestrA Galaxy Repository is using operating system security with the same domain.

Example(s)

```
Status = GetAccountStatus("Corporate_HQ","Operator");
```

IsAssignedRole() function

Determines whether the currently logged on user is a member of the specified user role. Only applies to

ArchestrA security.

Category

security

Syntax

```
DiscreteTag=IsAssignedRole( "RoleName" );
```

Arguments

RoleName

The role associated with an Application Server user.

Remarks

Valid for ArchestrA security mode only and applies to the currently logged on user. If a user is currently logged on and has the *RoleName* role assigned in the Galaxy IDE, then TRUE is returned. Otherwise, FALSE is returned.

Example

```
DiscreteTag=IsAssignedRole( "Administrators" );
```

See Also

AttemptInvisibleLogon(), PostLogonDialog(), InvisibleVerifyCredentials(), QueryGroupMembership(), AddPermission()

QueryGroupMembership() function

Determines whether the currently logged on user is a member of the specified user group. Only applies to operating system security.

Category

security

Syntax

```
DiscreteTag=QueryGroupMembership( "Domain", "Group" );
```

Arguments

Domain

Name of the domain or local computer in which the group is located

Group

Name of the group.

Remarks

Valid for operating system security mode only and applies to the currently logged on user. If a user is currently logged on and if he or she is part of the group located on the domain, then TRUE is returned. Otherwise, FALSE is returned.

The QueryGroupMembership() function works with operating system-based security and with ArchestrA security only when the ArchestrA security is set to operating system-based security.

Examples

```
DiscreteTag=QueryGroupMembership( "corporate_hq", "InTouchAdmins" );
DiscreteTag=QueryGroupMembership( "JohnS01", "InTouchUsers" );
```

See Also

PostLogonDialog(), InvisibleVerifyCredentials(), IsAssignedRole(), AttemptInvisibleLogon(), AddPermission()

$OperatorName system tag

Contains the full name of the operator if operating system-based or ArchestrA authentication is used and someone has logged on and has not logged off. Otherwise, the tag contains the name of the user logged on (same contents as the $Operator tag).

Category

Security

Data Type

String (read-only)

Examples

```
IF $OperatorName <> "" THEN
   {Configure some defaults}
ENDIF;
```

See Also

$Operator

$OperatorDomain system tag

Contains a different value depending on the type of security used:

- If operating system-based security is selected and an operator has successfully logged on, the $OperatorDomain tag contains the domain or node name that was specified at log on.
- If ArchestrA security is selected and a user is logged on, the $OperatorDomain contains "ArchestrA."
- If InTouch security is selected, the $OperatorDomain tag contains the string "InTouch".
- If "None" is selected, it is a empty string ("").

Category

Security

Data Type

String

Examples

```
IF $OperatorDomain == "PRODUCTION" THEN
   {Allow change to setpoint}
ELSE
   {Change denied}
ENDIF;
```

See Also

$Operator

$Operator System Tag

Contains the logon name of the user logged on.

Category

Security

Data Type

String

$VerifiedUserName system tag

Contains the verified user's full name if the call to the InvisibleVerifyCredentials() function is successful and if the

security mode is set to operating system-based or ArchestrA Application Server-based security. If the call fails, then the system tag is set to null.

Category

security

Usage

```
$VerifiedUserName
```

Remarks

When the $VerifiedUserName system tag changes (when the InvisibleVerifyCredentials() function is called), an event is generated.

Data Type

Message (read only)

Valid Values

A user's full name.

Example(s)

```
Tag = InvisibleVerifyCredentials( "john","password", "Plant_Floor");{ If the call is
successful, the $VerifiedUserName is set to "John Smith" and an Operator Event is
generated. If the above call is not successful, $VerifiedUserName is set to "".}
```

See Also

InvisibleVerifyCredentials(); $OperatorName, $Operator

Summary of security system tags and functions

The following table shows which security system tags and functions you can use with the different security modes:

	InTouch Security	Operating System Security	ArchestrA Security
$AccessLevel	Yes	Yes	Yes
$ChangePassword	Yes	Yes	Yes

	InTouch Security	Operating System Security	ArchestrA Security
$ConfigureUsers	Yes	No	No
$InactivityTimeout	Yes	Yes	Yes
$InactivityWarning	Yes	Yes	Yes
$Operator	Yes	Yes	Yes
$OperatorDomain	No	Yes	Yes*
$OperatorDomainEntered	No	Yes	Yes*
$OperatorEntered	Yes	Yes	Yes
$OperatorName	Yes	Yes	Yes
$PasswordEntered	Yes	Yes	Yes
$VerifiedUserName	No	Yes	Yes
AddPermission()	No	Yes	No
AttemptInvisibleLogon()	Yes	Yes	Yes
AttemptInvisibleLogonEx()	Yes	Yes	Yes
ChangePassword()	Yes	No	No
EnableDisableKeys()	Yes	Yes	Yes
GetAccountStatus()	No	Yes	Yes*
InvisibleVerifyCredentials()	No	Yes	Yes*
InvisibleVerifyCredentialsEx()	No	Yes	Yes*
IsAssignedRole()	No	No	Yes
Logoff()	Yes	Yes	Yes
LogonCurrentUser()	No	Yes	Yes*
PostLogonDialog()	Yes	Yes	Yes
QueryGroupMembership()	No	Yes	Yes*

* When ArchestrA security is OS user or group based

Application Manager operations allowed for a non administrator user

The operations allowed for a non-administrator users are a subset of the operations allowed for a administrator. Restricting access to essential operations for non-administrator users will prevent security threats. The following table lists the operations under each tab of the InTouch HMI Application Manager:

InTouch Tab

Operation	**Non-Administrator**
New Application	No
Launch WindowMaker	No
Launch WindowViewer	Yes
DBLoad	No
DBDump	No
Delete Application	No
Rename Application	No
Application Properties	No
Export as Template	No
Import Application	No
OPC UA Server Setting	Yes
Find Applications	Yes
Export for IoT	Yes
Upload to CONNECT	Yes
Configure Users for Edge Device	No
Publish Tag data as AVEVA Insight data source	Yes
Node Properties	
App Development > Start following application in WindowViewer as a Service	No
App Development > Enable Network Application Development	Yes

Operation	Non-Administrator
Resolution	Yes
Memory Settings	No
Performance	No
Refresh	Yes
Views	Yes
CONNECT Login	Yes
Change Thumbnail	Yes
Open Application Folder	Yes

Web Client Tab

Operation	Non-Administrator
Enable Web Client	No
Launch Web Client	Yes
Web Client Settings	
Graphic Refresh Rate	No
Alarm Refresh Rate	No
Web Client Site Name	No
Show Header	No
Enable NavBar	No
Allow Anonymous Access	No
AIM Registration Settings	
Use AIM Server as the authentication server	No
Identity Server	N/A
User Name	No
Password	No

Operation	Non-Administrator
Secure Gateway	No
Allow Industrial Graphics to be embedded in any website	No

Apply role-based security to the application folder

Applying role-based security to application folder ensures that only the authorized users can access the resources within the application folder and edit the application. The application with the role-based security is hereafter, referred to as the Secure Application. The application without the role-based security is referred to as the insecure application. You can secure the application folder even when you import or modify an existing application.

Defining Security User Roles

With the installation of InTouch HMI, the following InTouch user groups are automatically added as local users groups:

- InTouch Developers - Users belonging to this group have full control on the InTouch application root folder. They can edit and manage the application, with Read/Write permissions to the entire application.
- InTouch Operators - Users belonging to this group have limited control on the InTouch application root folder. They can run the application. They have Read-Only access to most file and will need Read/Write permissions to a few files.

You can limit access to InTouch application to users in 'InTouchDevelopers' and 'InTouchOperators' groups only.

Assigning Domain Groups

You can assign a domain group to the 'InTouchDevelopers' or 'InTouchOperators' group. The users belonging to the domain group will have the Developer or Operator level access.

Assigning a local group to the 'InTouchDevelopers' or 'InTouchOperators' group is not supported. It is recommended to add the users to the "InTouchDevelopers" or "InTouchOperators" group.

Enable security for the InTouch application folder

You can enable security for the InTouch Application Folder by one of the following ways.

Option 1: Enable Application Folder Security via Application Manager

1. Launch the Application Manager as an Administrator.
2. In the **Tools** menu, on the **Tools** tab, select **Security**.

   The **Security** screen appears.
3. Select the checkbox **Limit access to Standalone InTouch application to users in 'InTouchDevelopers' and 'InTouchOperators' groups only**.

Option 2: Enable Application Folder Security via Configuration.ini File

1. Locate the Configuration.ini file (C:\ProgramData\Wonderware\InTouch).
2. Open the file using a text editor such as Notepad.
3. To enable security, set the value of SecureApplicationFolder to 1.
   To disable security, set the value of SecureApplicationFolder to 0.

Add users and groups

1. Launch the Application Manager as an Administrator.
2. In the **Tools** menu, on the **Tools** tab, select **Security**.

   The **Security** screen appears.
3. Select the **Add** (+) icon.

   The **Select Users, Computers or Groups** dialog appears.
4. Enter the user or group name, and Select **OK**.

   Note: You can search for users and groups by object name, type, or location. Select **Check Name** to ensure the selected object resolves to the AD name or group.

   The entered user or group appears in the Users and Groups grid.
5. To assign the user/group as an InTouch Developer, select the checkbox in the **InTouchDevelopers** column.
6. To assign the user/group as an InTouch Operator, select the checkbox in the **InTouchOperators** column.
7. Select **Save**.

Delete users and groups:

1. In the **Users and Groups** grid, select the entry to be deleted.
2. Select the **Delete** (-) icon.

   The selected user/group is deleted.

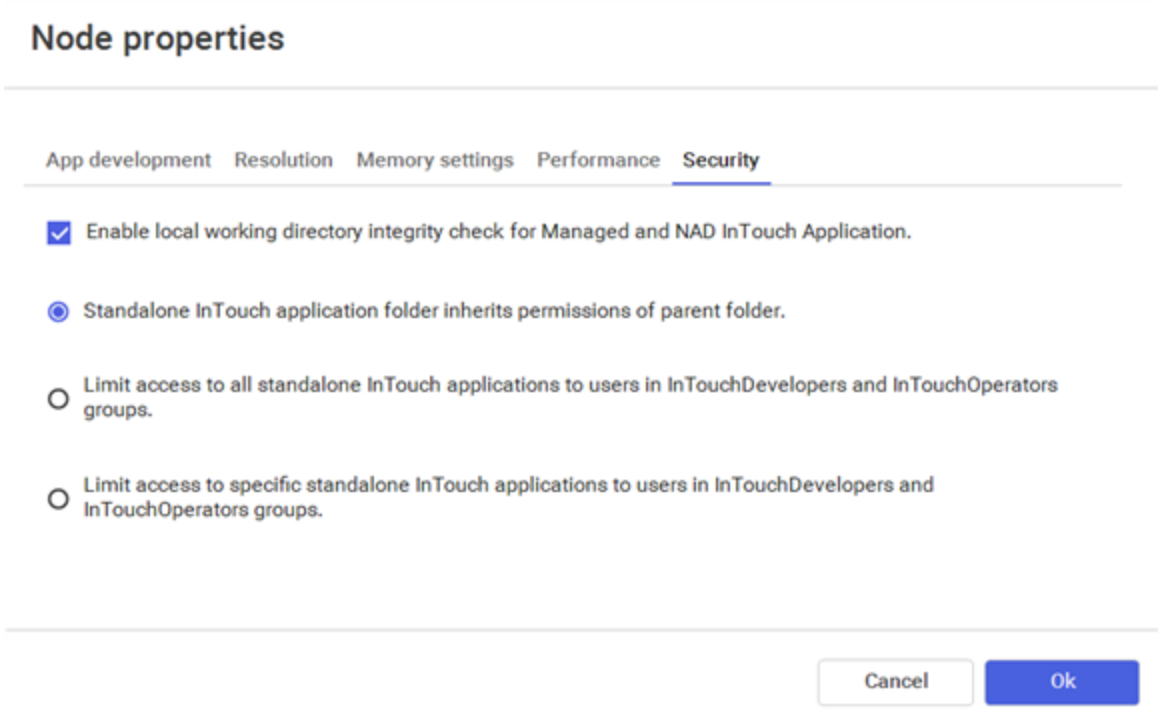
Local working directory integrity check for Managed and NAD InTouch applications

This feature allows to compare deployed application and the application in the local working directory. When launching WindowViewer for a managed and NAD application, a file time stamp and file content hash comparison is performed between the deployed application and the application in the local working directory. The comparison is limited to specific file types including .dlls, .exe, .vedef. If the files do not match WindowViewer will copy the deployed application to the local working directory and run the application.

To enable the integrity check

1. In the Application Manager, go to **Tools** > **Node properties** > **Security**.
2. Select the **Enable local working directory integrity check for Managed and NAD InTouch Applications** check box.

   Note: This option is independent of the **Limit access to Standalone InTouch applications to users in InTouchDevelopers and InTouchOperators groups** option. You may enable them both, or independently, on the same node.
3. Select **OK**.



Note: The copying process may cause delay in staring the WindowViewer. If Network Application Development (NAD) is enabled on a node, the security mode options will be disabled.

Behavior in Managed InTouch application

After comparison if the files do not match, then it indicates that either:

- There is a new version of the application that has been deployed.
- The contents of application files in the local working directory have been modified.

In both scenarios, WindowViewer will copy the deployed application to the local working directory and run the application. This will cause a delay in startup time.

The application's InTouch.ini file has two additional settings:

1. Name: NewDefaultLocalWorkingDirectory
   - For new applications in 2023R2 or later the value is 1.
   - For migrated applications in 2023R2 or later the value is 0.
2. Name: VIEWEDMANAGEDDIALOG

   When you navigate to **File** > **Configure** > **WindowViewer** > **Managed application**, the value is set to 1. This is because the notification that the local working directory has been updated is only shown once.

Notification:

For new managed applications, the notification states that the default local working directory has been updated to %LOCALAPPDATA%\ArchestrA\ManagedApp.

For migrated managed applications, the notification states the recommended local working directory is %LOCALAPPDATA%\ArchestrA\ManagedApp.

To restore the default working directory:

In WindowViewer of a managed application, go to **Files** > **Configure** > **WindowViewer** > **Managed application** and select the **Restore Default** present at the bottom of the page.

The Local working directory field will be updated to:

%LOCALAPPDATA%\ArchestrA\ManagedApp

Behavior in NAD InTouch application

After comparison if the files do not match, then it indicates that either:

- The NAD master has been updated.
- The contents of application files in the local working directory have been modified.

In both scenarios, WindowViewer will copy the NAD master application to local working directory and run the application.

If NAD was previously enabled on a node prior to SP 2023 R2, and then upgraded to SP 2023 R2 or later, then the local working directory path is not modified. A notification is displayed recommending the new file path.

The default Local working directory path for NAD on new installations of System Platform 2023 R2 or later is:

C:\Users*USER*\AppData\Local\NAD

Manage security for InTouch HMI

This section describes different security guidelines and security configuration for InTouch HMI.

General considerations for security

Before you review the information in this section, it is recommended that you go through the following checklist to ensure you plan to cover the security areas that apply to your ICS and organization.

Security Area	Reference Section
Physical and virtual access to the host	General guidelines for securing the host
Latest Windows patches applied	Windows updates
Protecting the host from viruses and malware	Scanning the Host
Access to content on the host	Protecting the applications and content on the host
Securing your network	Secure the network
Configuring services and ports	Manage network services and ports
Securing client/server communication	Secure communication between the client and server
User and group management	Secure systems through authentication and

Security Area	Reference Section
	authorization
Planning for emergencies	Contingency planning

For a list of security feature help topics refer to the table at the end of this section.

Introduction

This appendix provides a general overview on how to securely deploy your AVEVA software product as an Industrial Control Systems (ICS) application.

This appendix is not meant to be comprehensive, and it does not provide any detailed instructions. It is only a collection of basic concepts and recommendations that you can use as a checklist to secure your own systems. If you need help with a specific item in this guide, see the official documentation for that item -- for example, if you need help with your anti-virus software, see the documentation for that software.

AVEVA's approach to securing site networks and ICS software is driven by the following principles:

- View security from both Management and Technical perspectives
- Ensure that security is addressed from both IT and ICS perspectives.
- Design and develop multiple network, system and software security layers.
- Ensure industry, regulatory and international standards are taken into account.
- Aim to prevent security breaches, supported by detection and mitigation.

These principles are realized by implementing the following security recommendations:

- Prevent security breaches using the following components:
  - Firewalls
  - Network-based intrusion prevention/detection
  - Host-based intrusion prevention/detection
- Segregate IT and Plant networks
- Include a clearly defined and clearly communicated change management policy. For example, firewall configuration changes.

  Note: AVEVA strongly recommends following the guidelines prescribed by the U.S. Department of Commerce for securing ICS software. The document "Guide to Industrial Control Systems (ICS) Security" [NIST Special Publication 800-82 Revision 2] (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf) provides detailed information about ICS, typical system topologies, security threats and vulnerabilities, and recommendations for implementing security measures.

Secure the host

Given the sensitive nature of industrial control, it is important to secure not only the ICS software, but also:

- the host on which it runs

- the network to which it is connected
- the hardware used for the ICS software.

Note: The "host" is the Windows computer or Windows Embedded device on which your ICS software is installed and running.

There are several factors to consider for securing the host including:

- Access to the host
- Keeping track of and applying the latest Windows updates
- Keeping the host computer free of viruses and malware
- Protecting the applications and content on the host

Each of these factors is covered in the sections below.

General guidelines for securing the host

Here are a few guidelines to secure the host:

- Use an account with administrative privileges to install the ICS software, and one without administrative privileges to run the ICS software.
- Restrict configuration of ICS to a limited set of users.
- Consider running the ICS software as a Windows service, if that option is available. If the ICS software is run as a service, run it as a low privileged virtual service account.
- Once the host is fully configured and placed in its permanent location, restrict physical access and remote access to it so that only authorized personnel (for example, system administrators, application engineers, run-time operators) can use it.
- Consider disabling or removing physical ports (for example, USB, memory card) that might be used to connect external storage devices and then transfer data.

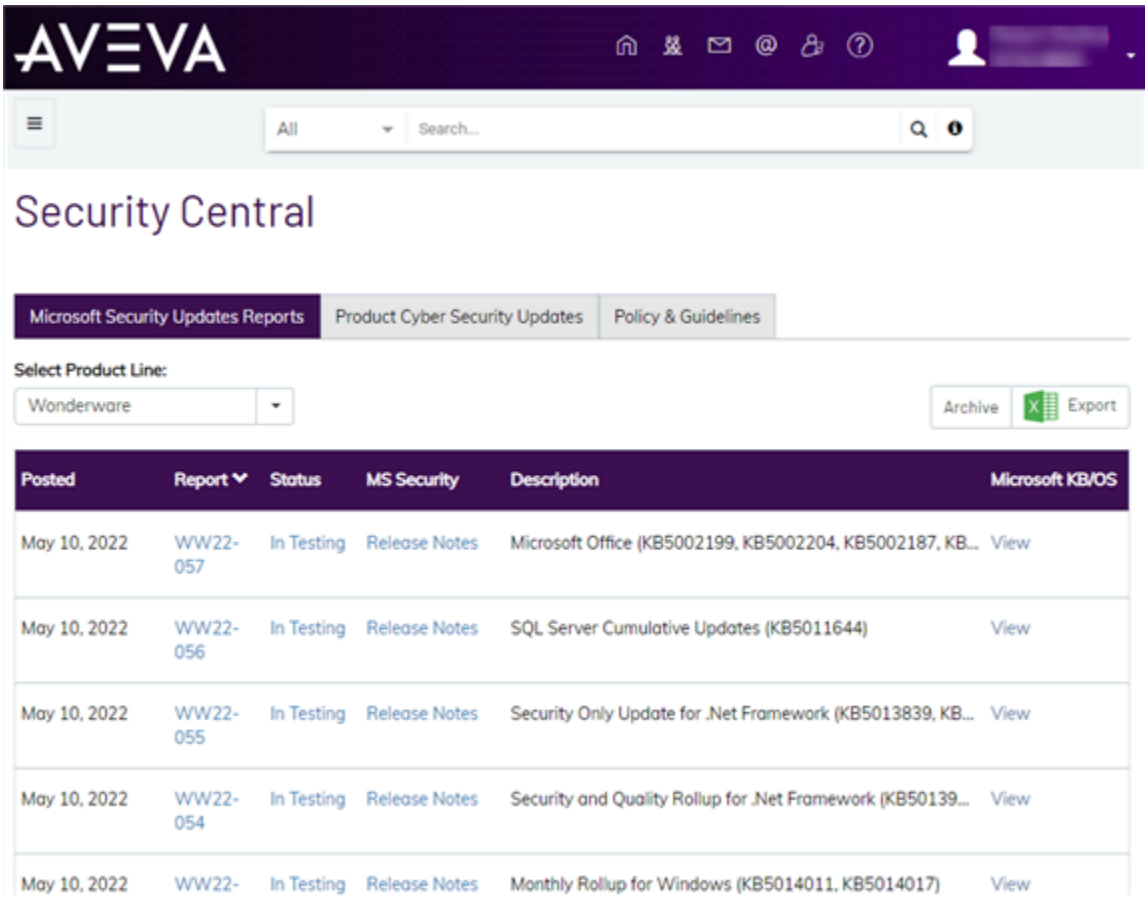
Windows updates

Check that the Windows operating system on the host is a version that is under what Microsoft calls "mainstream support", which means Microsoft actively maintains and releases updates for it. Older versions of Windows are under Microsoft "extended support", which means they are not actively maintained and therefore might become vulnerable without notice. For more information about the different versions of Windows and the different levels of support, see the Windows lifecycle fact sheet on the Microsoft Web site..

Automate Microsoft product updates using Microsoft Windows Server Update Services (WSUS), which enables you to manage and distribute updates to computers on your network. For more information about WSUS, see Windows Server Update Services on the Microsoft Web site. If the host does not or will not have a reliable connection to the WSUS server, perhaps because it is located on a private network, you can either develop a procedure to manually apply updates or consider changing the operating system to a Long-Term Servicing Channel (LTSC) version of Windows, which is updated less frequently.

In addition, AVEVA ICS software is tested for compatibility with Microsoft updates the results of which are published on the Security Central site. Security advisories and bulletins are also published on this site.

Note that when Windows updates run in the background, there is the possibility that different software processes can be adversely affected. Therefore, it is important to schedule the updates to run only during planned shutdown periods.



ICS software updates

Check that the ICS software on the host has all the recommended patches and hot fixes installed.

Some AVEVA applications release regular updates, which should be applied as soon as possible as they may contain security-related fixes.

Note: AVEVA's Global Customer Support (GCS) group publishes a Technology Matrix (https://gcsresource.aveva.com/TechnologyMatrix/Home/Index) for AVEVA software products. This matrix lists the Windows operating system versions against which a software product has been tested for compatibility. In addition, it lists compatible runtime, browser, and virtualization environments for the software. It also includes a list of other products that can be installed on the same computer and lists other products with which this software can communicate.

Scanning the Host

Use both anti-virus and anti-malware software and file integrity checking software to regularly scan the host.

Windows includes Windows Defender by default, but you may choose to install and use additional software that scans for more types of malware or performs other functions. If you do that, make sure the software is provided

by a reputable company. And, as with the operating system, if the host does not or will not have reliable access to the software's update service, develop a procedure to manually apply updates. If you develop a manual update procedure, it should account for all devices on a network or at a site, because a single outdated device can leave the entire network or site vulnerable.

Protecting the applications and content on the host

To protect the applications and content on the host:

- Enable Windows Firewall, and configure it to close all ports that are not used by the ICS software. For more information about port usage, see Manage network services and ports.
- Disable Windows features like remote desktop and file sharing, and remove unnecessary programs like games and social media.
- Restrict access to the files, databases, registry and other resources on the host.
- Use Windows BitLocker to encrypt the hard drive of computers that are either mobile or not located in a secure facility. However, BitLocker may impact the performance of computers.
- Consider using server-class storage (SANs) infrastructure to avoid storing sensitive data on mobile devices.
- If your application stores data in SQL Server, Windows authentication can provide better application security than SQL Authentication. If you switch from Windows Authentication to SQL Authentication, a pop up dialog will appear recommending that you use Windows Authentication for this reason. If you choose to ignore this warning and proceed with SQL Authentication, click **OK**. A similar message will be logged in the OCMC (SMC) Log Viewer.

AVEVA leverages the security built into the Windows operating system to store and manage encryption keys. The encryption keys are stored in a local storage location called the encryption store. For more information about the Windows encryption store, refer to the Microsoft documentation, located at Certificate Stores (https://docs.microsoft.com/en-us/windows-hardware/drivers/install/certificate-stores).

Phases of Data Protection

Data exists in three different phases, and protection must be provided for each phase:

- At rest
- In transit
- In use

Data at rest

Data at rest is data that is not currently being used or accessed, such as data stored on a hard drive, laptop, flash drive, RAID array, network attached storage (NAS), storage area network (SAN), or archived/stored in some other way. Data protection at rest aims to secure inactive data stored on any device or network. For protecting data at rest, you can simply encrypt sensitive files prior to storing them and/or choose to encrypt the storage drive itself. BitLocker Drive Encryption, which you can invoke via the Windows Control Panel, can be used to invoke whole-drive encryption.

In the context of SCADA and ICS systems, data at rest includes stored configuration data, historical data, backups,

and other static data. The duration of storage, that is, long term or short term, does not impact this classification of data at rest. Protection for data at rest is applicable for as long as the data exists in this condition; it is not a fixed condition.

Proper authorization rights need to be set in place to ensure the data is not being viewed by unauthorised users. Other steps can also help, such as storing individual data elements in separate locations, such as a corporate-approved offline backup to decrease the likelihood of attackers gaining enough information to commit fraud or other crimes. Offline backups are the best mitigation against the threat of ransomware.

Data in transit

Data in transit, or data in motion, is data that is actively moving from one location to another.

In the context of SCADA and ICS systems, this encompasses deploying a project to a run-time node, transmitting process variables, VTQ data, and other data that is sent between nodes in a running, production system. This includes alerts and alarms.

Data protection in transit is the protection of this data while the data traveling, including the following examples:

- From node to node within a network
- From network to network
- Accessed via internet
- Transferred from a local storage device to a cloud storage device

Wherever data is moving, effective data protection measures for in-transit data are critical as data is often considered less secure while in motion. Best security practice is to ensure TLS 1.2 encryption is used for all communications using the HTTPS protocol.

Data in use

Data in use refers to data that is being processed or accessed either locally or remotely. This generally involves placing data into memory (RAM) for access and processing by applications and users, potentially multiple users across different computers, mobile devices, remote terminals or other device. Data in use is particularly vulnerable to attack. To protect data in use, encryption, user authentication, and identity management is highly recommended.

In the context of SCADA and ICS systems, data in use can apply to databases, such as those used actively by a historian or deployed to a run-time node. This needs to be safeguarded by a secure transfer channel.

Configure encryption in SQL server

We recommend you enable encrypted connections for SQL Server. You enable encrypted connections for an instance of the SQL Server Database Engine and use SQL Server Configuration Manager to specify a certificate. The server computer must have a certificate provisioned. To provision the certificate on the server computer, you import it into Windows. The client machine must be set up to trust the certificate's root authority.

SQL Server can use Transport Layer Security (TLS) to encrypt data that is transmitted across a network between an instance of SQL Server and a client application. The TLS encryption is performed within the protocol layer and is available to all supported SQL Server clients.

Enabling TLS encryption increases the security of data transmitted across networks between instances of SQL

Server and applications. However, when all traffic between SQL Server and a client application is encrypted using TLS, the following additional processing is required:

- An extra network round trip is required at connect time.
- Packets sent from the application to the instance of SQL Server must be encrypted by the client TLS stack and decrypted by the server TLS stack.
- Packets sent from the instance of SQL Server to the application must be encrypted by the server TLS stack and decrypted by the client TLS stack.

Certificate requirements

For SQL Server to load a TLS certificate, the certificate must meet the following conditions:

- The certificate must be in either the local computer certificate store or the current user certificate store.
- The SQL Server Service Account must have the necessary permission to access the TLS certificate.
- The current system time must be after the **Valid from** property of the certificate and before the **Valid to** property of the certificate.

Install on a server

With SQL Server 2019 (15.x), certificate management is integrated into the SQL Server Configuration Manager. SQL Server Configuration Manager for SQL Server 2019 (15.x) can be used with earlier versions of SQL Server.

If using SQL Server 2012 (11.x) through SQL Server 2017 (14.x), and SQL Server Configuration Manager for SQL Server 2019 (15.x) is not available, follow these steps:

1. On the **Start** menu, click **Run**, and in the **Open** box, type **MMC** and click **OK**.
2. In the MMC console, on the **File** menu, click **Add/Remove Snap-in**.
3. In the **Add/Remove Snap-in** dialog box, click **Add**.
4. In the **Add Standalone Snap-in** dialog box, click **Certificates**, click **Add**.
5. In the **Certificates snap-in** dialog box, click **Computer account**, and then click **Finish**.
6. In the **Add Standalone Snap-in** dialog box, click **Close**.
7. In the **Add/Remove Snap-in** dialog box, click **OK**.
8. In the **Certificates** snap-in, expand **Certificates**, and then expand **Personal**.
9. Right-click **Certificates**, point to **All Tasks**, and then click **Import**.
10. Right-click the imported certificate, point to **All Tasks** and then click **Manage Private Keys**. In the **Security** dialog box, add read permission for the user account used by the SQL Server service account.
11. Complete the **Certificate Import Wizard**, to add a certificate to the computer, and close the MMC console. For more information about adding a certificate to a computer, see your Windows documentation.

Export server certificate

To export the server certificate:

1. From the **Certificates** snap-in, locate the certificate in the **Certificates / Personal** folder.

2. Right-click the **Certificate**, point to **All Tasks**, and then click **Export**.
3. Complete the **Certificate Export Wizard**, storing the certificate file in a convenient location.

Configure server

Configure the server to force encrypted connections. The SQL Server service account must have read permissions on the certificate used to force encryption on the SQL Server. For a non-privileged service account, read permissions will need to be added to the certificate. Failure to do so can cause the SQL Server service restart to fail.

To configure the server:

1. In **SQL Server Configuration Manager**, expand **SQL Server Network Configuration**, right-click **Protocols for <server instance>**, and then select **Properties**.
2. In the **Protocols for <instance name> Properties** dialog box, on the **Certificate** tab, select the desired certificate from the drop-down for the **Certificate** box, and then click **OK**.
3. On the **Flags** tab, in the **ForceEncryption** box, select **Yes**, and then click **OK** to close the dialog box.
4. Restart the SQL Server service.

Configure client

Configure the client to request encrypted connections.

1. Copy either the original certificate or the exported certificate file to the client computer.
2. On the client computer, use the **Certificates** snap-in to install either the root certificate or the exported certificate file.
3. Using SQL Server Configuration Manager, right-click **SQL Server Native Client Configuration**, and then click **Properties**.
4. On the Flags page, in the **Force protocol encryption** box, click **Yes**.

Note: The sections in this topic have been derived from the Microsoft documentation. For more information, refer to the topic 'Enable encrypted connections to the Database Engine' in Microsoft Documentation.

Secure the network

Usually, the host computer will have some sort of network access; it is increasingly rare for an ICS device to run as an entirely standalone device. The host may use the network to communicate with other ICS components such as controllers, sensors, databases, remote clients, and even other hosts in peer-to-peer relationships. You may also use the network to manage several ICS devices from a development or supervisory computer.

Once you determine that the host will have network access, decide how it will connect to the network. In recent years there has been a shift from wired networks (that is, "Ethernet") to wireless networks ("Wi-Fi"), even for business and industrial uses. We recommend against using Wi-Fi for your ICS network because you do not have physical control over who or what might access the network. Any computer or device within range of the Wireless Access Point (WAP) can try to access the network, and even if the network is ostensibly secure, an intruder can intercept and analyze network traffic and potentially discover a vulnerability.

Nevertheless, if you decide to use Wi-Fi for your ICS network, enable all access control features on the WAP

including encryption (for example, WPA/WPA2), a strong password and a list of authorized MAC addresses. Do not try to "hide" the Wi-Fi network by disabling broadcast of the Service Set Identifier (SSID), because doing so actually generates more network traffic that can be intercepted and analysed.

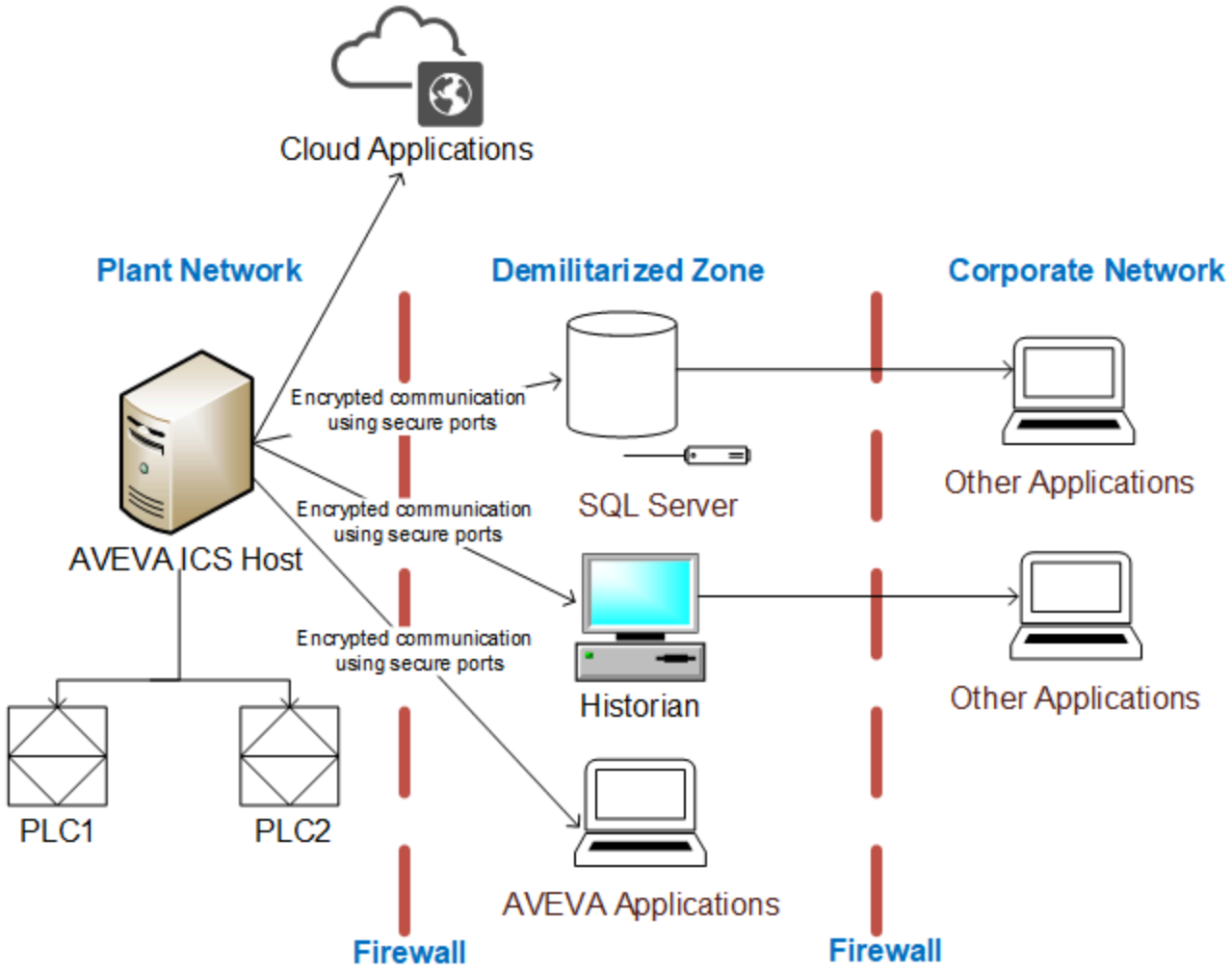
Segment the ICS network

The ICS network itself can be either physically or logically segmented from your other corporate networks. A physically segmented network is by definition the most secure. The network hardware and all computers and devices connected to it form a single closed network with no physical connection to any other network, so an intruder cannot access the network unless they also have access to the physical location.

In contrast, a logically segmented network is physically connected to your other corporate networks and/or the public internet, but it uses various methods to segregate ICS network traffic from other network traffic. This may include:

- Using a unidirectional gateway
- Implementing a Demilitarized Zone (DMZ) network architecture with firewalls to prevent network traffic from passing directly between the corporate and ICS networks
- Having different authentication mechanisms and credentials for users of the corporate and ICS networks.
- The ICS should also use a network topology that has multiple layers, with the most critical communications occurring in the most secure and reliable layer.

Given below is a sample deployment topology.



In no case should your ICS network and devices be directly accessible from the public internet. If there is some part of your ICS that you want to be accessible, (for example, if you want be able to view web-enabled HMI screens from a browser or smart phone), your ICS software should include features that securely pass the necessary traffic between your ICS network and a public-facing server.

Manage network services and ports

A network port is an endpoint of communication in an operating system. While the term is also used for hardware devices, in software it is a logical construct that identifies a specific process or a type of service. In other words, a network port is conceptually different from hardware ports like USB, memory card, and even the wired network connection.

Computers and devices can access many different network services at the same time by communicating on different network ports. Each network service or communication protocol has an associated port number. Some port numbers are specified by international standards, and therefore they are universally recognized. Other port numbers are claimed by proprietary software, and in most cases they can be changed in the software settings if there is a conflict with other software or services.

Firewalls control network traffic by either accepting or refusing communication on these network ports. If a port is open, it accepts communication, and if a port is closed, it refuses communication. Almost every layer of a network -- from the operating system on an individual computer or device, to the router that manages traffic

within a network, to the gateway that manages traffic between networks -- has its own firewall.

The documentation for your ICS software should include a list of network ports that are commonly used by the software. Given the nature of ICS, the list typically includes services like web, email, file transfer, external databases, device drivers, and the ICS software itself for server-client communications. Configure the firewalls to open only those network ports that are actually used by your ICS. Disable all unused services and close all unused ports.

Secure communication between the client and server

Like most server-client applications, your ICS software should support secure communication between the server and client in order to prevent the messages sent between those two stations from being read by any other stations on the same network. Note that this is different from securing the network itself in order to prevent unauthorized access to the network.

This sort of communication is also sometimes known as "Encrypted Channel" because it uses the Transport Layer Security (TLS) standard to encrypt the server-client messages. The latest version of the standard is TLS 1.3 (released August 2018), but it is not yet in common use. The latest version of the standard in common use is TLS 1.2 (released August 2008). TLS supersedes the earlier Secure Sockets Layer (SSL) standard, although SSL is still used in older applications.

Certificates

TLS and SSL use a system of certificates and keys to digitally "sign" the messages sent between the server and client. When the server establishes communication with the client (and vice versa), it presents its certificate which identifies its name, network address, organization, physical location, and so on. The client can then choose to either accept or refuse the certificate as presented. If it accepts the certificate, it agrees to accept messages encrypted with the same certificate, and it uses the associated key to decrypt those messages.

When you configure this sort of communication, you need to choose one of the following:

- Using self-signed certificate
- Using certificates signed by a Public Certificate Authority (CA)
- Using Domain-issued certificates or certificates signed by a Private Certificate Authority using systems like Microsoft Active Directory Certificate Service (AD CS)

A self-signed certificate is issued and signed by the same application that presents it. Self-signed certificates are easy to create and manage, but they are secure only if you control both the server and the client and therefore control which certificates are installed on each.

In contrast, CA-signed certificates are slightly difficult and expensive to acquire, but they are more flexible than self-signed certificates because you do not need to control both the server and the client. If you configure the server to present a CA-signed certificate, the client will accept the certificate because it recognizes the Certificate Authority.

Domain-issued certificates are internal certificates typically managed by your IT department. They are issued and validated by an Active Directory Certificate Authority. Domain-issued certificates are free and can be issued instantly.

You need to renew CA-signed and Domain-issued certificates at regular intervals.

For more information about how to enable Encrypted Channel features and manage self-signed certificates in your ICS software, see the documentation for that software. However, acquiring a CA-signed certificate and then using it to sign other certificates is typically beyond the scope of ICS software documentation.

Note: Encrypted and unencrypted communications typically use different network ports.

Cloud-based systems

It is possible that your ICS software might access cloud-based solutions, or might itself be hosted on the Cloud. It is important to mitigate the risks associated with cloud-based access and hosting.

Access cloud-based solutions

Several AVEVA applications are now being made available through the Cloud, and ICS software may need to connect to these applications. One of the main risks associated with accessing cloud-based applications is unauthorized access. Connecting ICS software to Cloud solutions must be done in a secure manner, and needs to use secure protocols such as Transport Layer Security (TLS).

It is important that data integrity is maintained at all times. Use data classification to identify data that is sensitive and data that can be made public. Secure computers, storage and networking in order to secure the data that is stored and transmitted. Work with your Cloud Service Provider (CSP) to configure users, assign access levels and monitor and control access. Ensure that the CSP's buildings are physically secure and protected from unauthorized access.

Cloud-based ICS software

While hosting ICS software on the Cloud provides several benefits such as flexibility, scalability and availability, it is also fraught with security risks such as susceptibility to hacking resulting in damage to the organization's reputation . Therefore, it is important to implement a security strategy before you make your ICS software accessible on the Cloud. For securing ICS software on the Cloud, you need to consider the following:

- Securing access points by putting in place authentication, monitoring and support mechanisms.
- Implementing cloud-based, centralized security measures including encrypting communications using TLS.

Note: It is recommended that you review the NIST Cybersecurity Framework (https://www.nist.gov/cyberframework) for additional information.

Secure systems through authentication and authorization

Typically, ICS software is comprised of a large number of systems, each accessed by a variety of users including engineers, operators and managers. The level of access that each type of user requires is different. So, it is necessary to manage user authentication and authorization to secure the system.

Authentication

Authentication is the process of verifying a user's/system's identity. Authentication can be managed in the following ways:

- Within the ICS software through application accounts
- Through Windows accounts, which can be local to a single computer
- Through Authentication systems (see the next section for details)

While ICS software allows for user and role management, it can become cumbersome and complicated to manage a large number of user accounts as employees and roles change. Because of this, use of Windows accounts is generally preferred.

Authentication systems

Authentication systems such as Active Directory and Lightweight Directory Access Protocol (LDAP), referred to as authentication servers, are a repository of and provide centralized management for all system accounts and individual user accounts. An authentication protocol is used for all communication between authentication servers and the user or server requesting authentication.

Even though use of authentication systems provides improved scalability, the following factors must be considered depending upon the size and complexity of your operations:

- It is important that the authentication servers are highly secured.
- The authentication server system creates a single system for managing all system accounts. Therefore, it requires to be available at all times. To ensure minimal disruption during an emergency, redundancy must be considered.
- Permit caching of user credentials only for users who have authenticated their identity recently.
- Networks that support the authentication protocol must be reliable and secure to assist in trouble-free authentication.

It may also be worthwhile implementing two-factor authentication using additional applications such as PingID.

Authorization

Authorization is the process of providing the correct level of privileges to users by applying access rules to authenticated users, systems (HMIs, field devices and SCADA servers) and networks (remote sites' LANs).

Manage users and groups through windows

When you configure security, you may choose to do one of the following:

- Keep the configuration local to a single application.
- Share the configuration between multiple applications.
- Manage the configuration as part of the network domain (for example, using Active Directory). This option typically allows users to have the same user account for the network, the host, and the ICS software. Using Active Directory gives you the following advantages:
  - A centralized repository for user and group data, enabling effective implementation of security policies and procedures.
  - Provides a single point of access to all network resources after the user is identified and authenticated.

To manage users and groups:

- First define a specific role for each group, and then configure the group privileges to fit that role.
- Groups may overlap, but it is often better to have clearly separate groups and then assign individual users to multiple groups, if necessary.
- Set or change the password for the ICS software's default user (e.g., "guest").
- Define stringent password policies to force users to create strong passwords. Enforce mandatory password updates on a regular basis.

Manage users and groups through ICS software

Your ICS software should have a built-in security system that controls who may use the software and what privileges they have.

Users should be assigned permissions that determine what each user is authorized to do within the ICS system. Permissions can be managed either on a per-account basis or on a group basis by making use of roles. Group or role-based access control is preferred as it greatly simplifies management. Users can be moved from one role to another as the organization's needs change, and can also be members of multiple roles if required.

Each user should have their own user account with a unique user name and a strong password. The user account can then be assigned to one or more groups.

Accounts should always be assigned the least privileges necessary to perform their functions. Accounts with Windows Administrator permissions should be reduced to the minimum, and typically only used to install and configure the software. Likewise, accounts with SQL Server SysAdmin privileges should be reduced to the minimum, and typically only used to install and configure the software.

In most cases, the ICS software will allow associating Windows Groups with roles within the product. While defining and assigning roles, consider the following:

- Roles should be defined to have the least privileges necessary for their functionality.
- Roles should be limited to a single purpose in order to simplify the permissions assigned to them.
- Users can be members of multiple roles if necessary.

Contingency planning

Incidents are inevitable. It is, therefore, important to develop a strategy to detect an incident quickly and respond to it in a timely manner in order to minimize loss and protect your system. An organization must consider contingencies arising from incidents such as fire, flood and so on, and those arising from failure of hardware or software components. Cyber attacks such as ransomware are becoming more common and must also be considered.

An organization should have contingency plans in place to cover the entire range of failures and eventualities. Employees should be trained and be familiar with the contents of the contingency plans.

As part of planning for contingencies, it is important to establish a site, physically separated from the central one, that has replication capability. Doing so will ensure the integrity of an operational system where the central site is at risk from fire, floods or other disasters. The replication capability includes having duplicated hardware, and requires software configuration and key state information to be periodically propagated from the central site to the recovery site. Each recovery scenario is unique, so it is important to consult with system integration experts regarding the design of communications equipment, hardware and the configuration of the software.

Protecting the data stored in your system is also of paramount importance. Full and incremental backups must be scheduled on a regular basis. Backups should be verified by running tests to restore from backed up data. Backups should be stored offline so that they are safe from cyber attacks such as ransomware.

Organizations should also have business continuity and disaster recovery plans that are similar to contingency plans. These plans are covered briefly in the sections to follow.

Audit and log

As part of implementing security for ICS software, it is important to incorporate auditing and logging activities on various systems and networks.

Auditing and logging provide information on the current state of your ICS, and help to ensure that the system is functioning as expected. If an incident occurs, you can use the activity logs to trace the origin of the incident to a computer, user or network. Auditing and logging can also help with troubleshooting issues.

If you are connecting to cloud-based solutions, audit all virtual machines (VMs) to ensure data integrity.

Business continuity planning

Business continuity planning addresses strategies to maintain or re-establish production in the event of any disruption. These disruptions may be caused by a natural disaster (flood, earthquake, etc), by an intentional or unintentional human-made event (arson, operator error, power outage, etc), or by system failure.

Depending upon the duration of the potential ICS application outage caused by a disruption, operational recovery plans for short-term outages and disaster recovery plans for long-term outages must be formulated. It is also important to employ physical security for areas of a production site that house data acquisition and control systems that might have higher-level risks. Your business continuity plan should specify system and data recovery procedures for your systems. Once the recovery procedures are documented, a schedule should be developed to test the recovery procedures. Particular attention must be paid to the verification of backups of system configuration data and product or production data. The procedures should be reviewed periodically.

If you are accessing cloud-based solutions, ensure that systems are available at all times. In case of a disaster, services should switch to a new physical location to provide continued service.

Disaster recovery planning

A disaster recovery plan (DRP) is a set of procedures to protect and recover an IT infrastructure in case of a disaster. It contains the procedures to follow before, during and after a disaster. Disasters can be natural, environmental or human-made (intentional or unintentional).

A DRP is essential for continued availability of the ICS, and should cover the following:

- When the DRP should be activated depending upon an event, its duration and its severity.
- Detailed course of action for operating the ICS manually until external connections are secured.
- Personnel responsible for each procedure.
- Processes for securely backing up data and restoring it. This should cover:
  - Requirements for building redundancy.
  - File backup procedures.
  - Frequency of backups.
  - Storage mechanism for full and incremental backups.
  - Safe storage of installation media, license keys and configuration information.
  - List of individuals responsible for performing, testing, maintaining and restoring backups.
- List of personnel with physical and virtual access to the ICS.
- Detailed configuration information about the components of the ICS.

- Schedule for testing the DRP.

Conclusion

Security lapses present a serious threat to ICS software and infrastructure. Therefore, it is important for every organization to:

- Be proactive about preventing security lapses
- Identify potential lapses
- Detect them in a timely manner when they occur
- Address lapses to ensure minimum disruption and maximum availability

To this end:

- Computers and networks must be secured
- Users and groups must be authenticated and authorized
- Contingency plans must be in place to recover from untoward or intentional events

Refer to the document "Guide to Industrial Control Systems (ICS) Security" [NIST Special Publication 800-82 Revision 2](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf) for additional details and recommendations.

Security configuration for InTouch HMI

The table below lists the security areas that you need to configure for InTouch HMI and the details of the section(s) in this guide that provide the corresponding instructions.

Security Area(s)	**Topic(s) in this guide**	**Summary**
Protecting the Applications and Content on the Host	Configure an inactivity time-out	Configure WindowViewer to automatically log off an inactive operator from an InTouch application. Inactivity time-out is not supported when connected experience is enabled.
Securing Systems through Authentication and Authorization	Authentication and Authorization Based Security	Users needs to authenticate themselves before using an InTouch application. InTouch verifies whether the authenticated user is authorized to use specific functionality.

Security Area(s)	Topic(s) in this guide	Summary
	Using Virtual Accounts	Using virtual accounts provides an additional layer of security when accessing alarm functions. See Using Virtual Accounts in the InTouch HMI Alarms and Events Guide.
Protecting the Applications and Content on the Host	Locking System Keys	Restrict operator access to standard Windows functions by disabling system keys on the computer running an InTouch application.
Securing Systems through Authentication and Authorization		
Managing Users and Groups through ICS Software	Using InTouch-Based Security	Restrict which functions an operator is allowed to perform by linking those functions to internal tags.
Managing Users and Groups through Windows	Using Operating System-Based Security	Inherit some user/group account policies from the Windows operating system.
Securing the application data folder	Applying Role-based Security to the Application Folder	Users can define user roles and user groups to manage access to the application data folder.

AVEVA Group plc
High Cross
Madingley Road
Cambridge
CB3 0HB
UK

Tel +44 (0)1223 556655

www.aveva.com
To find your local AVEVA office, visit **www.aveva.com/offices**