



Wonderware
InTouch® HMI
Application
Management and
Extension Guide

All rights reserved. No part of this documentation shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Invensys Systems, Inc. No copyright or patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this documentation, the publisher and the author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

The information in this documentation is subject to change without notice and does not represent a commitment on the part of Invensys Systems, Inc. The software described in this documentation is furnished under a license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of these agreements.

© 2010, 2014 by Invensys Systems, Inc. All rights reserved.

Invensys is a Schneider Electric company.

Invensys Systems, Inc.
26561 Rancho Parkway South
Lake Forest, CA 92630 U.S.A.
(949) 727-3200

<http://www.wonderware.com>

For comments or suggestions about the product documentation, send an e-mail message to ProductDocumentationComments@invensys.com.

All terms mentioned in this documentation that are known to be trademarks or service marks have been appropriately capitalized. Invensys Systems, Inc. cannot attest to the accuracy of this information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

Alarm Logger, ActiveFactory, ArchestrA, Avantis, DBDump, DBLoad, DT Analyst, Factelligence, FactoryFocus, FactoryOffice, FactorySuite, FactorySuite A², InBatch, InControl, IndustrialRAD, IndustrialSQL Server, InTouch, MaintenanceSuite, MuniSuite, QI Analyst, SCADAAlarm, SCADASuite, SuiteLink, SuiteVoyager, WindowMaker, WindowViewer, Wonderware, Wonderware Factelligence, and Wonderware Logger are trademarks of Invensys plc, its subsidiaries and affiliates. All other brands may be trademarks of their respective owners.

Contents

Welcome	11
Documentation Conventions	11
Technical Support	12
 Chapter 1 Managing InTouch Applications	13
About the InTouch Application Manager	14
Managed InTouch Applications	15
Starting the Application Manager	16
Starting the ArchestrA IDE from the Application Manager	17
Creating an InTouch Application	18
Opening an Application in WindowMaker and WindowViewer	20
Modifying an InTouch Application	21
Deleting an InTouch Application from the Application Manager	22
Customizing the Application Manager Window	22
Finding InTouch Applications	22
Publishing Applications to Remote Nodes	23
Contents of a Published File	25
Publishing a Stand-alone InTouch Application	26

Migrating Earlier InTouch Applications to the Current Version	29
Converting Legacy Alarm Displays	29
Managing Application Settings for Windows Vista and Newer Operating Systems	30
Chapter 2 Distributing Applications	31
Supported InTouch Architectures	31
Single Computer Architecture	32
Client-Based Architecture	32
Server-Based Architecture	33
Network Application Development (NAD)	34
Planning Considerations for Networked Applications	35
I/O Data Access for Networked Applications	35
Using Global I/O Addresses	35
Using Local I/O Addresses	36
Wonderware SuiteLink	37
Access to Shared Files	37
Using Global Addresses to File Data	38
Using Local Addresses to File Data	38
Access to Shared Files through UNC	39
Logging Data in a Distributed Environment	40
Configuring Remote History Providers	42
Dynamically Configuring Remote History Providers	44
Configuring Distributed Historical Logging	44
Considerations for Special Networks	45
Configuring an InTouch Application for NAD	46
Performing an Automatic NAD Update	48
Performing a Manual NAD Update	49
\$ApplicationChanged System Tag	49
\$ApplicationVersion System Tag	50
RestartWindowViewer() Function	51
ReloadWindowViewer() Function	51
Application Editing Locks	52
Changes to an Application During a NAD Update	52
Scaling the Application Resolution at Run Time	53
Locking the Application Resolution	55
Using Terminal Services	57
Planning Considerations for Terminal Server Applications	58
Deploying InTouch Applications in a Terminal Services Environment	58
Alarms in a Terminal Services Environment	58

Security in a Terminal Services Environment	59
I/O in a Terminal Services Environment	59
Script Execution in a Terminal Services Environment	60
Logging on to a Terminal Session Properly to Run InTouch	60
Alarm Query Syntax in a Terminal Service Environment	60
Miscellaneous Limitations in a Terminal Services Environment	61
Retrieving Information About the InTouch Client Session Using Scripts	62
TseGetClientId() Function	62
TseGetClientNodeName() Function	62
TseQueryRunningOnConsole() Function	63
TseQueryRunningOnClient() Function	63
Chapter 3 Managing InTouch Services	65
Running WindowViewer as a Service	66
Configuring WindowViewer to Start as a Service	66
Manually Starting a Service	67
Stopping a Service	68
Configuring the User Account for InTouch Services	68
Troubleshooting InTouch Services	69
Viewing Error Messages for Services	70
Troubleshooting Problems with the Services User Account	70
Deactivating Advised I/O Items	71
Registry Keys for the InTouch Services	71
Chapter 4 Exporting and Importing Tag Definitions, Windows, and Scripts	73
Exporting Tag Definitions	73
Viewing Exported Tag Definitions	75
Importing Tag Definitions	75
Tagname Dictionary Import File Format	76
Creating an Import File Template	77
Setting the Operating Mode for Dictionary Import Files	79
:MODE=REPLACE	79
:MODE=UPDATE	79
:MODE=ASK	81
:MODE=IGNORE	81
:MODE=TERMINATE	81
:MODE=TEST	82

Setting Access Names and Alarm Groups	82
:IOAccess Keyword Attributes	82
:AlarmGroup Keyword Attributes	84
Defining Tag Type Keywords and Attributes	87
Tag Keyword Attributes	88
:MemoryDisc Keyword Attributes	94
:IODisc Keyword Attributes	95
:MemoryInt Keyword Attributes	96
:IOInt Keyword Attributes	99
:MemoryReal Keyword Attributes	102
:IOReal Keyword Attributes	104
:MemoryMsg Keyword Attributes	107
:IOMsg Keyword Attributes	107
:GroupVar Keyword Attributes	108
:HistoryTrend Keyword Attributes	109
:TagID Keyword Attributes	109
:IndirectDisc Keyword Attributes	110
:IndirectAnalog Keyword Attributes	110
:IndirectMsg Keyword Attributes	111
Using Blank Strings in an Import File	111
Using Default Values for Fields	112
Creating SuperTag Instances	113
Importing Tag Definitions with DBLoad	114
Importing Windows	115
Converting Placeholder Tags for an Imported Window	117
Exporting Windows	118
Converting InTouch Windows to ArchestrA Symbols	120
Preparing to Convert Windows	120
Converting Windows	120
Converting Animation Scripts	121
After Converting Windows	122
Diagnosing Window Conversion Errors	122
Completing the Window Conversion Procedure	123
Importing Scripts	125
Converting Placeholder Tags in an Imported Script	127
Tag Placeholders for Imported Windows and Scripts	128

Chapter 5 Securing InTouch 131

InTouch Security Features	132
Configuring an Inactivity Time-Out	132
\$InactivityTimeout System Tag	134
\$InactivityWarning System Tag	134

Locking System Keys	135
EnableDisableKeys() Function	137
Hiding Menu Items at Run Time	138
Authentication and Authorization Based Security	141
Comparing Authentication and Authorization	141
Different Authentication Security Modes	141
Using InTouch-Based Security	141
Using Operating System-Based Security	142
Using ArchestrA-based Security	143
Using Smart Cards for Authentication	143
Setting up Smart Card Authentication	144
Enabling Smart Card Authentication in WindowMaker	145
Logging on with Your Smart Card	145
Using Secured and Verified Writes	146
Performing a Secured Write	147
Performing a Verified Write	150
Customizing the Secured/Verified Write Dialog Box	152
Working with the SignedWrite() Function at Run Time	153
Managing Users and Setting Their Authorization Levels	153
Configuring InTouch Security Authentication and Authorization	154
Changing an InTouch Operator Password at Run Time	155
Setting Up Operating System-Based Authentication and Authorization	156
Setting Up ArchestrA-Based Security	157
AddPermission() Function	157
ChangePassword() Function	158
\$AccessLevel System Tag	159
\$ChangePassword System Tag	160
\$ConfigureUsers System Tag	161
Logging On and Off	162
Logging on to an InTouch-Secured Application	162
Logging On to an Operating System-Secured Application	162
Logging On to an ArchestrA-Secured Application	163
Logging Off from an InTouch Application	163
Creating a Custom Logon Window	164
PostLogonDialog() Function	164
LogonCurrentUser() Function	165
Logoff() Function	166
AttemptInvisibleLogon() Function	166

\$OperatorEntered System Tag	167
\$PasswordEntered System Tag	168
\$OperatorDomainEntered System Tag	169
Enabling and Disabling Functionality Based Upon	
Operator or Access Levels	169
InvisibleVerifyCredentials() Function	170
Retrieving Information About the Currently Logged-on	
Operator	171
GetAccountStatus() Function	171
IsAssignedRole() Function	172
QueryGroupMembership() Function	173
\$OperatorName System Tag	174
\$OperatorDomain System Tag	174
\$Operator System Tag	175
\$VerifiedUserName System Tag	175
Summary of Security System Tags and Functions	176
 Chapter 6 Switching a Language at Run Time	179
Configuring Languages for Run-time Language	
Switching	180
Changing the Font Settings for a Configured Language	181
Adding Run-Time Language Switching Functionality	182
SwitchDisplayLanguage() Function	185
\$Language System Tag	185
Exporting Application Text for Offline Translation	186
Exporting Text to an Existing Dictionary File	187
Translating an Exported Dictionary File	188
Importing Translated Dictionary Files	190
Exporting Alarm Comments for Translation	191
Understanding Two-Character Application IDs	191
Exporting Alarm Comments	192
Exporting to an Existing Alarm Comment File	193
Editing the Dictionary File	194
Importing Translated Alarm Comments	196
Testing the Language Switching Functionality at	
Run Time	197
Distributing Localized Files to Network Application	
Development Clients	198
 Chapter 7 Viewing Applications at Run Time	199
About WindowViewer	199

Customizing Your Run time Environment	199
Configuring General WindowViewer Properties	200
Configuring Visual Characteristics of WindowViewer	203
Configuring User Access to Applications Running in Remote Sessions	205
About Managing Memory for WindowViewer	206
Configuring Memory Usage for WindowViewer Windows	206
Configuring the Memory Health Check Interval	208
Configuring wwHeap Memory Settings	209
Setting Advanced Formatting Properties	211
Configuring Core Affinity for WindowViewer in a Terminal Server Environment	212
Working with WindowViewer Windows	214
Common Dialog Box Features	214
Opening Windows from WindowViewer	216
Closing Windows from WindowViewer	217
Transferring from WindowViewer to WindowMaker	217
About InTouchView Applications	218
Creating a New InTouchView Application	219
Converting an Application Between InTouch and InTouchView	220
Converting an InTouchView Application to an InTouch Application	220
Converting an InTouch Application to an InTouchView Application	221
InTouchView Licensing	222
 Chapter 8 Setting Up a Multi-Monitor System	223
Multi-Monitor Configurations	224
Single Video Card Configuration	224
Characteristics of a Single Card Configuration	224
Characteristics of Single Card Drivers	225
Multiple Video Card Configuration	225
Characteristics of a Multiple Card Configuration	226
Characteristics of Multiple Card Drivers	226
Planning a Multi-Monitor Application	227
Choosing a Multi-Monitor Video Card	227
Determining the Application Screen Resolution	227
Determining the Number of Monitors to Display the Application	228
Determining the Placement of Application Windows	229
Windows Show in a Forced Location	229

Windows Are Manually Moved	229
Windows Are Placed Automatically Based on Environment	229
Developing a Multi-Monitor InTouch Application	230
Configuring Multi-Monitor Parameters	230
Configuring Screen Resolution Conversion	231
Deploying the Application and Verifying Multi-Monitor Settings	231
Verifying Multi-Monitor Support During Run Time	232
 Chapter 9 Using InTouch on a Tablet PC	233
Annotating and Sending Visualization Screens as E-mail Messages	234
Making Window Annotations	234
Selecting, Copying, and Deleting Window Annotations	235
Saving, Printing, and E-Mailing an Annotated Window	235
AnnotateLayout() Function	236
Changing Screen Orientation	237
 Appendix A Customizing Applications Settings from the INTOUCH.ini File.....	239
Custom INTOUCH.ini Parameters	240
Setting Custom Logging Properties	241
Setting Logging Frequency	241
Logging Remote Referenced Tags	241
Disabling WindowMaker Shortcut Menus	242
Setting Custom WindowViewer Properties	242
Adding a Script Loop Timer	242
Scaling InTouch Windows to Different Screen Resolutions	243
Setting the Length of the Print Waiting Period	243
Logging Alarm Comments	243
Setting the Drawing Mode of a 16-Pen Trend	244
Resizing a Numeric Keypad	244
Resizing the Input Fields of Analog and String User Input Links	245
Resolving Stuck Application Button or Displayed Value Problems	245
 Index.....	247

Welcome

This documentation describes how to manage and extend your InTouch applications. This documentation covers how to create applications, open them in WindowMaker and WindowViewer, move them from one computer to another, and distribute them across a network. You can also extend applications to run in certain environments, such as across multiple monitors and on Tablet PCs.

This documentation assumes you know how to use Microsoft Windows, including navigating menus, moving from application to application, and drawing objects on the screen. If you need help with these tasks, see the Microsoft documentation.

Documentation Conventions

This documentation uses the following conventions:

Convention	Used for
Initial Capitals	Paths and file names.
Bold	Menus, commands, dialog box names, and dialog box options.
Monospace	Code samples and display text.

Technical Support

Wonderware Technical Support offers a variety of support options to answer any questions on Wonderware products and their implementation.

Before you contact Technical Support, refer to the relevant section(s) in this documentation for a possible solution to the problem. If you need to contact technical support for help, have the following information ready:

- The type and version of the operating system you are using.
- Details of how to recreate the problem.
- The exact wording of the error messages you saw.
- Any relevant output listing from the Log Viewer or any other diagnostic applications.
- Details of what you did to try to solve the problem(s) and your results.
- If known, the Wonderware Technical Support case number assigned to your problem, if this is an ongoing problem.

Chapter 1

Managing InTouch Applications

When managing InTouch applications, you:

- Create or delete InTouch applications. See "Creating an InTouch Application" on page 18 and "Deleting an InTouch Application from the Application Manager" on page 22.
- Open applications in either WindowMaker or WindowViewer. See "Opening an Application in WindowMaker and WindowViewer" on page 20.
- Search for applications. See "Finding InTouch Applications" on page 22.
- Move an application to a different computer. See "Publishing Applications to Remote Nodes" on page 23.
- Distribute applications among multiple computers. See "Distributing Applications" on page 31.
- Manage InTouch services. See "Managing InTouch Services" on page 65.
- Import or export tag definitions, windows, and scripts. See "Exporting and Importing Tag Definitions, Windows, and Scripts" on page 73.
- Configure security. See "Securing InTouch" on page 131.

You can extend your application by:

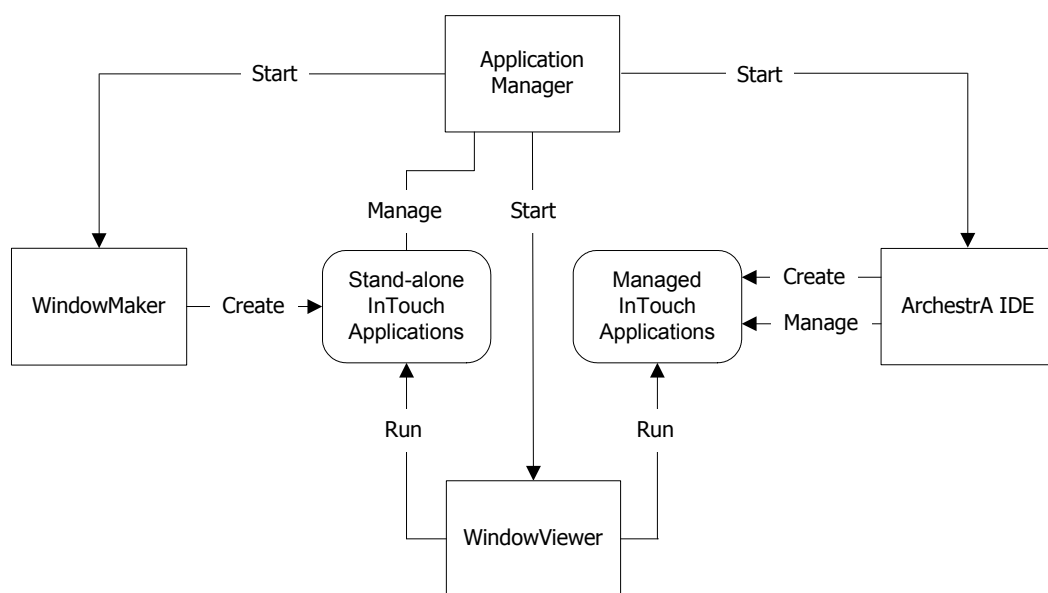
- Translating text strings and alarm comments into different languages. See "Switching a Language at Run Time" on page 179.
- Integrating an application with ArchestrA. See "Managed InTouch Applications" on page 15 and "Viewing Applications at Run Time" on page 199.
- Displaying applications on multiple monitors. See "Setting Up a Multi-Monitor System" on page 223.
- Using applications on a Tablet PC. See "Using InTouch on a Tablet PC" on page 233.

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.

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.

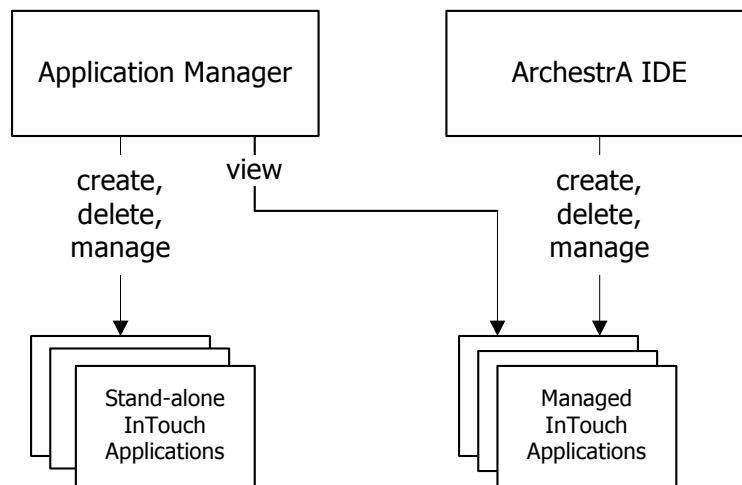
Application Manager shows a list of your current InTouch applications. You select an application from the list to open in WindowMaker or WindowViewer.



Managed InTouch Applications

You can manage InTouch applications using the ArcestraA Integrated Development Environment (IDE) if it is installed on the same computer as the InTouch HMI. These applications are called managed InTouch applications. Unlike stand-alone InTouch applications that are managed entirely by InTouch Application Manager, managed applications are more integrated into the ArcestraA environment and support advanced graphics.

You can start the ArcestraA IDE from the Application Manager. Managed InTouch applications appear in the InTouch Application Manager as “Managed” and can be edited only by starting WindowMaker from within the IDE.



For more information, see "About InTouch HMI and ArcestraA Integration" on page 2 in the *InTouch® HMI and ArcestraA® Integration Guide*

Starting the Application Manager

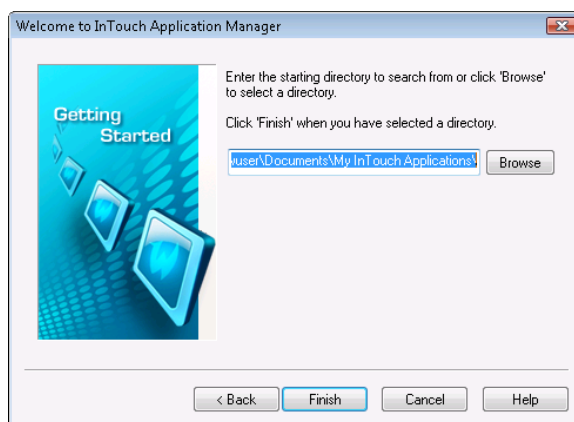
You can start the Application Manager from the **Start** menu or from a shortcut placed on your computer's desktop.

To start the Application Manager for the first time

- 1 On the taskbar, click **Start**, point to **Programs**, point to **Wonderware**, and then click InTouch. The **Welcome to InTouch Application Manager** wizard appears.



- 2 Click **Next**. The next page appears.

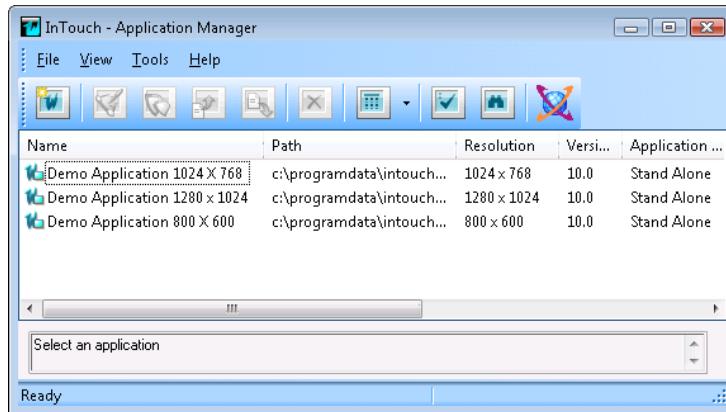


The default application folder is shown. For example, C:\Documents and Settings\UserName\My Documents\My InTouch Applications. For computers running Microsoft Windows Vista, the default application folder is C:\Users\UserName\Documents\My InTouch Applications.

- 3 To select a different folder or create a new folder, click **Browse**, select or create the folder, and then click **OK**.

The InTouch program changes the default folder to the one you specify. This default folder applies for all InTouch users on the computer.

- 4 Click **Finish**. The Application Manager appears.



The window shows InTouch applications on your computer that you created or found using the Application Manager. The **Version** column shows the InTouch version that was last used to save the application.

Starting the ArchestrA IDE from the Application Manager

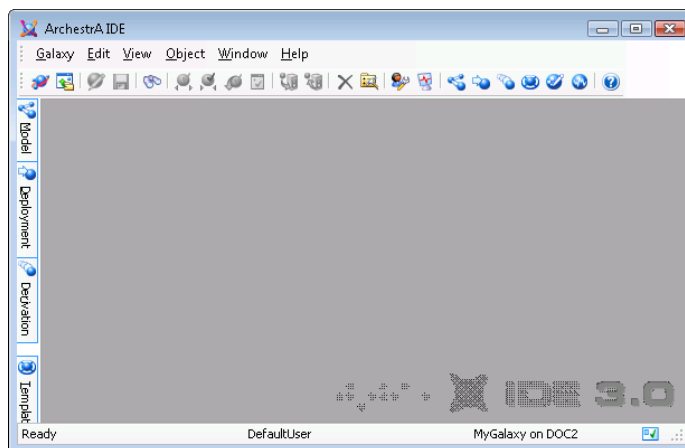
You can switch to the ArchestrA Integrated Development Environment (IDE) to create or edit your managed InTouch application.

Note: Application Server Bootstrap and IDE must be installed on the same computer as the InTouch HMI to start the ArchestrA IDE from the Application Manager.

To start the ArchestrA IDE from the Application Manager

- 1 Start the Application Manager.
- 2 On the **File** menu, click **ArchestrA IDE**. The Application Server **Connect to Galaxy** dialog box appears.

- 3 Connect to an existing Galaxy or create a new Galaxy. The **Archestra IDE** dialog box appears after you connect to a Galaxy.



- 4 Create or edit the objects designated as a managed InTouch application.

Creating 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.

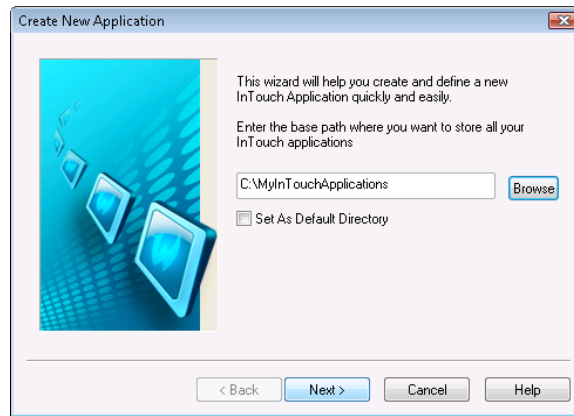
The INTOUCH.ini file is created when you create an application. The INTOUCH.ini file contains the default configuration settings for your application. As you configure your application, the new settings are written to the INTOUCH.ini file.

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.

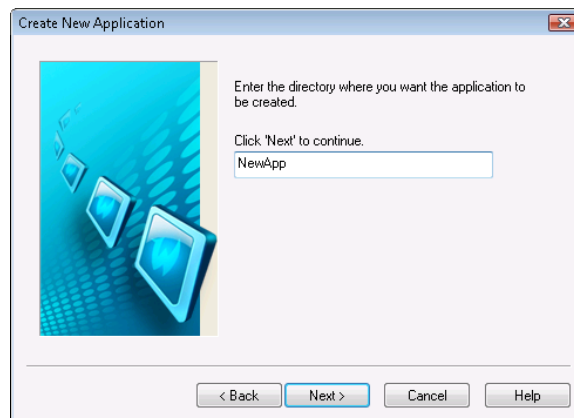
To create a new application



- 1 On the **File** menu, click **New**. The **Create New Application** wizard appears.

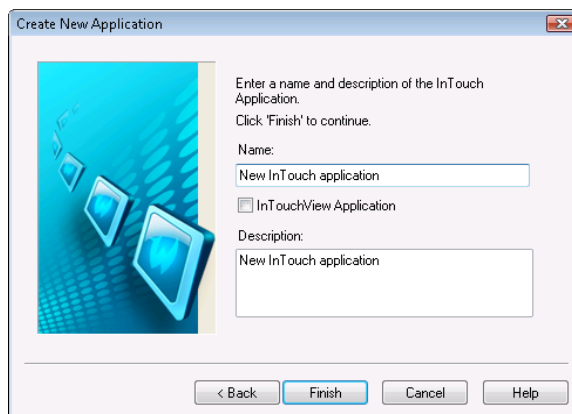


- 2 Configure the application folder. If the folder you specify does not exist, you are prompted to create it.
 - Click **Browse** to specify a folder other than the default.
 - Select the **Set As Default Directory** check box to make the specified folder the default.
- 3 Click **Next**. The next wizard page appears.



- 4 Type the application folder name.

5 Click **Next**. The next wizard page appears.



6 Configure the application details.

- In the **Name** box, type a unique name for the application.
- In the **Description** box, type an optional description up to a maximum of 255 characters.
- Select the InTouchView **Application** check box to create an InTouchView application. For more information about InTouchView, see "Viewing Applications at Run Time" on page 199.

7 Click **Finish**.

Opening an Application in WindowMaker and WindowViewer

You must open a new application in WindowMaker before you can open it in WindowViewer.

To open an application in WindowMaker

1 Select the application in the Application Manager window.



2 On the **File** menu, click WindowMaker.

Tip You can also double-click an application to open it in WindowMaker.

To open an application in WindowViewer

1 Select the application in the Application Manager window.



2 On the **File** menu, click WindowViewer.

Modifying an InTouch Application

You can rename an application and change the application properties.

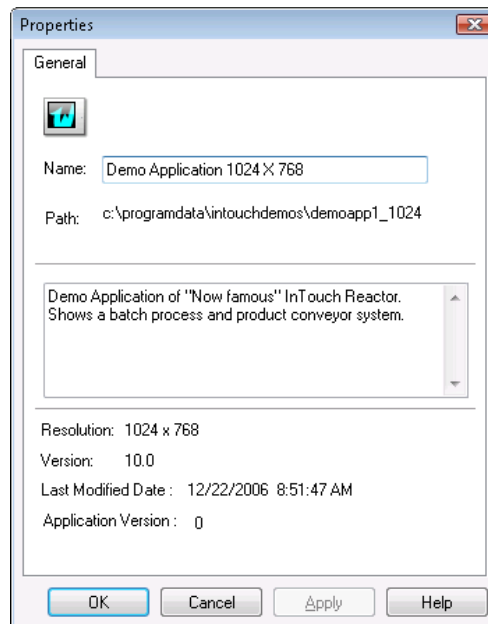
To rename an application

- 1 Select the application in the list.
- 2 On the **File** menu, click **Rename**.

To modify application properties



- 1 Select the application in the list.
- 2 On the **File** menu, click **Properties**. The **Properties** dialog box appears.



- 3 Configure the application properties.
 - In the **Name** box, type a new name for the application.
 - In the window, type another description for the application.
- 4 Click **OK**.

Deleting an InTouch Application from the Application Manager

When you delete an application from the Application Manager, the application files remain on your computer.

To delete an application



- 1 Select an application in the list.
- 2 On the **File** menu, click **Delete**.
- 3 In the message that appears, click **Yes**.

Customizing 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 large icons, small icons, in a brief list, or in a detailed list.

To hide the toolbar

- On the **View** menu, click **Toolbar** so that no check mark is shown.

To hide the status bar

- On the **View** menu, click **Status Bar** so that no check mark is shown.

To change the view for the application list



- On the **View** menu, click the appropriate command.

Finding InTouch Applications

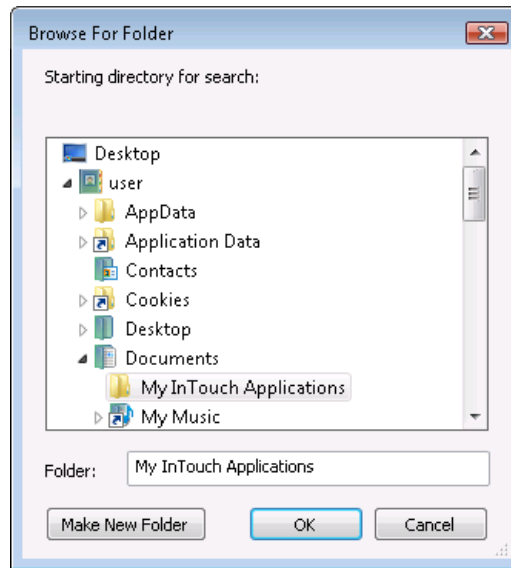
You can search for existing InTouch applications. Application Manager shows any applications that are found.

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.

To find applications



- 1 On the **Tools** menu, click **Find Applications**. The **Browse For Folder** dialog box appears.



- 2 Select the folder in which you want to search for applications.
- 3 Click **OK**.

Publishing 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 stand-alone InTouch applications. You publish managed InTouch applications using the Archestra 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 *.www 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	The appedit.lok file, which indicates that the application is open in WindowMaker.
	retentiv.d	
	retentiv.a	
	retentiv..s (two dots)	
	retentiv.h	
	wm.ini	
	db.ini	
	linkdefs.ini	
	tbox.ini	
	group.def	
	itocx.cfg	Compiled window files with the file name extension .www.
	Any files with names of the form SSD_*.xml.	
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.

Publishing a Stand-alone InTouch Application

Use the Application Publisher to publish a stand-alone 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.

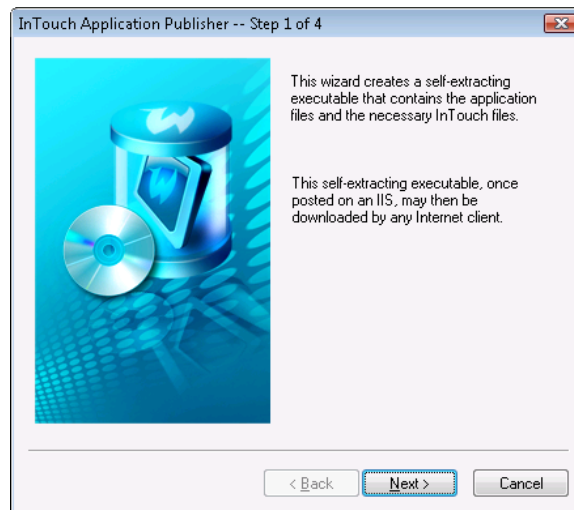
To publish a managed InTouch application, use the Arcestra IDE.

To publish a stand-alone InTouch application

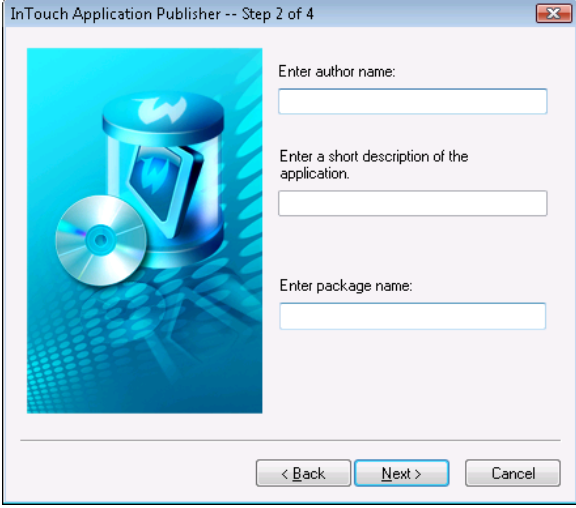
1 Start the Application Publisher.

- a** Open WindowMaker.
- b** Show the Classic View and expand the **Tools** pane.
- c** Expand **Applications**.
- d** Double-click **Application Publisher**.

The InTouch **Application Publisher – Step 1 of 4** dialog box appears.



- 2 Click **Next**. The InTouch **Application Publisher – Step 2 of 4** dialog box appears.

The dialog box is titled "InTouch Application Publisher -- Step 2 of 4". It features a blue graphic on the left showing a CD and a folder. On the right, there are three text input fields with labels: "Enter author name:", "Enter a short description of the application.", and "Enter package name:". At the bottom, there are three buttons: "< Back", "Next >", and "Cancel".

InTouch Application Publisher -- Step 2 of 4

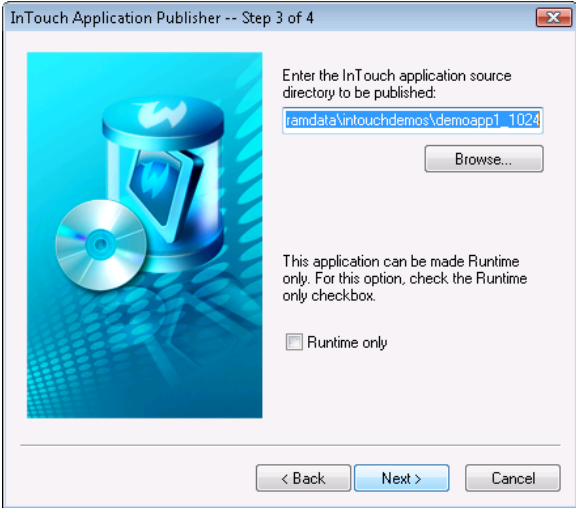
Enter author name:

Enter a short description of the application:

Enter package name:

< Back Next > Cancel

- 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 Click **Next**. The InTouch **Application Publisher – Step 3 of 4** dialog box appears.

The dialog box is titled "InTouch Application Publisher -- Step 3 of 4". It features the same blue graphic as the previous step. On the right, there is a text input field with the label "Enter the InTouch application source directory to be published:". The field contains the text "ramdata\intouchdemos\demoapp1_1024". Below the field is a "Browse..." button. Further down, there is a checkbox labeled "Runtime only" with the text "This application can be made Runtime only. For this option, check the Runtime only checkbox." above it. At the bottom, there are three buttons: "< Back", "Next >", and "Cancel".

InTouch Application Publisher -- Step 3 of 4

Enter the InTouch application source directory to be published:

ramdata\intouchdemos\demoapp1_1024

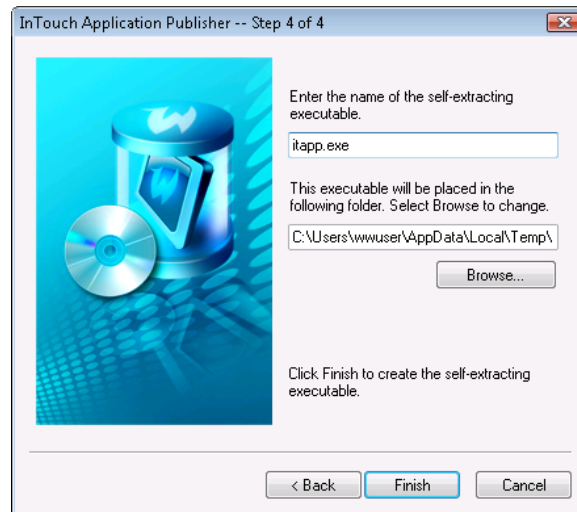
Browse...

This application can be made Runtime only. For this option, check the Runtime only checkbox.

☐ Runtime only

< Back Next > Cancel

- 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 Click **Next**. The InTouch **Application Publisher – Step 4 of 4** dialog box appears.



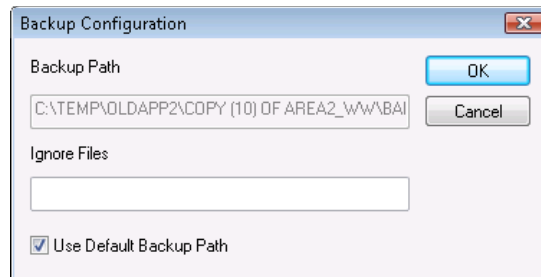
- 7 Configure the details for the executable application package.
 - In the first box, verify the executable name in the first box is correct. By default, the executable name is the same as the package name.
 - In the second box, type the path to the folder in which to save the executable file, or click **Browse** to select a different folder. By default, the executable is saved in your current temporary folder.
- 8 Click **Finish**.

Migrating 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 two dialog boxes:

- The first dialog box confirms that you want to migrate the older application to the current version of the InTouch HMI.
- The second dialog box requests information from you to create a backup copy before migrating the old application to the current version of the InTouch HMI.

You can migrate existing native, managed, or published InTouch applications to the current InTouch version. You see the **Backup Configuration** dialog box after you confirm that you want to migrate an older application to the current version of the InTouch HMI. You must specify the folder to create the backup copy and if you want to exclude any files from the backup.



To change the default backup path (<Application Directory>\Bak), clear **Use Default Backup Path** box. 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.

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 each file name separated by a semicolon (;). Or, use standard wild card characters ("*" and "?") to exclude a set of files by the common characters in their names.

Converting 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.

Managing Application Settings for Windows Vista and Newer Operating Systems

InTouch application settings, such as the application path, are stored in the Win.ini file. The Win.ini file is located in different folders, based on the operating system:

- Operating systems other than Vista and Windows 2008:
C:\Windows
- Terminal Services: C:\Documents and Settings\<User Name>\WINDOWS
- Vista and Windows 2008 operating systems: C:\Users\<User Name>\AppData\Local\Wonderware

On Windows Vista and newer operating systems, 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.

Chapter 2

Distributing 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. 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 ArcestrA IDE. For more information about using the ArcestrA IDE with the InTouch HMI, see About InTouch HMI and ArcestrA Integration in the *InTouch® HMI and ArcestrA® Integration Guide*

Supported InTouch Architectures

Supported InTouch network architectures are:

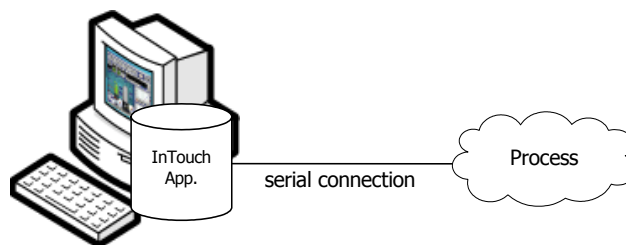
- Single computer
- Client-based
- Server-based
- NAD

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.

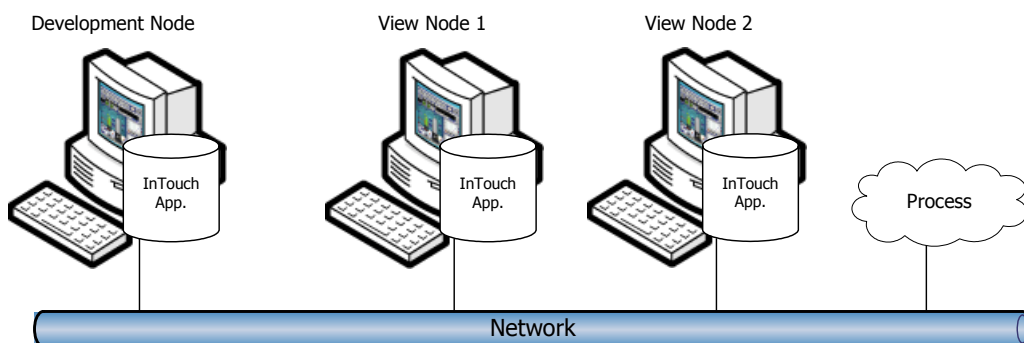
In this architecture, you develop the InTouch application on the single computer. You can copy the application to another computer to modify it and then copy it back to the original computer.

Development and View Node



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.



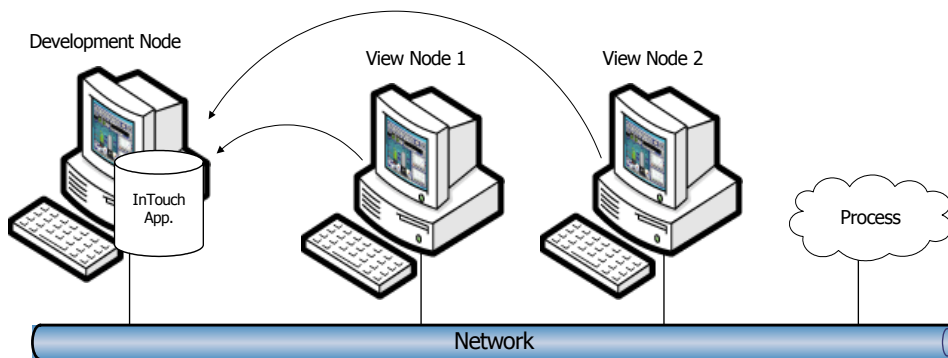
There is inherent redundancy because each node can be self-sufficient, and there is no limit to the number of View nodes you can use.

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 Wonderware Historian. 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

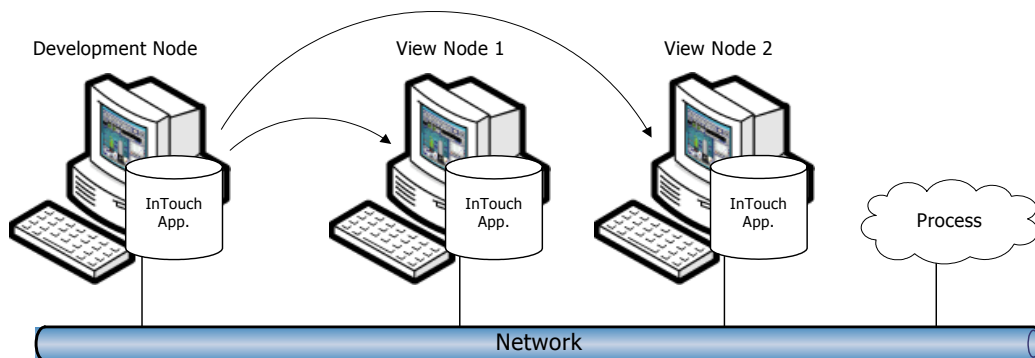
Network Application Development (NAD)

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.

Planning Considerations for 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.

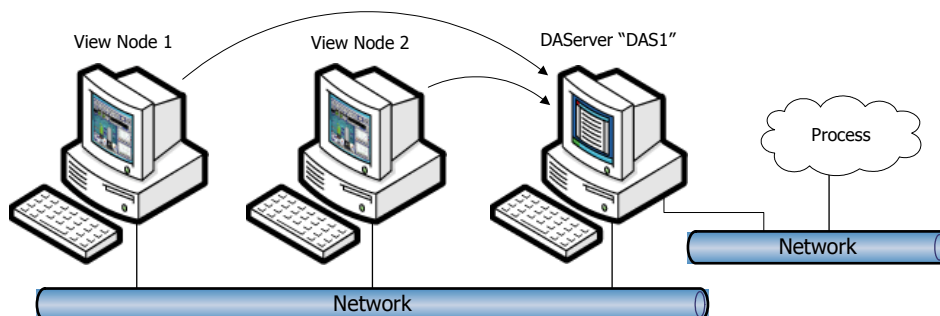
Note: InTouchView is restricted to the single Galaxy Access Name. You cannot create other Access Names for InTouchView. For more information about the restrictions of InTouchView, see "Viewing Applications at Run Time" on page 199.

The View node must have the same access to data sources as the development node.

Using 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.



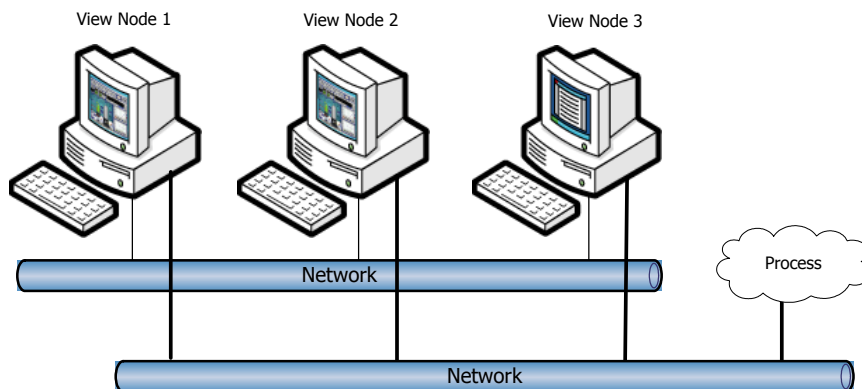
You can set up an InTouch application to identify an element of data stored on another node by using a three-part addressing convention in an Access Name. The Access Name addressing convention includes the node name, application name, and topic name where the remote data is located. An InTouch application 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 the *InTouch® HMI Data Management Guide*

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.

Using 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 the *InTouch® HMI Data Management Guide*

Wonderware SuiteLink

The Wonderware 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:

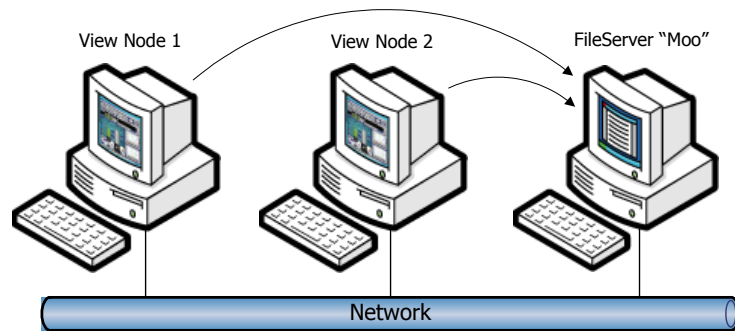
- Global addresses to a network file server.
- Local addresses to local files.

The View node must have the same access to data sources as the development node.

Using 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.



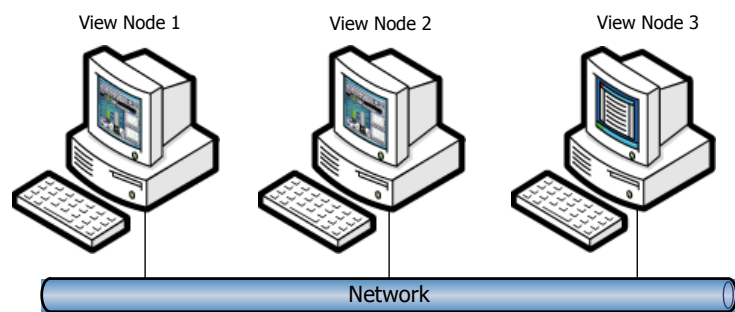
To set up a shared file

- 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");
```

Using 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 Wonderware 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:

```
\\EnginerM\Boiler
```

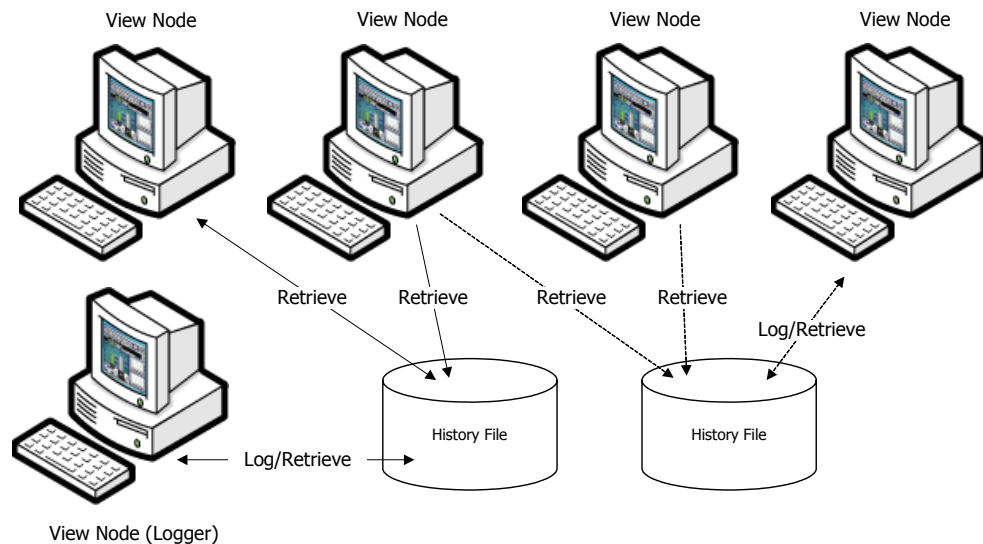
No path is required if the share is a path specified in the PATH environment variable.

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.

Logging 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.

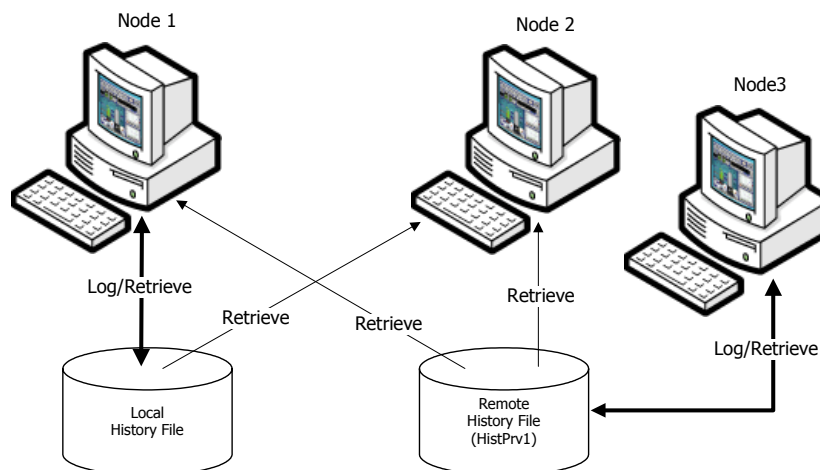
Data for each tag checked for 'Log Data' is automatically written to disk after 22 samples for that tag have been collected. If the `HTUpdateToCurrentTime()` function is called, data is written to disk regardless of the number of samples collected. By default, data is written to disk once an hour. You 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 Wonderware 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.

Do the following major steps to create this type of application:

- 1** Create a history provider list. See "Configuring Remote History Providers" on page 42.
- 2** Create and configure a historical trend object. For more information, see Trending Tag Data in the *InTouch® HMI Data Management Guide*
- 3** Configure the application for distributed logging. See "Configuring Distributed Historical Logging" on page 44.
- 4** Distribute the application. See "Configuring an InTouch Application for NAD" on page 46.

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.

Configuring 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 Wonderware Historian history provider.

Note: A remote history provider cannot be configured for an InTouchView application. For more information about the limitations of InTouchView applications, see "About InTouchView Applications" on page 218.

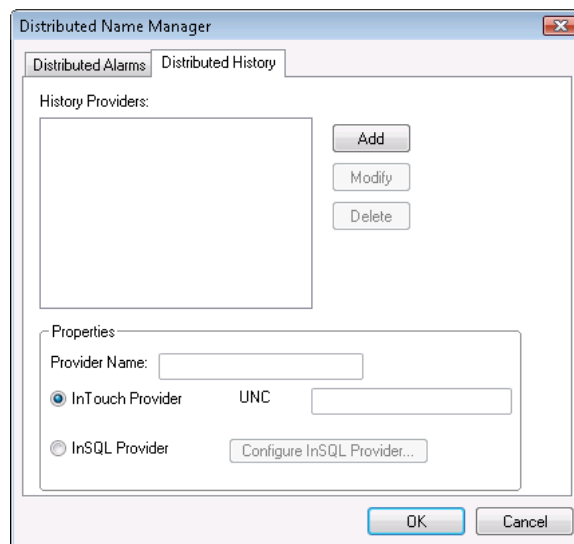
While the local InTouch application is considered a history provider, you do not need to define it for your application.

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 Wonderware Historian provider.

To configure a history provider

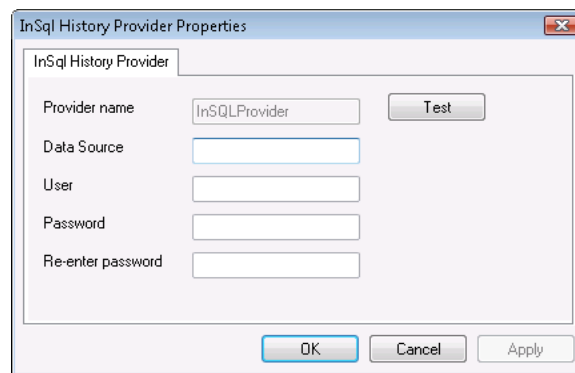
- 1 On the **Special** menu, point to **Configure**, and then click **Distributed Name Manager**. The **Distributed Name Manager** dialog box appears.
- 2 Click the **Distributed History** tab.



- 3 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.
- 4 To configure an InTouch history provider, do the following:
 - a Click **InTouch Provider**.
 - b In the **UNC** box, type the UNC path to the InTouch application directory and then click **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.
- 5 To configure a Wonderware Historian history provider, do the following:
 - a Click **InSQL Provider**.
 - b Click **Configure InSQL Provider**. The **InSql History Provider Properties** dialog box appears.



- c In the **Data Source** box, type the name, up to 35 characters, of the node where the Wonderware Historian database resides.
 - d In the **User** box, type the user name for the logon account. The user account must have database permissions to retrieve data.
 - e In the **Password** and **Re-enter password** boxes, type the password for the logon account.
 - f Click **Test** to validate the connection to the Wonderware Historian. When a message appears, click **OK**.
 - g Click **OK** to close the **InSql History Providers Properties** dialog box.
- 6 Click **OK**.

Dynamically Configuring 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.

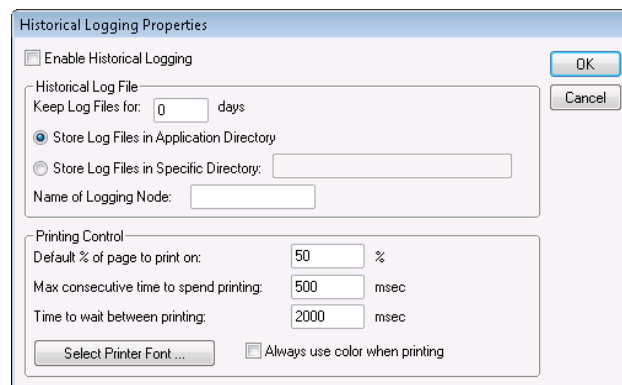
Configuring Distributed Historical Logging

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 "About InTouchView Applications" on page 218.

To configure distributed historical logging

- 1 On the **Special** menu, point to **Configure**, and then click **Historical Logging**. The **Historical Logging Properties** dialog box 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 Click **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.

Configuring 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.

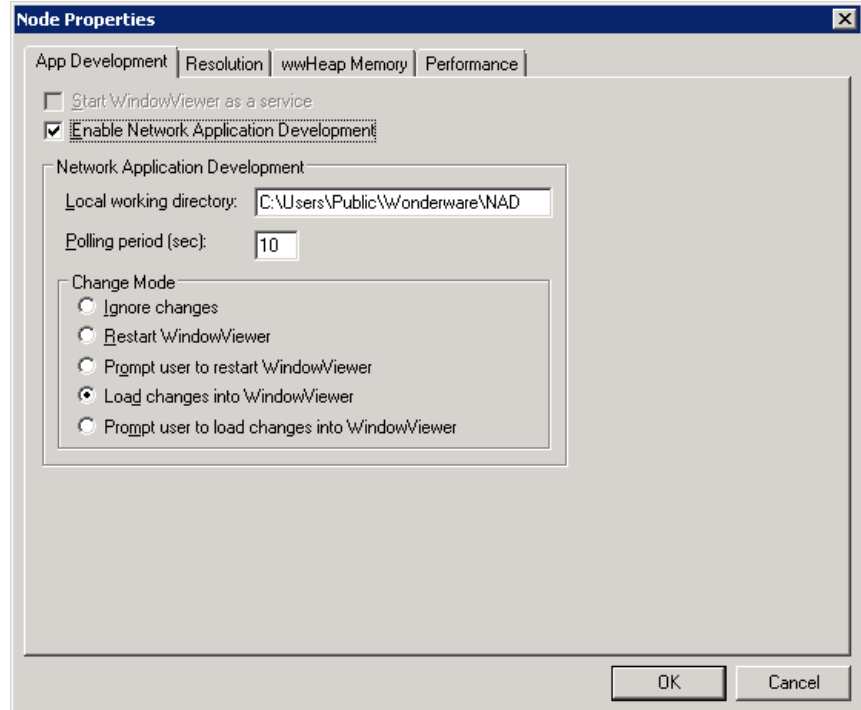
You cannot use NAD update features if you are running WindowViewer as a service. For more information, see "Running WindowViewer as a Service" on page 66.

To configure an application for NAD



- 1 Start Application Manager.

On the **Tools** menu, click **Node Properties**. The **Node Properties** dialog box appears.



- 2 Select the **Enable Network Application Development** check box.
- 3 In the **Local working directory** box, type the path to the folder that you want WindowViewer to copy the master application.
- 4 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.

- 5 In the **Change Mode** area, select the option that determines the action WindowViewer takes when the master application changes.
- Click **Ignore changes** to have the WindowViewer node ignore any changes made on the development node.
 - Click **Restart Window Viewer** to have the WindowViewer node copy over the updated master application (if configured to do so) and then restart itself.
 - Click **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.
 - Click **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.

- Click **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.
- 6 Click **OK**.

Performing 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.

To perform an automatic update

- 1 Open the application in WindowMaker.
- 2 On the **Special** menu, click **Notify Clients**.

Performing 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 "Configuring an InTouch Application for NAD" on page 46.

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.lock` file may not be deleted. You can manually remove the lock by deleting the `appedit.lock` 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.

Scaling 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.

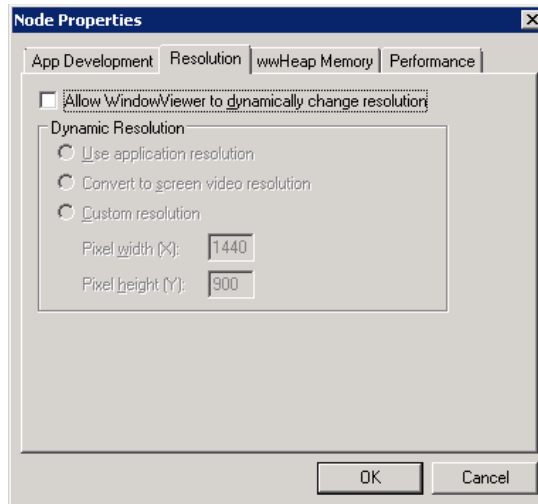
Each View node can scale the application appropriately, including scaling to a custom resolution. This scaling takes place while WindowViewer compiles the application and does not require WindowMaker. Because each View node can use a different DRC setting, each View node must have its own settings configured.

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.

To configure an application for DRC



- 1 Start **Application Manager**.
- 2 On the **Tools** menu, click **Node Properties**. The **Node Properties** dialog box appears.
- 3 Click 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.
 - Select **Custom Resolution** if you want WindowViewer to run the application at a specific resolution you specify in the **Pixel width (X)** and **Pixel height (Y)** (must be integer values) boxes. The application's resolution and the node's resolution are both ignored. For example, if **Pixel width (X)** and **Pixel 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.
- 6 Click **OK**.

Locking the Application Resolution

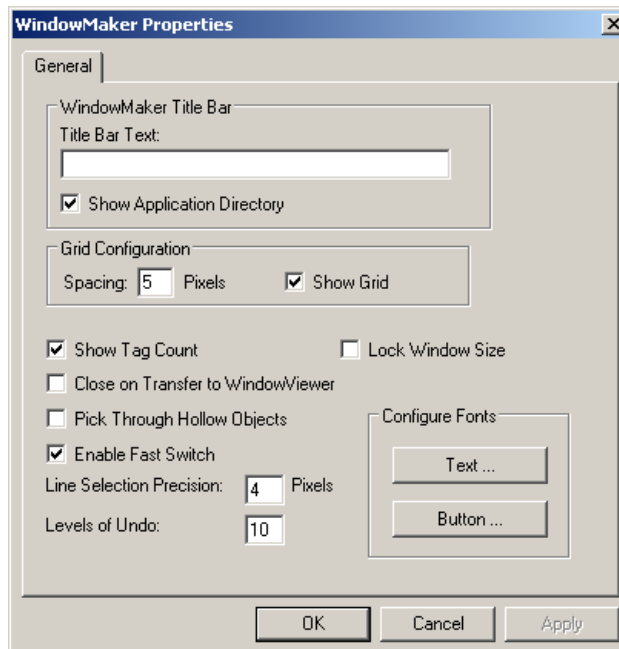
You can configure the WindowMaker properties to lock the size of InTouch application windows. This allows you to convert applications to a different resolution without scaling the windows and graphics.

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.

To lock the application resolution from WindowMaker

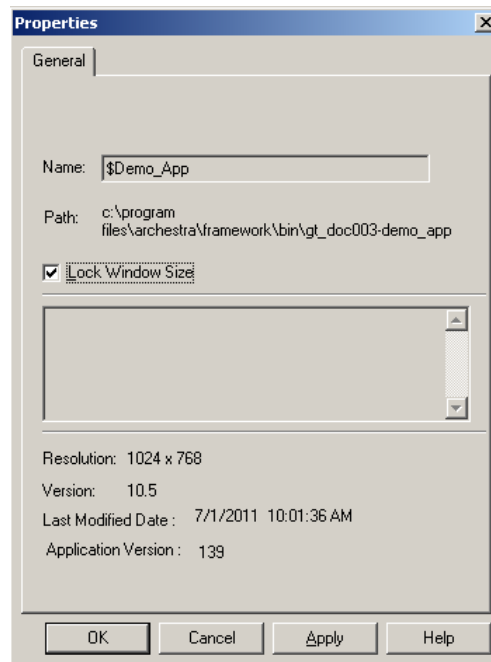
- 1 Open WindowMaker.
- 2 On the **Special** menu, point to **Configure** and then click **WindowMaker**. The **WindowMaker Properties** dialog box appears.



- 3 Select the **Lock Window Size** check box. By default, the check box is not selected.
- 4 Click **OK**.

To lock the application resolution from Application Manager

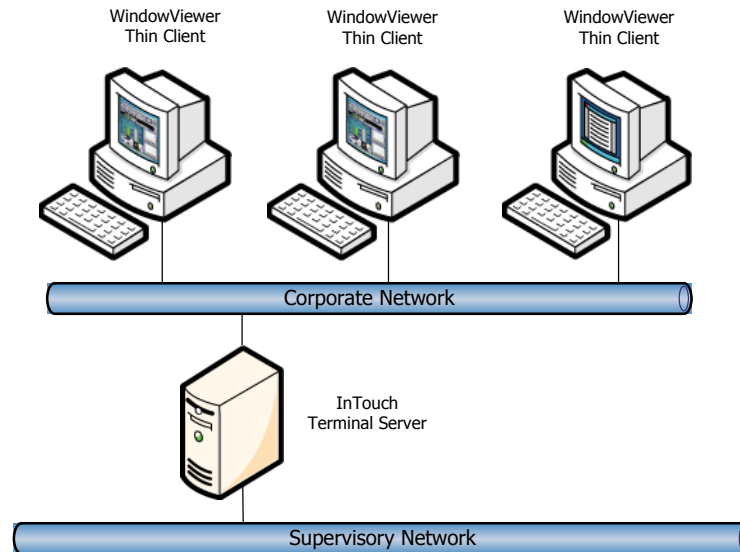
- 1 Open Application Manager. Click to select the application you want to configure.
- 2 Click **File** on the menu bar, then click **Properties**. The **Properties** dialog box appears.



- 3 Select the **Lock Window Size** check box. By default, the check box is not selected.
- 4 Click **OK**.

Using Terminal 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.

Planning Considerations 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, Wonderware 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.

Deploying 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, Wonderware Historian, 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 "Securing InTouch" on page 131.

- 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.

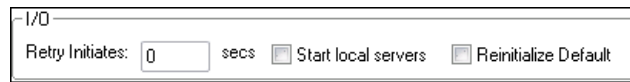
Use security auditing to monitor intrusion attempts. If you suspect that your system is under any sort of attack, then you can enable logging for an array of auditable events. By default, security logging/auditing is disabled because it usually requires excessive processing resources.

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.

To avoid receiving an “initializing I/O” error message when WindowViewer starts, clear the **Start Local Servers** check box on the **General** tab on the WindowViewer **Properties** dialog box.



Note: Running WindowViewer as a Windows Service is not a supported configuration because of the Terminal Services architecture.

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.

Logging 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.
AlarmSuite Logger	No	Use a tag server (separate computer only).
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.
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.

Retrieving 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;
```


Chapter 3

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 login.

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.

Services are started without compromising the Windows security system.

The InTouch HMI includes the following Windows services:

- Alarm DB Logger
- Alarm DB Purge/Archive
- Wonderware NetDDE Helper
- Wonderware SuiteLink
- Wonderware WindowViewer

Running 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 until a user logs on, at which point WindowViewer opens.

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.

When WindowViewer runs as a service, if you attempt to start WindowViewer from a shortcut icon or by clicking WindowViewer on the Windows **Start** menu, a message appears. The message describes the restrictions to starting WindowViewer when it has been configured to run as a service.

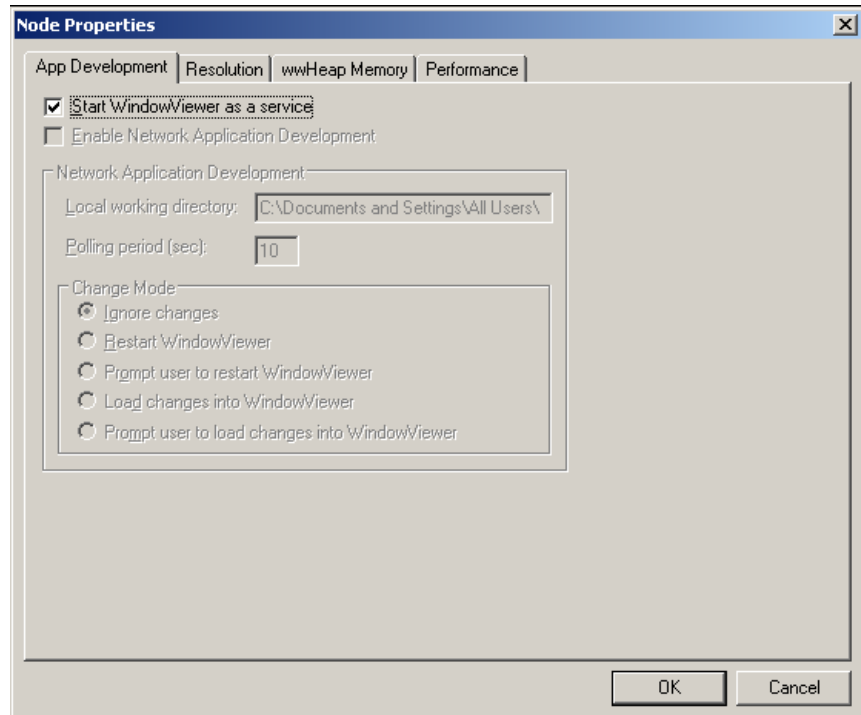
Important: When running WindowViewer as a service, the user account privileges have been set to non-interactive to reduce the potential security exposure of running a service with administrator privileges.

Configuring 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.

To configure WindowViewer to start as a service

- 1 Start InTouch. The InTouch Application Manager appears.
- 2 On the **Tools** menu, click **Node Properties**. The **Node Properties** dialog box appears.



- 3 Select the **Start WindowViewer as a service** check box to configure WindowViewer to automatically run as a service.
- 4 Click **OK**.

Manually Starting a Service

You can manually start the Wonderware 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 "Configuring WindowViewer to Start as a Service" on page 66.

To 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 click **Start**.

To start the WindowViewer service using a command prompt

1 Start a command prompt.

2 Type `Net Start view`.

view is the name of the Wonderware WindowViewer executable program file.

Stopping a Service

You can manually stop the WindowViewer service using the Control Panel.

To 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 **Wonderware WindowViewer** and then click **Stop**.

To stop the service using a command prompt

4 Start a command prompt.

5 Type `Net Stop View`.

View is the name of the WindowViewer executable program file.

Configuring 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 ArchestrA services run under, if the account was not created already. This account is considered the master ArchestrA account. The InTouch services use the master ArchestrA 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 ArchestrA Change Network Account Utility.

Caution: Changing the master account affects all ArchestrA services, not just InTouch services.

To change the ArchestrA master account

- 1 On the **Start** menu, point to **Programs**, point to **Wonderware**, point to **Common** and then click **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 Click **OK**.

Troubleshooting InTouch Services

Important: Running WindowViewer as a service is not supported on the Windows Vista operating system.

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 Wonderware 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 Wonderware 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 Wonderware 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 Wonderware SuiteLink service must be running.

The Wonderware 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 ArchestrA Logger service is installed.

Both the Wonderware SuiteLink and ArchestrA Logger services should be installed and configured to run in automatic startup.

Viewing 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 Wonderware WindowViewer service failed to start, the most likely cause is a dependency on a prerequisite service that is not running.

Troubleshooting 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.

To 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 "Configuring the User Account for InTouch Services" on page 68.

- 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.

Deactivating 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:

Wonderware SuiteLink:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SLS
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\slssvc
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SuiteLink

Wonderware NetDDE Helper:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WWNetDDE

Wonderware WindowViewer:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VIEW

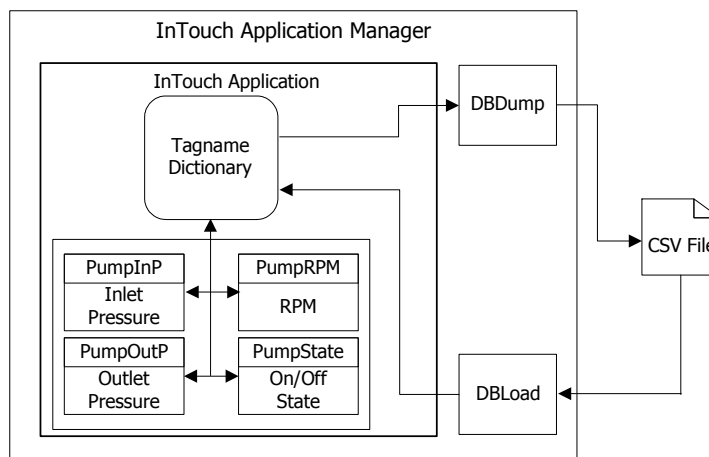
Chapter 4

Exporting and Importing Tag Definitions, Windows, and Scripts

You can create InTouch applications more quickly by importing or exporting some or all of the components of an existing application. You can import tag definitions, windows, and scripts from your existing application to a new application.

Exporting 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.

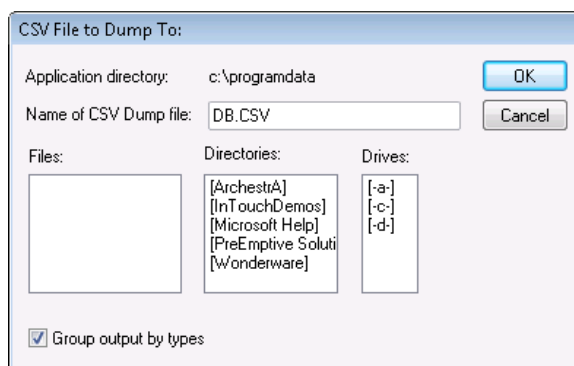


You use the DBDump utility within the Application Manager to export the contents of the Tagname Dictionary to a Comma Separated Value (CSV) file. You can view and edit the exported file with Microsoft Notepad or Microsoft Excel. After making edits, you then import the tag definitions to an InTouch application 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.

To export tag definitions

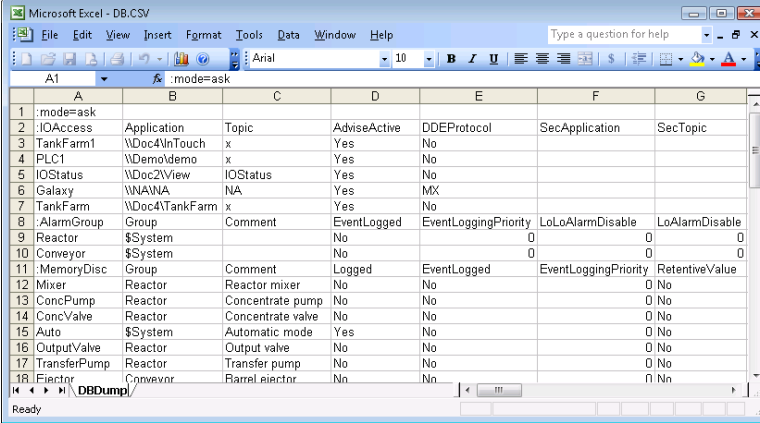
- 1 Close WindowMaker and WindowViewer.
- 2 Start Application Manager. The **Application Manager** dialog box shows a list of InTouch applications.
- 3 Select the application from the list.
- 4 Click the DBDump icon. The **CSV File to Dump To:** dialog box appears.



- 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 Click **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.

Viewing 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\UnTouch	x	Yes	No		
4	PLCT	\Doc4\demo	x	Yes	No		
5	IOStatus	\Doc2\View	IOStatus	Yes	No		
6	Galaxy	\NANA	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.

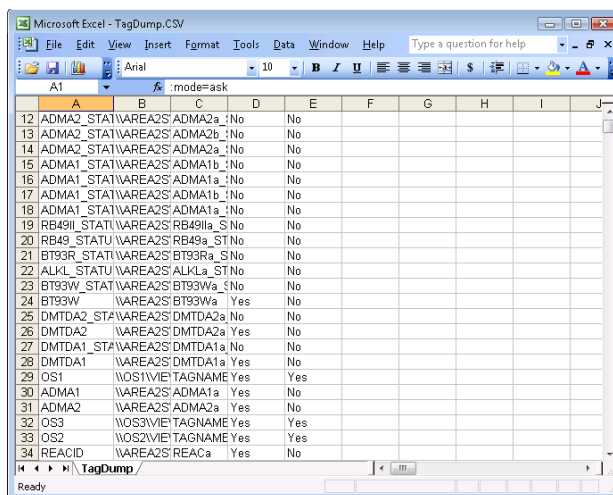
Importing Tag Definitions

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 instead of the InTouch TemplateMaker to create SuperTag instances. For more information, see "Creating SuperTag Instances" on page 113.

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 "Creating an Import File Template" on page 77.

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.
:MemoryDisc	Memory discrete tags.

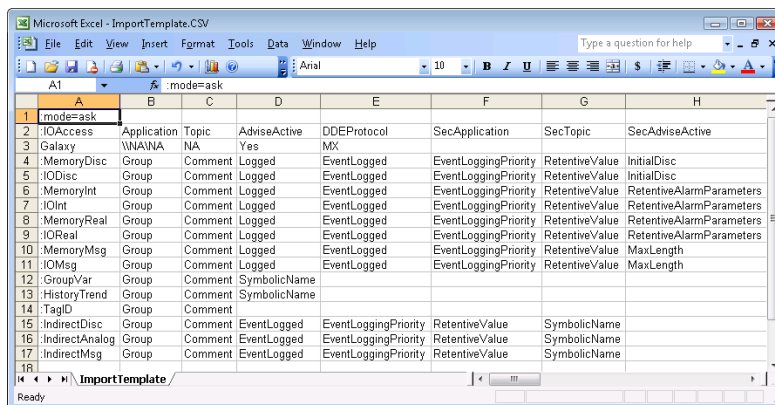
Keyword	Description
:IODisc	I/O discrete tags.
:MemoryInt	Memory integer tags.
:IOInt	I/O integer tags.
:MemoryReal	Memory real tags.
:IOReal	I/O real tags.
:MemoryMsg	Memory message tags.
:IOMsg	I/O message tags.
:GroupVar	Group Var tags.
:HistoryTrend	Hist Trend tags.
:TagID	Tag ID tags.
:IndirectDisc	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.

Creating 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
mode=ask							
IOAccess	Application	Topic	AdviseActive	DDEProtocol	SecApplication	SecTopic	SecAdviseActive
Galaxy	\\NA\NA	NA	Yes	MX			
MemoryDisc	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	InitialDisc
IODisc	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	InitialDisc
MemoryInt	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	RetentiveAlarmParameters
IOInt	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	RetentiveAlarmParameters
MemoryReal	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	RetentiveAlarmParameters
IOReal	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	RetentiveAlarmParameters
MemoryMsg	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	MaxLength
IOMsg	Group	Comment	Logged	EventLogged	EventLoggingPriority	RetentiveValue	MaxLength
GroupVar	Group	Comment	SymbolicName				
HistoryTrend	Group	Comment	SymbolicName				
TagID	Group	Comment					
IndirectDisc	Group	Comment	EventLogged	EventLoggingPriority	RetentiveValue	SymbolicName	
IndirectAnalog	Group	Comment	EventLogged	EventLoggingPriority	RetentiveValue	SymbolicName	
IndirectMsg	Group	Comment	EventLogged	EventLoggingPriority	RetentiveValue	SymbolicName	

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.

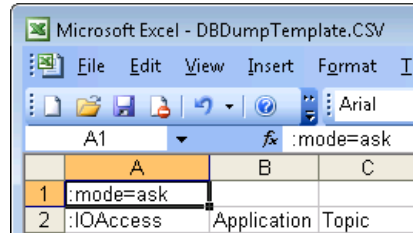
To create a template import file

- 1 Open the Application Manager.
- 2 Create a new InTouch application.
For more information about the steps to create an application, see "Creating an InTouch Application" on page 18.
- 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 "Exporting Tag Definitions" on page 73.
- 5 Edit the file to insert tag data that you want to import.

Setting 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 an 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

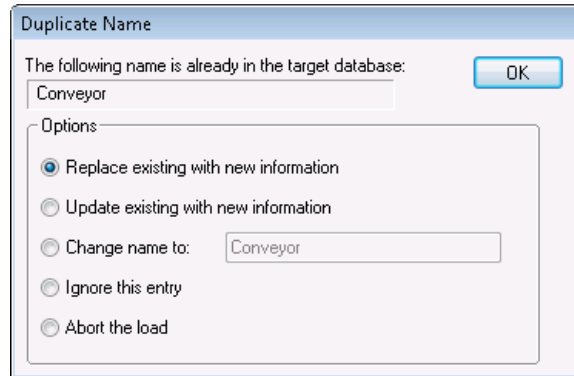
Tagname4,,

; 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:

- Click **Replace existing with new information** to replace the existing tag record with the record from the import file.
- Click **Update existing with new information** to overwrite the existing tag record with only the fields that are explicitly defined in the import file.
- Click **Change Name to** to replace the name of the imported tag with the name you type in the box of the **Duplicate Name** dialog box.
- Click **Ignore this entry** to ignore the tag and continue importing the contents of the file.
- Click **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.

Setting 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.

	:IOAccess	Topic Attribute	DDEProtocol	SecTopic Attribute				
	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	RSLINK	M22	Yes	Yes			
5	IOStatus	View	IOStatus	Yes	Yes			
6	Galaxy	WNAWA	NA	Yes	MX			

Application Attribute AdviseActive SecApplication

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

String Position	Attributes	Acceptable Values	Default Values
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
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

String Position	Attributes	Acceptable Values	Default Values
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 condition clears	None
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
12	DSCAlarmDisable	Discrete alarms disabled or enabled 0 = Discrete alarm enabled 1 = Discrete alarm disabled	0

String Position	Attributes	Acceptable Values	Default Values
13	LoLoAlarmInhibitor	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
20	DSCAlarmInhibitor	Name of the tag used to inhibit discrete alarms Tag reference: any discrete or analog tag	None

Defining 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.

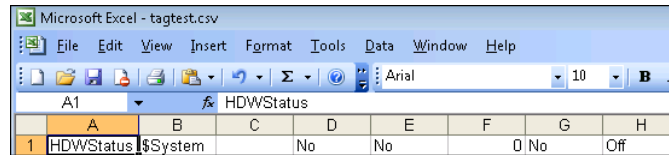
```
:IODisc,Group,Comment,Logged,EventLogged,
EventLoggingPriority,RetentiveValue,InitialDis,
OffMsg,OnMsg,AlarmState,AlarmPri,DConversion,
AccessName,ItemUseTagname,ItemName,ReadOnly,
AlarmComment,AlarmAckModel,DSCAlarmDisable,
DSCAlarmInhibitor,SymbolicName
```

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, "Histda
taViewstr", 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.



The screenshot shows a Microsoft Excel window titled 'tagtest.csv'. The spreadsheet has columns A through H. Row 1 contains the tag name 'HDWStatus' in column A, and the remaining cells are blank. Row 2 contains the attribute values: '\$System' in column B, 'No' in column C, 'No' in column D, '0' in column E, 'No' in column F, 'Off' in column G, and a blank cell in column H.

	A	B	C	D	E	F	G	H
1	HDWStatus	\$System		No	No	0	No	Off

Tag Keyword Attributes

The following table lists all attributes associated with InTouch tag keywords. The table includes 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

Attribute	Acceptable Value	Default Value
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
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

Attribute	Acceptable Value	Default Value
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
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

Attribute	Acceptable Value	Default Value
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
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

Attribute	Acceptable Value	Default Value
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
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

Attribute	Acceptable Value	Default Value
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
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

Attribute	Acceptable Value	Default Value
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
ROCArmPri	Priority assigned to Rate of Change alarm 1 to 999	1
ROCArmState	Rate of Change alarm enabled or disabled No or Off = Disabled Yes or On = Enabled	No
ROCArmValue	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 Wonderware 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" on page 88 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
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" on page 88 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
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" on page 88 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
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

String Position	Attribute
26	HiHiAlarmValue
27	HiHiAlarmPri
28	MinorDevAlarmState
29	MinorDevAlarmValue
30	MinorDevAlarmPri
31	MajorDevAlarmState
32	MajorDevAlarmValue
33	MajorDevAlarmPri
34	DevTarget
35	ROCArmState
36	ROCArmValue
37	ROCArmPri
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

String Position	Attribute
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" on page 88 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
12	MinEU
13	MaxEU
14	Deadband
15	LogDeadband
16	LoLoAlarmState

String Position	Attribute
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	ROCArmState
36	ROCArmValue
37	ROCArmPri
38	ROCTimeBase
39	AlarmComment
39	MinRaw
40	MaxRaw
41	Conversion
42	AccessName

String Position	Attribute
43	ItemUseTagname
44	ItemName
45	ReadOnly
46	AlarmComment
47	AlarmAckModel
48	LoLoAlarmDisable
49	LoAlarmDisable
50	HiAlarmDisable
51	HiHiAlarmDisable
52	MinDevAlarmDisable
53	MajDevAlarmDisable
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" on page 88 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
12	MinValue
13	MaxValue
14	Deadband
15	LogDeadband
16	LoLoAlarmState
17	LoLoAlarmValue
18	LoLoAlarmPri
19	LoAlarmState
20	LoAlarmValue
21	LoAlarmPri

String Position	Attribute
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	ROCArmState
36	ROCArmValue
37	ROCArmPri
38	ROCTimeBase
39	AlarmComment
40	AlarmAckModel
41	LoLoAlarmDisable
42	LoAlarmDisable
43	HiAlarmDisable
44	HiHiAlarmDisable
45	MinDevAlarmDisable
46	MajDevAlarmDisable
47	RocAlarmDisable
48	LoLoAlarmInhibitor

String Position	Attribute
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" on page 88 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
12	MinEU

String Position	Attribute
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
29	MinorDevAlarmValue
30	MinorDevAlarmPri
31	MajorDevAlarmState
32	MajorDevAlarmValue
33	MajorDevAlarmPri
34	DevTarget
35	ROCArmState
36	ROCArmValue
37	ROCArmPri
38	ROCTimeBase
39	MinRaw

String Position	Attribute
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
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" on page 88 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" on page 88 for the data associated with these attributes and their default values.

String Position	Attribute
1	Group
2	Comment
3	Logged

String Position	Attribute
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" on page 88 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" on page 88 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" on page 88 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" on page 88 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" on page 88 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" on page 88 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

Using 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.

Using 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.

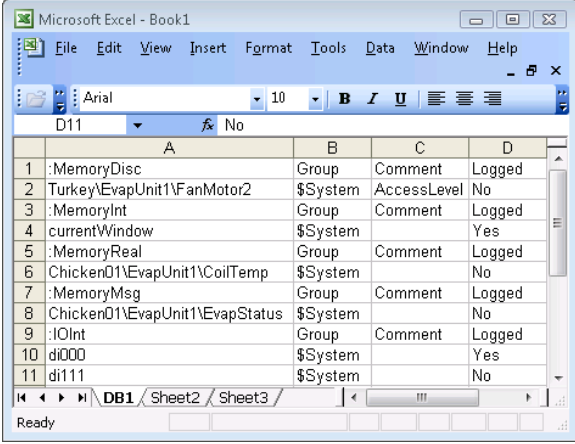
You can reset an individual keyword to its original default value by omitting the value in the equation. For example, :GROUP=.

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.

Creating 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 in the TemplateMaker.

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:



The screenshot shows a Microsoft Excel window titled 'Microsoft Excel - Book1'. The active sheet is 'DB1 / Sheet2 / Sheet3'. The table contains the following data:

	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.

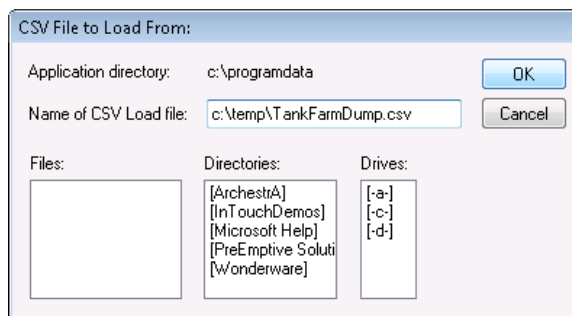
Importing Tag Definitions with DBLoad

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.

To 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 Click the DBLoad icon. A message appears requesting confirmation that you backed up the InTouch application.
- 6 Click **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 Click **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 "Setting the Operating Mode for Dictionary Import Files" on page 79.

Importing 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" on page 128. If the associated tags already exist in the target application, during the import you can select to use these instead.

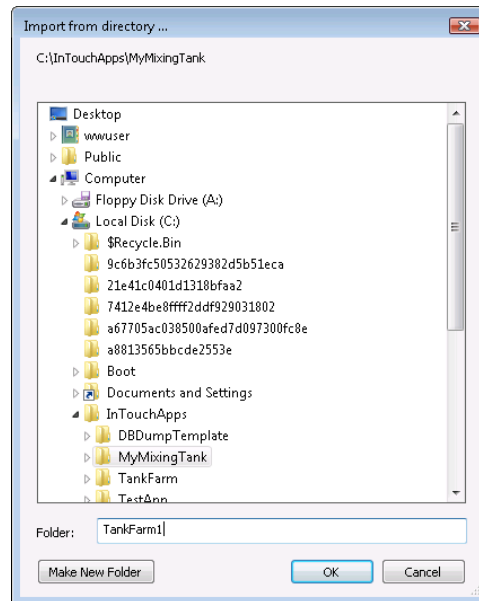
When you import windows containing SmartSymbols and select to use existing tags, the InTouch HMI still keeps placeholders for the recovered symbols, even though the tags are available in the target application.

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.

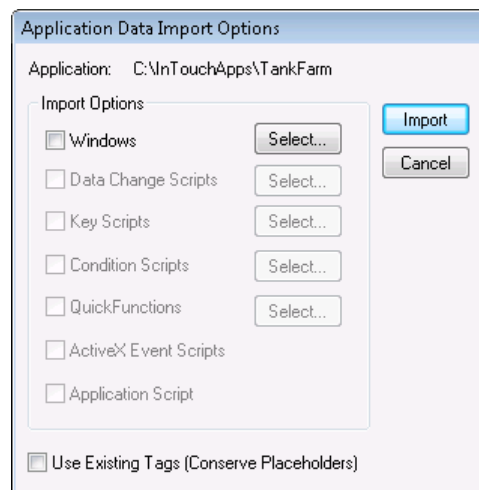
Important: If you move InTouch window files using any method other than importing or exporting them, the contents of the application Tagname Dictionary can become corrupt.

To import a window

- 1 Close all windows in your current application.
- 2 On the **File** menu, click **Import**. The **Import from directory** dialog box appears.



- 3 Select the folder for the application containing the windows to import.
- 4 Click **OK**. The **Application Data Import Options** dialog box appears.



- 5 Select the **Windows** check box and then click **Select** to select the individual windows to import.
- 6 Select the **Use Existing Tags (Conserve Placeholders)** check box if the tags associated with the imported windows already exist in your application and you want to use them instead of placeholders.

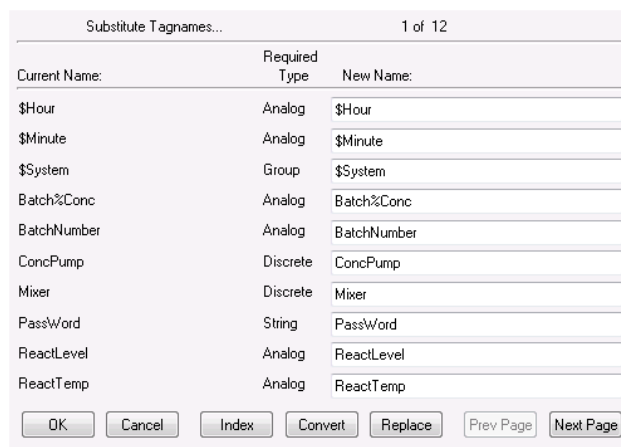
- 7 Click **Import**.
- 8 Convert the placeholder tags to either local tags or remote tag references. For more information, see "Converting Placeholder Tags for an Imported Window" on page 117.
- 9 If an imported window contains one or more wizards, double-click on each wizard to open its properties panel. If an imported window contains one or more SmartSymbols, edit each SmartSymbol and create new instances.

Converting Placeholder Tags for an Imported Window

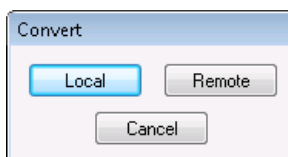
When you import or export a window or QuickScript to or from your current application, all the tags associated with that window or QuickScript are transferred with the window. But, the tags are not added to your new application's Tagname Dictionary. Instead, the tags are automatically marked as placeholder 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 **Special** menu, click **Substitute Tags**. The **Substitute Tagnames** dialog box appears.



- 4 Click **Convert**. The **Convert** dialog box appears.



5 Convert the tags.

- Click **Local** to convert the placeholder tags to local tags. You are prompted to define each tag in the Tagname Dictionary.
- Click **Remote** to convert the placeholder tags to remote tag references. The **Access Names** dialog box appears. Select the Access Name and click **Close**.

After the conversion, the **Substitute Tagnames** dialog box shows the new tags.

The dialog box titled "Substitute Tagnames..." shows "1 of 13" pages. It contains a table with three columns: "Current Name:", "Required Type", and "New Name:". The table lists the following tags:

Current Name:	Required Type	New Name:
Auto	Discrete	PLC1:Auto
ConcPump	Discrete	PLC1:ConcPump
ConcValve	Discrete	PLC1:ConcValve
Mixer	Discrete	PLC1:Mixer
OutputValve	Discrete	PLC1:OutputValve
PassWord	String	PLC1:PassWord
ProdLevel	Analog	PLC1:ProdLevel
ReactLevel	Analog	PLC1:ReactLevel
ReactTemp	Analog	PLC1:ReactTemp
SteamValve	Discrete	PLC1:SteamValve

At the bottom of the dialog box are buttons for "OK", "Cancel", "Index", "Convert", "Replace", "Prev Page", and "Next Page".

6 Click **OK**.

Exporting Windows

You can export application windows to:

- Create or maintain a library application of all windows.
- Create remote tag references in another application.

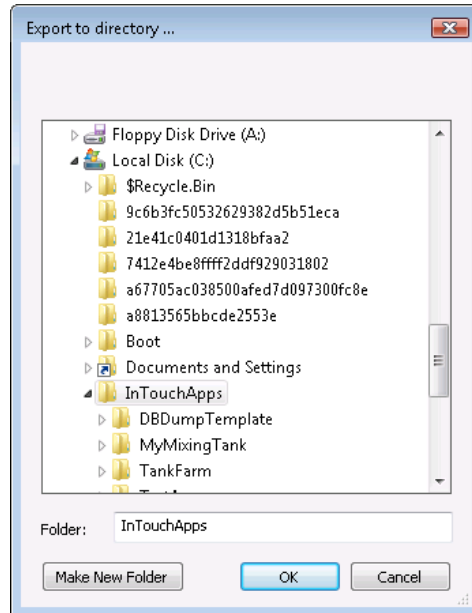
You must convert an application to the current version of the InTouch HMI software before you can export windows.

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 "Converting Placeholder Tags for an Imported Window" on page 117.

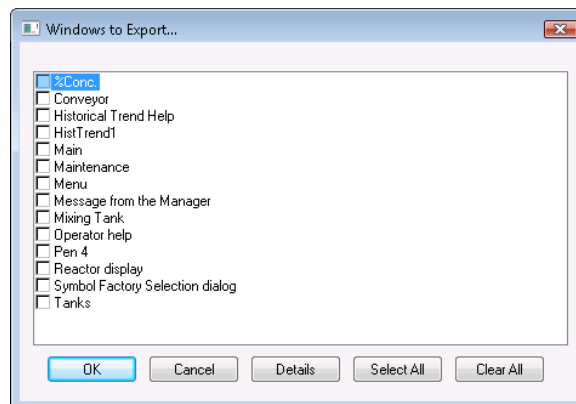
Important: If you move InTouch window files using any method other than importing or exporting them, the application's Tagname Dictionary can be corrupted.

To export a window

- 1 Close all windows in your current application.
- 2 On the **File** menu, click **Export Window**. The **Export to directory** dialog box appears.



- 3 Select the folder of the application to which to export the windows.
- 4 Click **OK**. The **Windows to Export** dialog box appears.



- 5 Select the windows to export.
- 6 Click **OK**.

If a problem occurs, the **Problem with Export Operation** dialog box appears. Click the option for the action you want to take and then click **OK**.

Converting InTouch Windows to ArcestrA Symbols

You can convert the windows of a managed InTouch application to ArcestrA symbols. The converted ArcestrA symbols appear in the WindowMaker ArcestrA Graphic Toolbox and the ArcestrA IDE Graphic Toolbox. In addition to graphics that appear in windows, InTouch scripts are converted to ArcestrA scripts.

Preparing to Convert Windows

- Before you convert InTouch windows:
 - Only windows from InTouch managed applications can be converted to ArcestrA symbols.
 - Windows must be closed from WindowMaker to be converted.
- You can convert windows by two methods:
 - Use the **File** menu method when you want to convert multiple windows simultaneously.
 - Use the shortcut menu method when you want to convert a single selected window.

Converting 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 ArcestrA graphic primitives.
- An InTouch Smart Symbol is converted to an ArcestrA embedded symbol.
- An ArcestrA symbol 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 ArcestrA Windows controls.

- Some InTouch graphic components cannot be converted to ArcestrA symbols:
 - InTouch Real-time and Historical trends cannot be converted.
 - ActiveX controls and the Distributed Alarm Display cannot be converted.

Converting Animation Scripts

All InTouch animation links embedded in windows are converted to the corresponding ArcestrA animation.

No validation warning or error messages are logged during the conversion. You should validate converted scripts with ArcestrA symbol 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 ArcestrA 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 ArcestrA attribute reference handling.

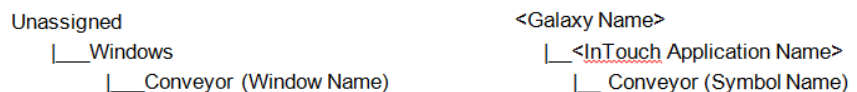
After Converting Windows

After conversion, the symbol is added to the ArcestrA IDE **Toolbox** and the WindowMaker **ArcestrA 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 ArcestrA symbols.

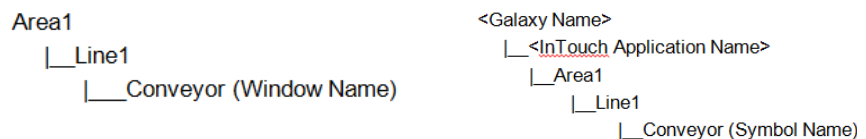
The name of the converted window is assigned by default as the name of the new ArcestrA symbol. If the InTouch window name contains unsupported ArcestrA characters, an underscore (_) replaces each unsupported character. If the symbol already exists, a numerical suffix is appended to the name of the converted symbol. For example, Main_001.

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.



Diagnosing 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 to log useful diagnostic information for every element, animation, or script while converting windows.

After conversion, click **View 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.

Completing the Window Conversion Procedure

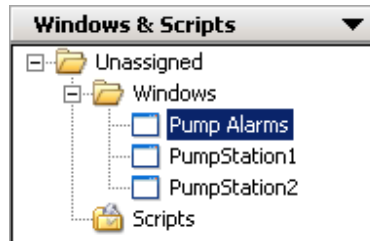
To convert windows to ArchedrA symbols

- 1 If necessary, close all InTouch windows that will be converted.
- 2 Convert one or more windows to ArchedrA symbols by one of the following methods:

Shortcut Menu Method

Use this method when you want to convert a single window.

- a Select the window to be converted from the **Windows & Scripts** pane.



- b Right-click to show the shortcut menu and select **Convert to ArchedrA Symbol**.

A message appears and indicates the window is being converted.

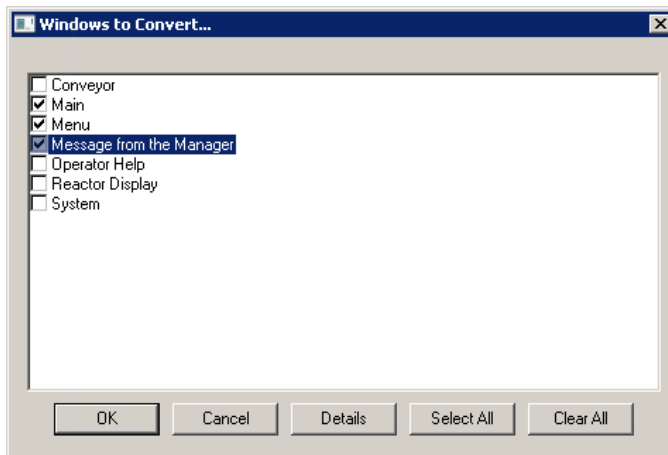
- c Continue at Step 3.

File Menu Method

Use this method when you want to convert multiple windows simultaneously.

- a Click **File** from the menu bar and select **Convert to ArchedrA Symbol**.

The **Windows to Convert** dialog box appears to select one or more windows.



- b** Select the windows that you want to convert.
- c** Click **OK**.

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.

- 3** Observe the WindowMaker **ArchestrA Graphic Toolbox** and the ArchestrA IDE **Graphic Toolbox**.

The converted windows should appear as ArchestrA symbols in both tool boxes.



Importing 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.

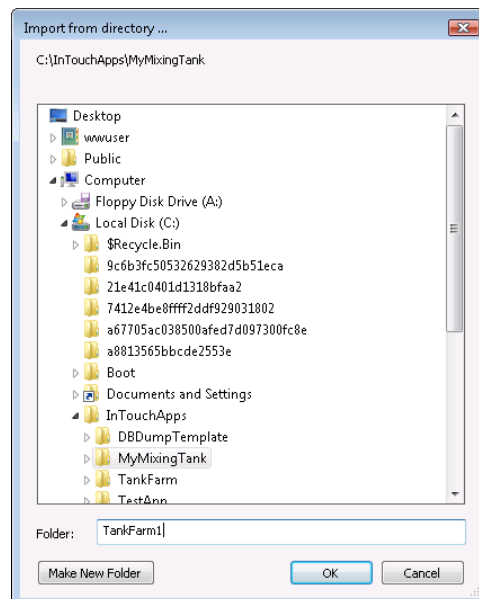
By default, placeholders are created for the tags associated with an imported QuickScript. After importing, you can convert the placeholders to local tags or remote tag references. For more information, see "Tag Placeholders for Imported Windows and Scripts" on page 128. If the associated tags already 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.

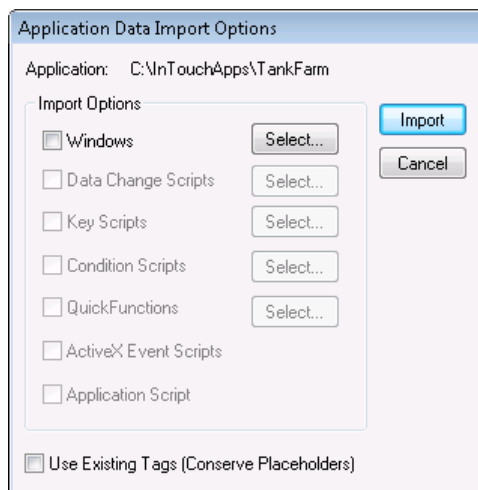
To import a QuickScript

- 1 Close all windows in your current application.
- 2 On the **File** menu, click **Import**. The **Import from directory** dialog box appears.



- 3 Select the folder for the application that contains the scripts to import.

- 4 Click **OK**. The **Application Data Import Options** dialog box appears.



- 5 Select the check box for the QuickScript type(s) that you want to import and then click **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 "Importing Windows" on page 115.

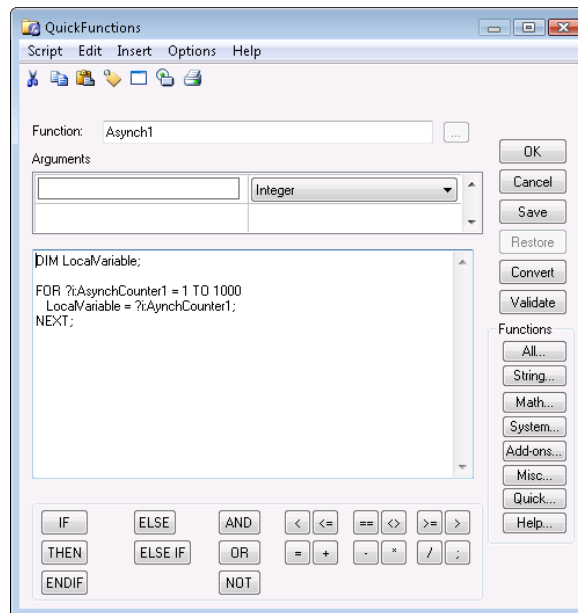
- 6 Select the **Use Existing Tags (Conserve Placeholders)** check box if the tags associated with the imported script(s) already exist in your application and you want to use them instead of placeholders.
- 7 Click **Import**. If your application has scripts with identical names, you are prompted to overwrite, skip, or rename.
- 8 Convert the placeholder tags to either local tags or remote tag references. For more information, see "Converting Placeholder Tags in an Imported Script" on page 127.

Converting Placeholder Tags in an Imported Script

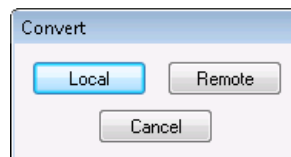
When you import or export a QuickScript to or from your current application, all the tags associated with that QuickScript are transferred. But, the tags are not added to your new application's Tagname Dictionary. Instead, they are automatically marked as placeholder tags. You must convert these placeholder tags and, if required, define them in your new application Tagname Dictionary.

To convert placeholder tags in an imported script

- 1 On the **Special** menu, point to **Scripts**, and then click the type of QuickScript you imported. The QuickScript editor appears, showing the first QuickScript on file for the selected type of script.



- 2 Click **Convert**. The **Convert** dialog box appears.



- 3 Convert the tags.
 - Click **Local** to convert the placeholder tags to local tags. You are prompted to define each tag in the Tagname Dictionary.
 - Click **Remote** to convert the placeholder tags to remote tag references. The **Access Names** dialog box appears. Select the Access Name and click **Close**.
- 4 After the tags are converted, click **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.

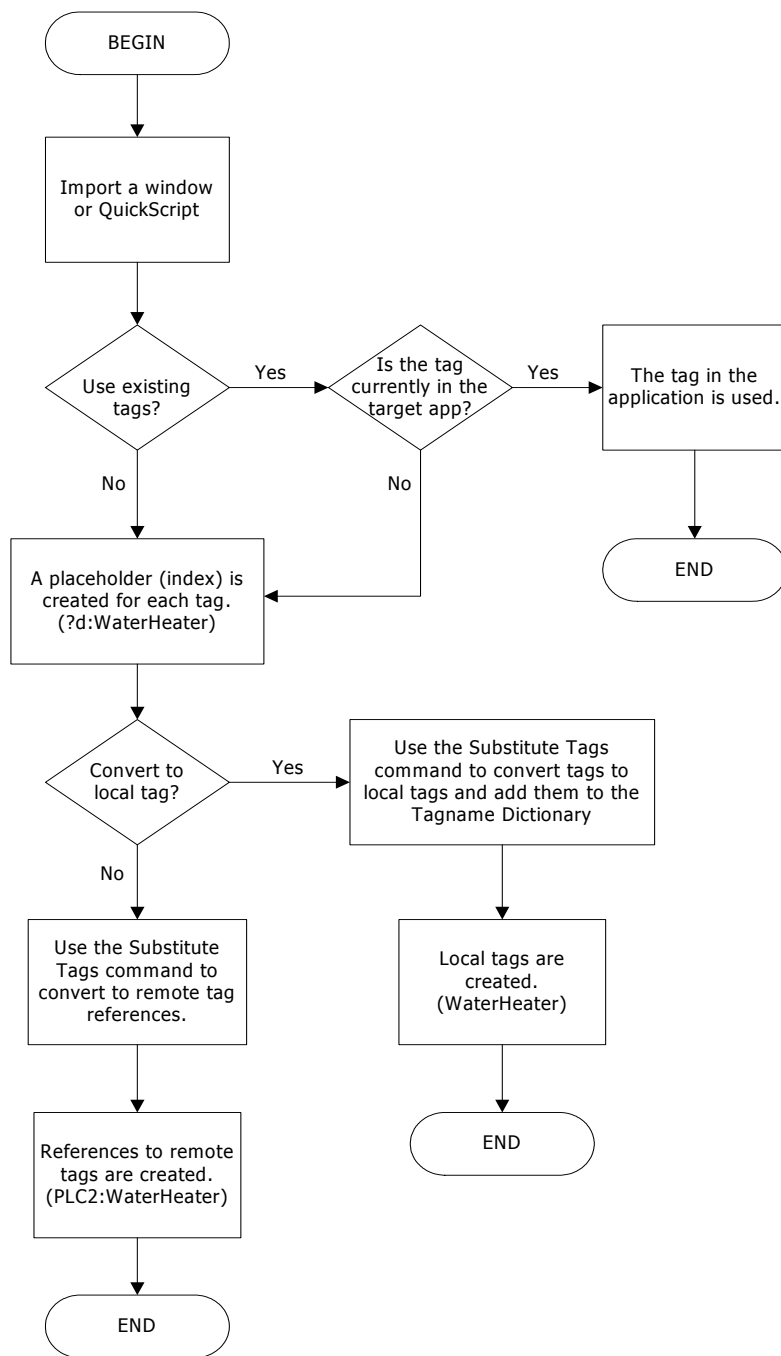
If you import a tag that contains 30, 31, or 32 characters, the placeholder prefix is still added to the beginning of the tag, and the length of the existing tag is not truncated. For example, for placeholder tags only, a 32 character tag is increased to 35 characters. This increase in tag length is 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.



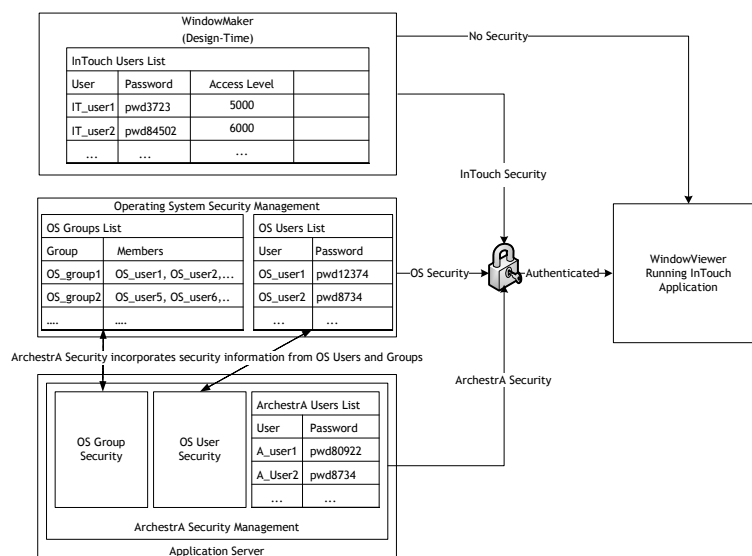
Chapter 5

Securing 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

Configuring 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.

Note: The inactivity timer does not reset for Active-X controls, and OLE Automation controls.

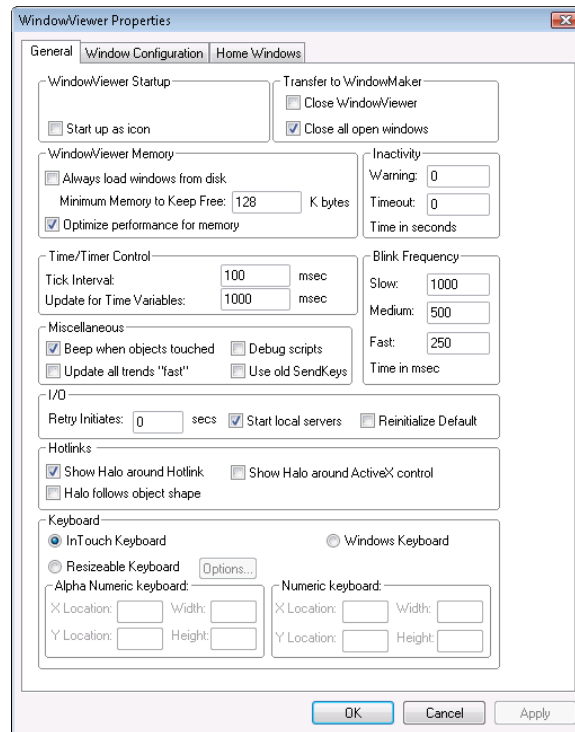
Automatically logging off an operator is a three-step process:

- 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.
- 3** You can write a script to automatically log off the operator if the `$InactivityTimeout` tag is set to 1.

You can use the time-out feature independently of the warning feature.

To configure an inactivity time-out

- 1 On the **Special** menu, point to **Configure**, and then click **WindowViewer**. The **WindowViewer Properties** dialog box appears.



- 2 In the **Inactivity** 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.
- 3 Click **OK**.
- 4 To show the 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";
```
- 5 To log off the user and show the window named "Logged Off" after the inactivity timeout elapses, create a condition script with "\$InactivityTimeout" as the condition and the following script body:


```
LogOff();  
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 "Configuring an Inactivity Time-Out" on page 132.

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 by mouse clicks or keyboard activity only. For more information on setting the log off warning, see "Configuring an Inactivity Time-Out" on page 132.

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

Locking 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.

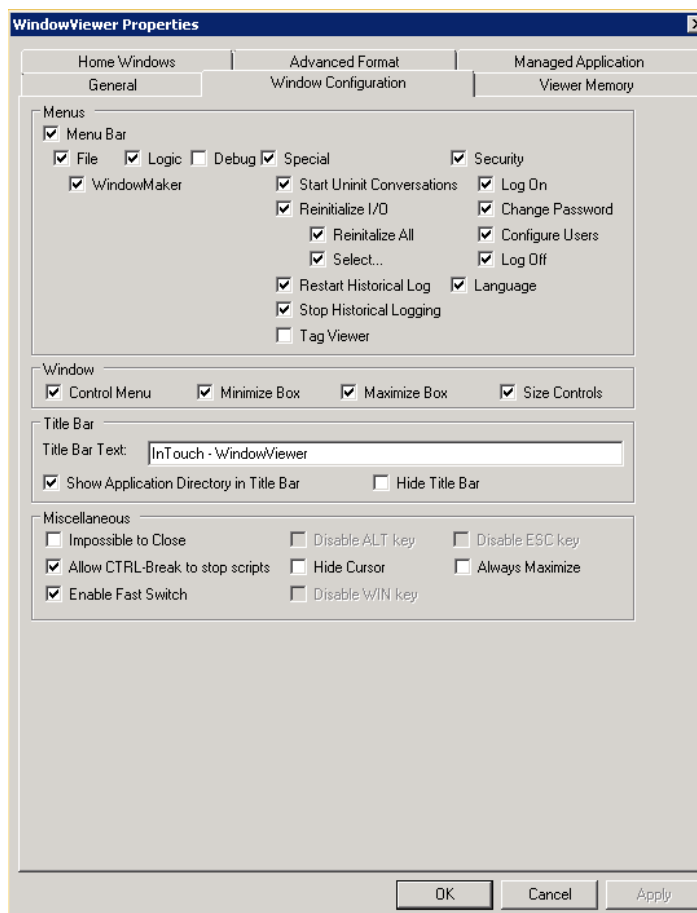
WindowViewer has key filter options that set the default state of system keys when an InTouch application starts. A key filter disables a system key when it is active.

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.

To enable key filters

- 1 On the **Special** menu, point to **Configure**, and then click WindowViewer. The WindowViewer **Properties** dialog box appears.
- 2 Click the **Window Configuration** tab.



- 3 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.

- 4 Click **OK**.
- 5 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)

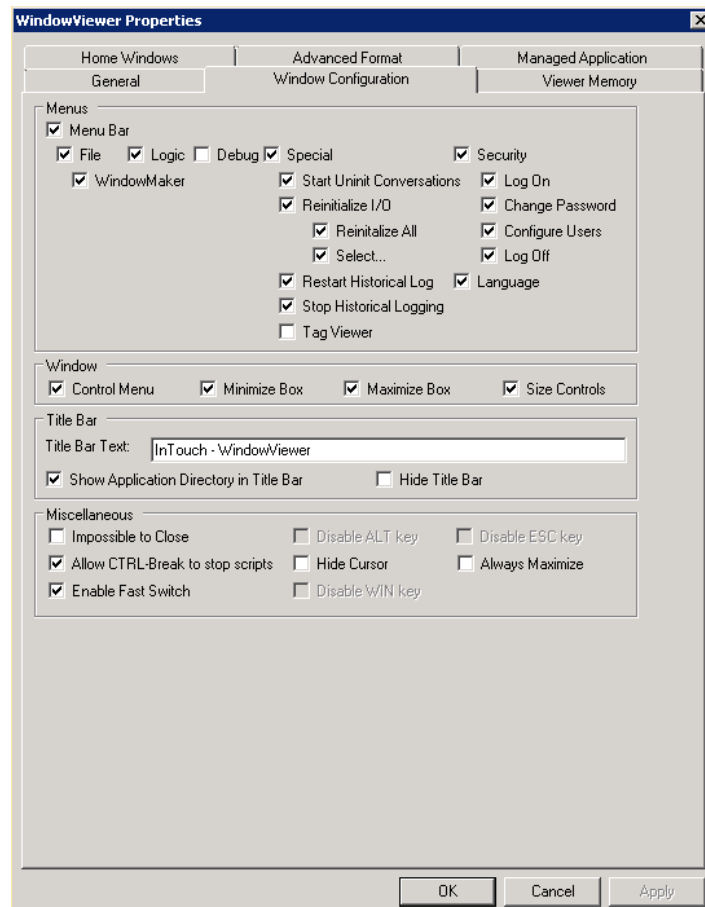
```
EnableDisableKeys(0,0,0); // enable all three keys
EnableDisableKeys(1,1,1); // disable all three keys
EnableDisableKeys(0,0,1); // enable Alt and Escape keys, disable
                             Windows key.
```

Hiding Menu Items at Run Time

You restrict operator access to WindowViewer menus and commands by hiding them while an InTouch application is running.

To hide menu items at run time

- 1 On the **Special** menu, point to **Configure**, and then click WindowViewer. The WindowViewer **Properties** dialog box appears.
- 2 Click the **Window Configuration** tab.



- 3 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 to 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.
- 4 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.

- 5 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.
- 6 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.
- 7 Click **OK**.
 - 8 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.

Comparing 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.

Using 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.

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.

When you create a new application, by default, the user name is set to Administrator with an access level of 9999, which allows access to all security commands. The default administrator password is wonderware. The maximum number of characters for a password is 29.

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.

Using 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** on the **Special** menu 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.

Using ArcestrA-based Security

When you configure a node to use ArcestrA 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.

ArcestrA 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 ArcestrA-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.

Using 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 "Using Secured and Verified Writes" on page 146.

Setting 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 ArcestrA OS security. The ArcestrA security can be either user-based or group-based. You configure ArcestrA security using the ArcestrA IDE. For more information, see the ArcestrA 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 "Enabling Smart Card Authentication in WindowMaker" on page 145.
- 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**.

Enabling Smart Card Authentication in WindowMaker

You must enable Smart Card authentication in WindowMaker before you can use the Smart Card reader for authentication.

To configure the Smart Card Option

- 1 Open WindowMaker.
- 2 On the **Special** menu, point to **Security**, point to **Select Security Type**, and then click **ArchestrA** or **OS**.

Note: If you click **ArchestrA**, be sure that you have configured ArchestrA OS security (OS user-based or OS group-based) using the ArchestrA IDE.

- 3 On the **Special** menu, click **Smart Card Authentication** so that a check mark appears. By default, this is not checked.

Logging 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.

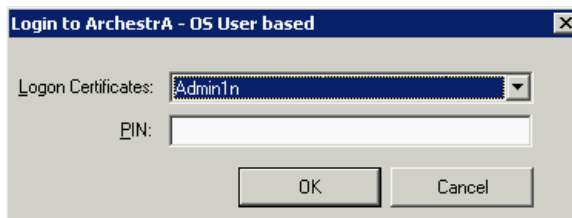
If a smart card is not detected in the reader when you try to log on, you are prompted to authenticate using your user name and password instead.

You can use smart card for authentication for secured and verified data writes. For more information, see "Using Secured and Verified Writes" on page 146.

To 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 click **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 Click **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.

Using 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.
- ArcestrA 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 ArcestrA attribute.

You can use smart cards for authentication for secured and verified data writes. For more information, see “Using Secured and Verified Writes” on page 146.

Performing a Secured Write

If you attempt to modify the value of an ArcestrA 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.

To 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 Click **OK**.



- 4 If you are authenticating using a smart card, the smart card options appear.

The **Secured Write** dialog box contains the following fields and controls:

- Reason Description:** A large text area for notes.
- Attribute:** A text box containing "DataUD0.SecUDA".
- Value:** A text box containing "37".
- Comment:** A dropdown menu showing "Select predefined comment..." and a larger text area below it.
- Mode:** A section with two icons: a person (selected) and a smart card.
- Certificate:** A dropdown menu.
- PIN:** A text box.
- Buttons:** "OK" and "Cancel" at the bottom right.

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** Click **OK**.

When a smart card is present, if you want to authenticate using your name, password, and domain instead, click the Mode button. Go to Step 3.

Performing a Verified Write

If you attempt to modify the value for an ArchestrA 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.

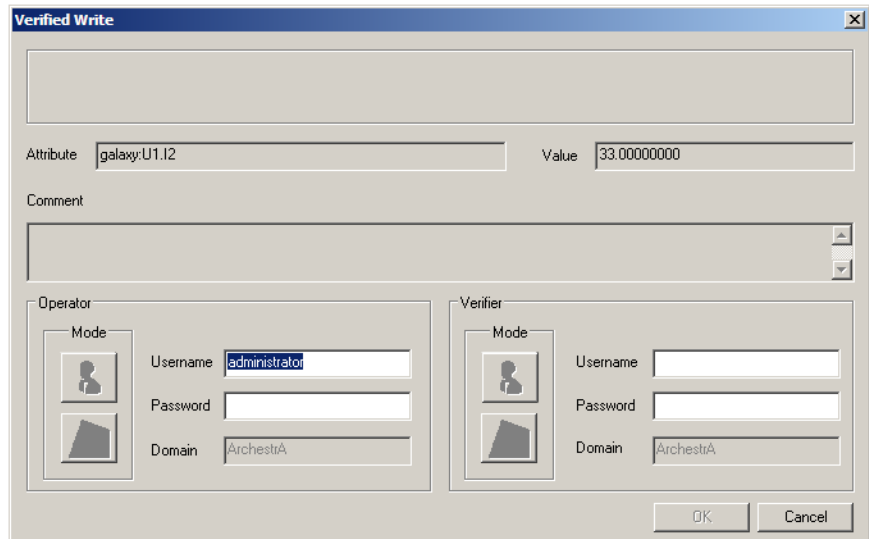
The smart card option is only available if a smart card reader is attached to the WindowViewer computer. You can use smart cards for logging in as either an operator, a verifier, or both, but the operator and the verifier must be two different people. If you previously logged on with your smart card, 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.

To 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.

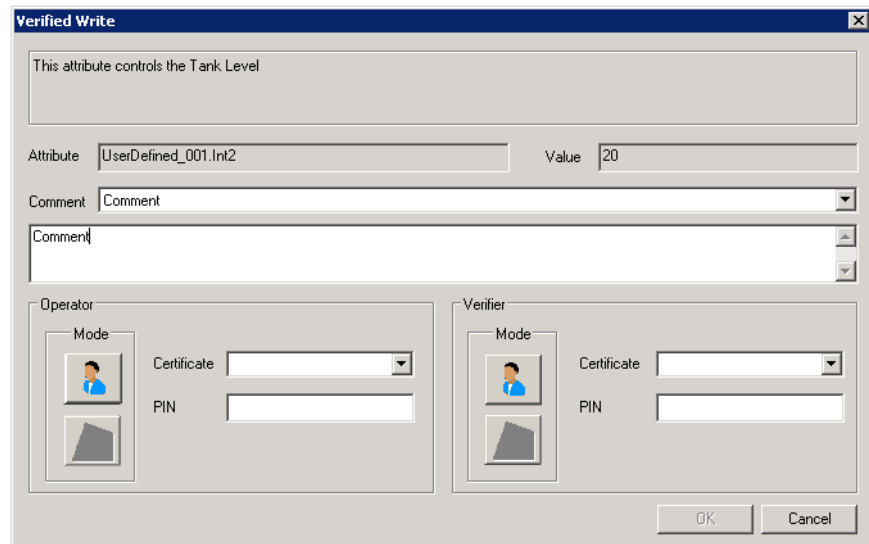
Do the following.

- 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 Click **OK**.



To Authenticate using a smart card instead, click 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 Click **OK**.

When a smart card is present, if you want to authenticate using your name, password, and domain instead, click the Mode button. Go to step 3.

Customizing 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 *Creating and Managing ArchestrA Graphics User's Guide*, Chapter 3 "Managing Symbols".

Working with the SignedWrite() Function at Run Time

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.

Managing 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.

Configuring InTouch Security Authentication and Authorization

For each of your operators, you need to assign a user name, password, and access level.

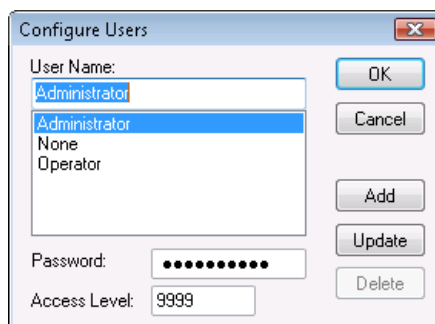
The **None** and **Administrator** names are reserved and only the password of the Administrator can be changed (the default is wonderware). After you configure user names for your application, change the Administrator password. The Administrator default access level (9999) is the highest and allows 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 clicks 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.

To configure security for operators of your application

- 1 On the WindowMaker **Special** menu, point to **Security**, and then click **Log On**.
- 2 Log on with your InTouch administrator account.
- 3 On the **Special** menu, point to **Security**, then click **Configure Users**. The **Configure Users** dialog box appears.



- 4 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 Click **Add** to add the user name to the InTouch security list.
- 5 To change a user name, select the name, make any changes, and then click **Update**.
- 6 To delete a user name, select the name and then click **Delete**.
- 7 Click **OK**.

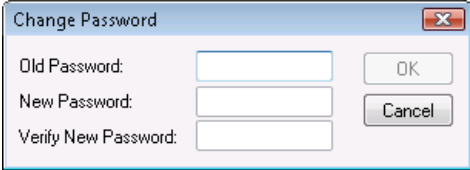
Changing an InTouch Operator Password at Run Time

Operators can change their passwords at run time using the **Special** menu in WindowViewer.

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.

To change an operator password

- 1 On the **Special** menu, point to **Security** and then click **Change Password**. The **Change Password** dialog box appears.

A screenshot of the 'Change Password' dialog box. It has a title bar with 'Change Password' and a close button (X). Inside, there are three text input fields: 'Old Password:', 'New Password:', and 'Verify New Password:'. To the right of the 'Old Password' field is an 'OK' button, and below it is a 'Cancel' button.

- 2 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.
- 3 Click **OK**.

Setting 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.

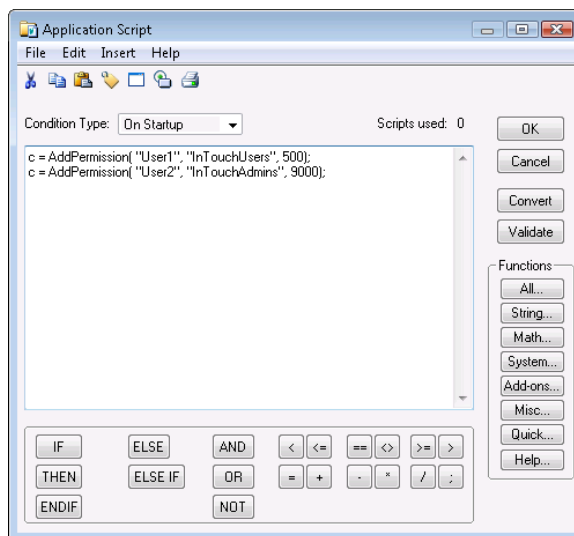
You then assign InTouch access levels to the Windows groups using the `AddPermission()` function in a script. The `AddPermission()` function is typically called on application start-up so `WindowViewer` recognizes all the authorized user groups when a user is ready to log on.

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.

To set operating system-based security and configure access levels

- 1 On the WindowMaker **Special** menu, point to **Security**, point to **Select Security Type**, and then click **OS**.
- 2 On the **Special** menu, point to **Scripts**, and then click **Application Scripts**. The **Application Script** dialog box appears.



- 3 In the **Condition Type** list, click **On Startup**.

- 4 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.
- 5 Click **OK**.

Setting 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 on the **Special...Security** menu are unavailable in WindowMaker.

To set ArchestrA-based security

- 1 Open a window in WindowMaker.
- 2 On the **Special** menu, point to **Security**, point to **Select Security Type**, and then click **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);
```

See Also

PostLogonDialog(), InvisibleVerifyCredentials(), IsAssignedRole(), AttemptInvisibleLogon(), QueryGroupMembership()

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

Logging On and Off

Logging on to and logging off from an InTouch application varies by the type of security used to protect an application.

Logging 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" on page 164.

To log on to an application

- 1 On the **Special** menu, point to **Security**, and then click **Log On**. The **Log On** dialog box appears.
- 2 In the **Name** box, type your user name.
- 3 In the **Password** box, type your password.
- 4 Click **OK**.

Logging 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 clicks **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.

Logging On to an ArcestrA-Secured Application

Users typically log on and log off from an ArcestrA-secured InTouch application by entering a valid user name and password.

If your InTouch application has been configured for the ArcestrA 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 ArcestrA authentication modes, such as “Galaxy”, “OS User based”, or “OS Group based”.

To log on

- 1 Start the ArcestrA-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.

Logging 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 "Configuring an Inactivity Time-Out" on page 132.

To log off from an application

- ◆ On the **Special** menu, point to **Security** and then click **Log Off**.

Creating 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.

To 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 clicks 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 ArcestraA security: the user must be a member of ArcestraA OS user-based or OS group-based security.
- InTouch configured with ArcestraA 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 ArcestraA client scripting.

Example

```
IntegerResult = LogonCurrentUser();
```

See Also

PostLogonDialog(), InvisibleVerifyCredentials(), IsAssignedRole(),
AttemptInvisibleLogon(), QueryGroupMembership(), AddPermission()

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

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 login.

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

Enabling and Disabling 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.

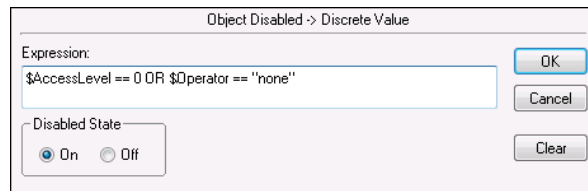
For example, to make an object become visible based on the access level of the logged on user, use the following statement in a visibility animation link expression:

```
$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 ArcestrA security is in use and if ArcestrA 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

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 ArcestrA security mode.

If the GetAccountStatus() function is used with ArcestrA security, the script attempts to retrieve the account information directly from the domain controller. This works as long as the ArcestrA 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 ArcestrA security.

Category

security

Syntax

```
DiscreteTag=IsAssignedRole( "RoleName" );
```

Arguments

RoleName

The role associated with an Application Server user.

Remarks

Valid for ArcestrA 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 ArcestrA security only when the ArcestrA 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 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	ArchestraA Security
\$AccessLevel	Yes	Yes	Yes
\$ChangePassword	Yes	Yes	Yes
\$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
ChangePassword()	Yes	No	No
EnableDisableKeys()	Yes	Yes	Yes

	InTouch Security	Operating System Security	Archestra Security
GetAccountStatus()	No	Yes	Yes*
InvisibleVerifyCredentials()	No	Yes	Yes*
IsAssignedRole()	No	No	Yes
Logoff()	Yes	Yes	Yes
LogonCurrentUser()	No	Yes	Yes*
PostLogonDialog()	Yes	Yes	Yes
QueryGroupMembership()	No	Yes	Yes*

* When Orchestra security is OS user or group based

Chapter 6

Switching a Language at Run Time

You can develop applications that can be switched to another language at run time.

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.

As part of the setup for run-time language switching, you can also localize alarm comments and alarm fields. 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.

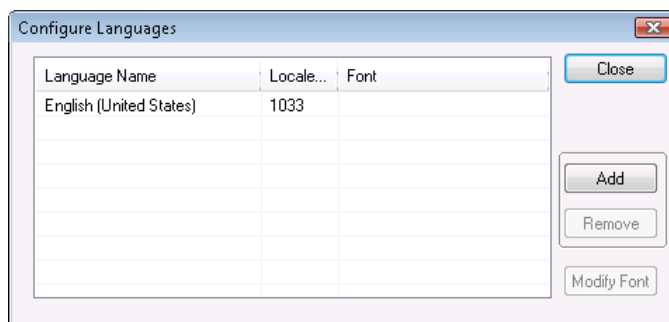
Configuring Languages for Run-time Language Switching

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 recommended 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.

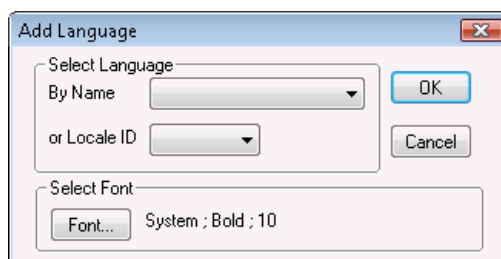
To configure languages for run-time language switching

- 1 In WindowMaker, open the application for which you want to configure languages.
- 2 On the **Special** menu, point to **Language**, and then click **Configure Languages**. The **Configure Languages** dialog box appears.

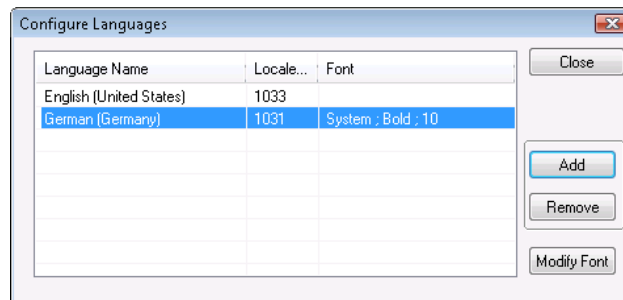


The **Configure Languages** dialog box shows the base language of the application.

- 3 Click **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, click the language to add. If you select the language by name, the corresponding locale ID appears in the **Locale ID** list, and vice versa.
 - Click **Font**. The **Font** dialog box opens. Configure the font and then click **OK**.
- 5 Click **OK** to close the **Add Language** dialog box. The language you configured is listed in the **Configure Languages** dialog box.



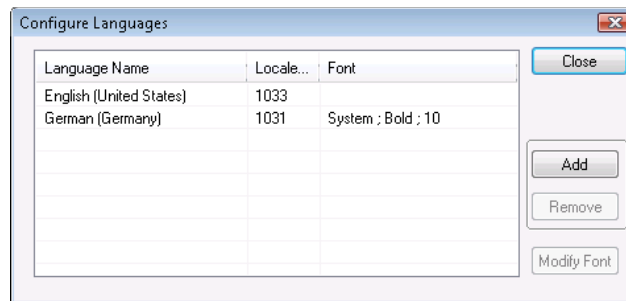
- 6 To add more languages, repeat steps 3 through 5.
- 7 When you are done, click **Close**.

Changing 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.

To 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 **Special** menu, point to **Language**, and then click **Configure Languages**. The **Configure Languages** dialog box appears.



- 3 In the list of languages, select the target language, and then click **Modify Font**. A standard Windows **Font** dialog box appears.
- 4 Make your changes and then click **OK**.
- 5 Click **OK** to close the **Configure Languages** dialog box.

Adding Run-Time Language Switching Functionality

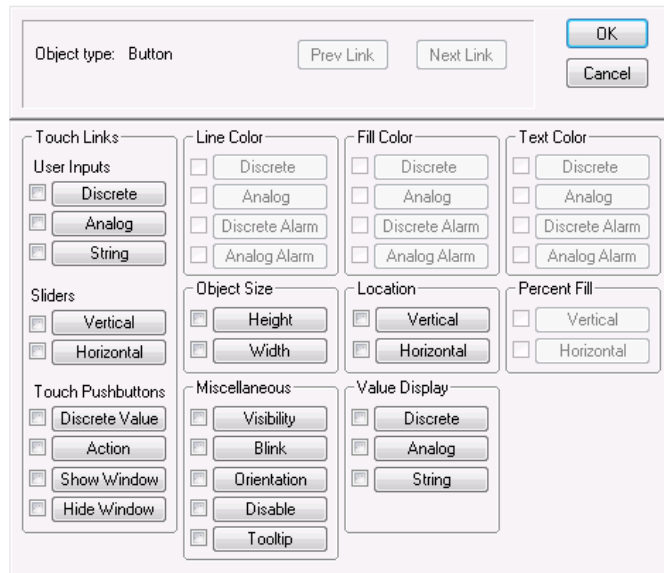
Run-time users can switch the language of an application interface by using the WindowViewer **Language** command on the **Special** menu.

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 Run-time Language Switching" on page 180.

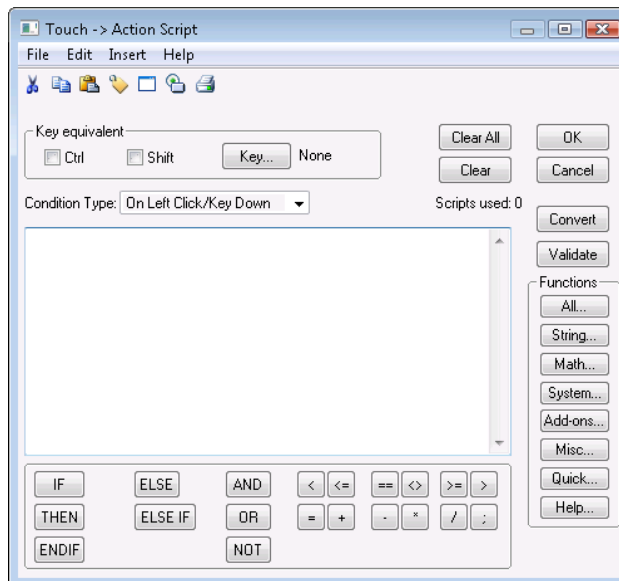
To add a button for switching languages at run time

- 1 In WindowMaker, open the application window that you want to add the language switching button.
- 2 In the window, draw a button.
- 3 Assign a text label to the button that indicates the language to be switched to when selected.

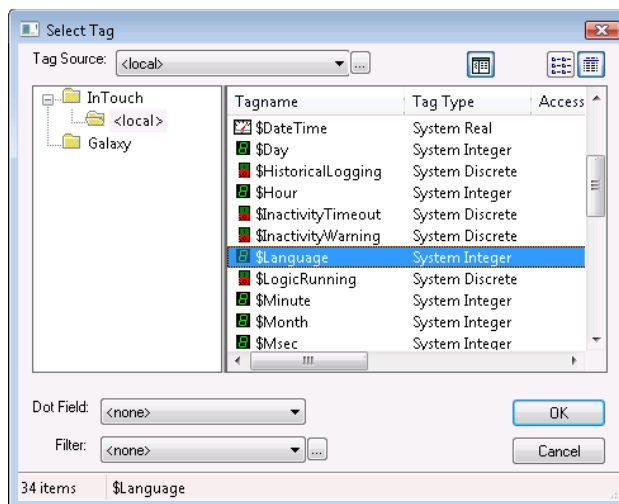
- 4 Double-click the button. The animation selection dialog box appears.



- 5 In the **Touch Pushbuttons** area, click **Action**. The **Touch -> Action Script** dialog box appears.

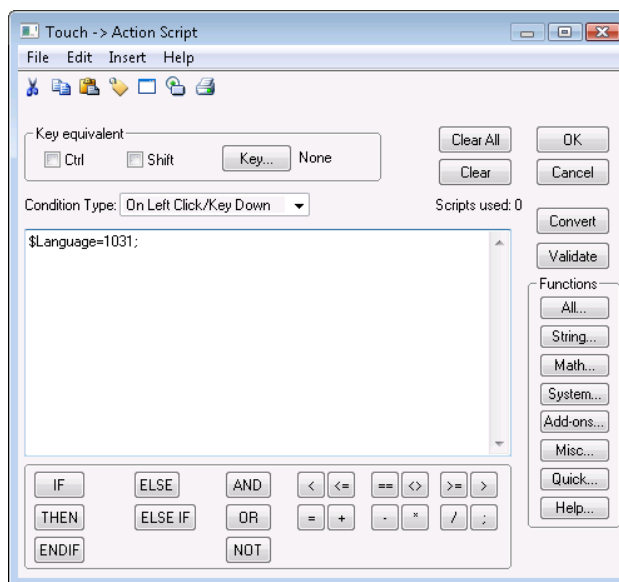


- 6 Double-click anywhere in the script area of the **Touch -> Action Script** dialog box. The **Select Tag** dialog box appears.



- 7 Click the **\$Language** system tag and then click **OK**.

Set the **\$Language** system tag equal to the locale ID of the language you are assigning to the button and click **OK**.



Note: You can also use the script function `SwitchDisplayLanguage(LocaleID)` instead of the `$Language` tag.

- 8 Click **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.

Category

system

Data Type

Integer (read / write)

Exporting 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 "Exporting Text to an Existing Dictionary File" on page 187.

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.

To 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 **Special** menu, point to **Language**, and then click **Export Dictionary**. The **Export Dictionary** dialog box appears.

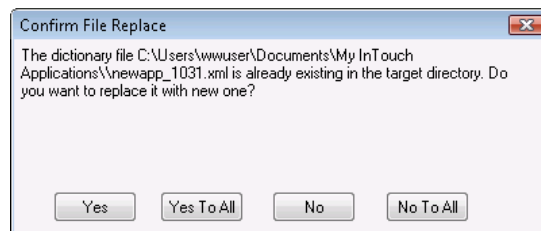


- 3 Configure the export settings.
 - In the **Defined Languages** list, click the language dictionary to export.
 - In the **Path** box, type the folder to which you want to export the dictionary. Click **Browse** to select an existing folder or create a new folder.
- 4 Click **Export**. The export progress is shown. If the export is successful, the **Export Successful** dialog box appears.
- 5 Click **Close** to return to the WindowMaker window or click **Close and Launch Explorer** to open the folder containing the dictionary files.

Exporting 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 "Exporting Application Text for Offline Translation" on page 186.

If you export more than one time to the same directory, the **Confirm File Replace** dialog box appears.



If you click **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.

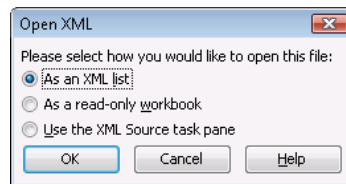
Translating an Exported Dictionary File

After you export the dictionary file containing your application text, use Microsoft Excel 2003 or later to edit the text.

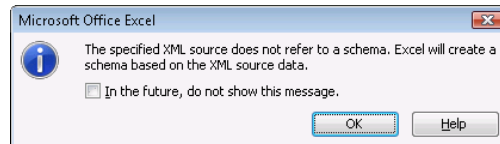
The InTouch HMI creates a separate dictionary file for each language that you export. The InTouch HMI also creates separate dictionary files for each SmartSymbol in your application. Be sure to translate all dictionary files for all languages and SmartSymbols.

To translate an exported dictionary file

- 1 Open the XML file in Excel. The **Open XML** dialog box appears.



- 2 Click **As an XML list**, then click **OK**. A message may appear.

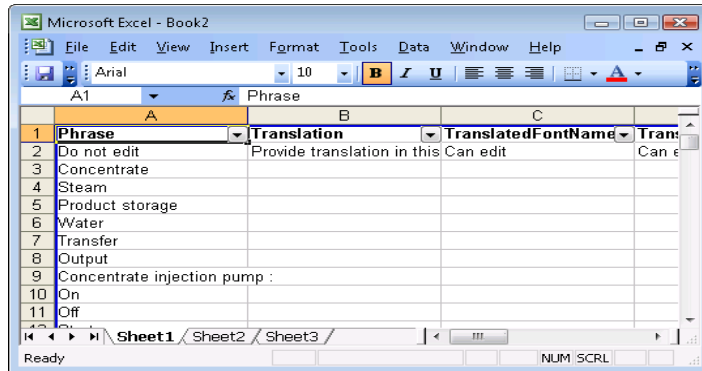


- 3 Click **OK**.

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:

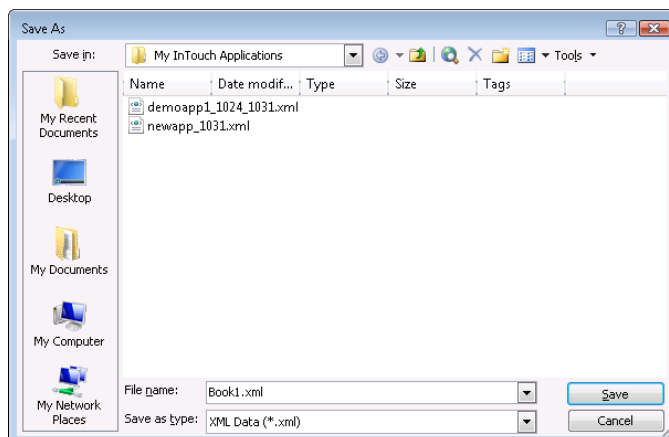
B = **bold**

I = *italic*

U = underline

For example, if you want 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.

Importing 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.

To import a translated dictionary file

- 1 Start WindowMaker and open the application to import translated dictionary files into.
- 2 On the **Special** menu, point to **Language**, and then click **Import Dictionary**. The **Import Dictionary** dialog box opens.



- 3 Configure the import settings.
 - In the **Defined Languages** list, click the language dictionary to import.
 - In the **Path** box, type the path to the dictionary file to import. Click **Browse** to browse and select the file.
- 4 Click **Import**.
- 5 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.

Exporting 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.

Understanding 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 >=.

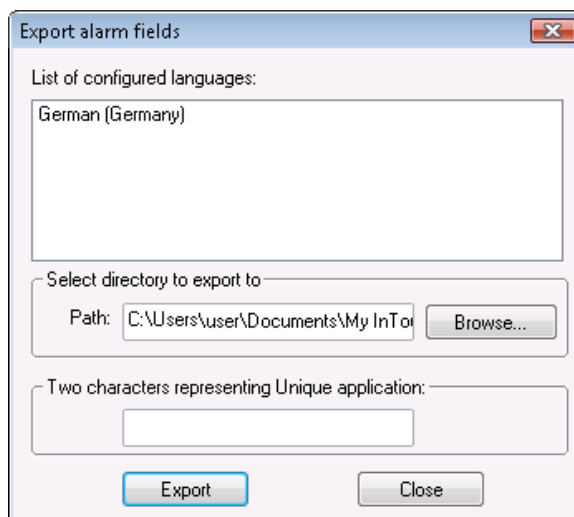
Exporting Alarm Comments

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.

To 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 **Special** menu, point to **Language**, and then click **Export Alarm Fields**. The **Export Alarm Fields** dialog box appears.



- 3 In the **Path** box, type the folder to which you want to export the dictionary. Click **Browse** to select an existing folder or create a new folder.
- 4 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.

- 5 Click **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.

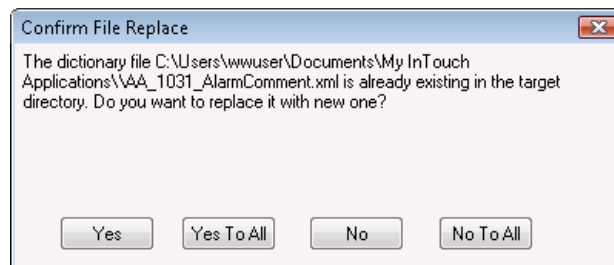
Note: If the size of the alarm comment configured in the tag dictionary is greater than 127 characters or the tag comment is greater than 46 characters, the alarm or tag comment is not exported. You are notified that the comment was not exported to the dictionary file at the end of the export process and an AlarmComment.log or TagComment.log file is created in the export directory.

- 6 Click **Close** to return to the WindowMaker window or click **Close and Launch Explorer** to open the folder containing the dictionary file.

Exporting 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 "Exporting Alarm Comments for Translation" on page 191.

If you export more than one time to the same directory, the **Confirm File Replace** dialog box appears.



Click **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.

Click **Yes to All** to update existing dictionary files for all languages configured in the InTouch HMI.

Click **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.

Editing 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
DSC	DSC
EVENT	EVENT
VALUE	VALUE

Importing 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.

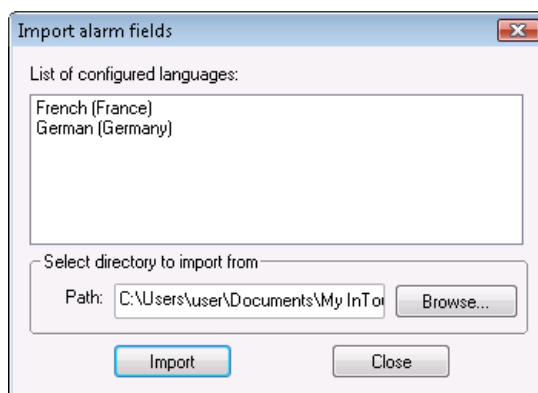
After you import the translated alarm comment dictionary files, they are copied into the respective language folders inside the application directory.

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.

To import a translated alarm comment file

- 1 Start WindowMaker and open the application to import translated dictionary files into.
- 2 On the **Special** menu, point to **Language**, and then click **Import Alarm Fields**. The **Import Alarm Fields** dialog box appears.



- 3 In the **Path** box, type the path to the dictionary file to import or click **Browse** to browse and select the file.
- 4 Click **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.
- 5 Click **OK**. If the import is successful, the **Import Successful** dialog box appears.

Testing the Language Switching Functionality at Run Time

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 50 characters for tag comments.

To test the language switching functionality

- 1** Open the application in WindowViewer.
- 2** On the **Special** menu, point to **Language**, and then click 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, click the button to test the script.

Distributing 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.

Chapter 7

Viewing Applications at Run Time

You use WindowViewer to run your InTouch applications. Applications that are designed specifically for use in an Archestra Application Server environment are called InTouchView applications. These applications run in WindowViewer, but the Application Server provides most of the HMI functionality.

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.

Customizing Your Run time Environment

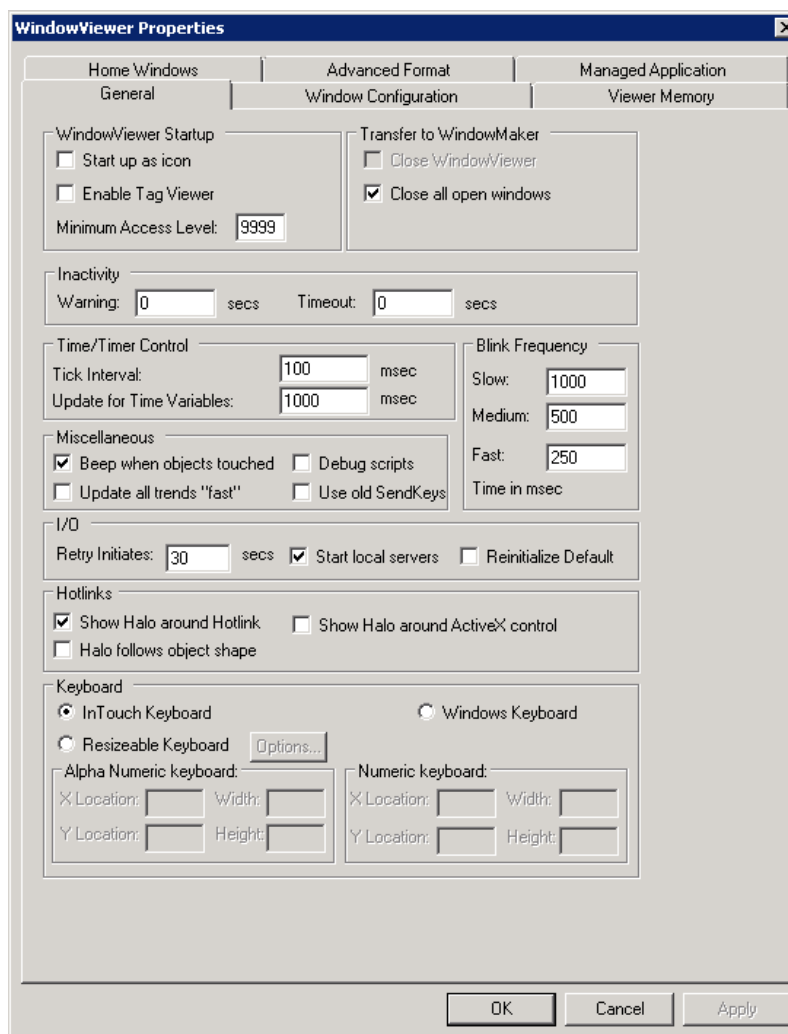
You can set properties to customize your run time WindowViewer environment.

- 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.

Configuring 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.

- 1 Open WindowMaker.
- 2 On the **Special** menu, point to **Configure**, and then click WindowViewer. The WindowViewer **Properties** dialog box appears.
- 3 Click the **General** tab.



4 In the **WindowViewer Startup** area, 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 the Enabling Tag Viewer section in the *InTouch® HMI Tag Viewer Guide*.
- 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 the Enabling Tag Viewer section in the *InTouch® HMI Tag Viewer Guide*.

5 In the **Transfer to WindowMaker** area, do the following:

- Select the **Close WindowViewer** 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 **Properties/General** tab 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.

6 In the **Inactivity** area, set warning and time-out periods for operator inactivity.

For more information about setting warning and time-out periods, see "Configuring an Inactivity Time-Out" on page 132.

7 In the **Time/Timer Control** 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.

8 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 **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 **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 the **Use old SendKeys** check box if you are using an international application developed with InTouch version 3.26 or earlier.

9 In the **Blink Frequency** area, type the interval length in milliseconds for your **Slow**, **Medium**, and **Fast** blink animation links.

10 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.

11 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.
- 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.
- If you want a halo around Active X controls, select the **Show halo around Active X control** check box.

12 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 Animating Objects in the *InTouch® HMI Visualization Guide*.

13 Click **OK**.

Configuring Visual Characteristics of WindowViewer

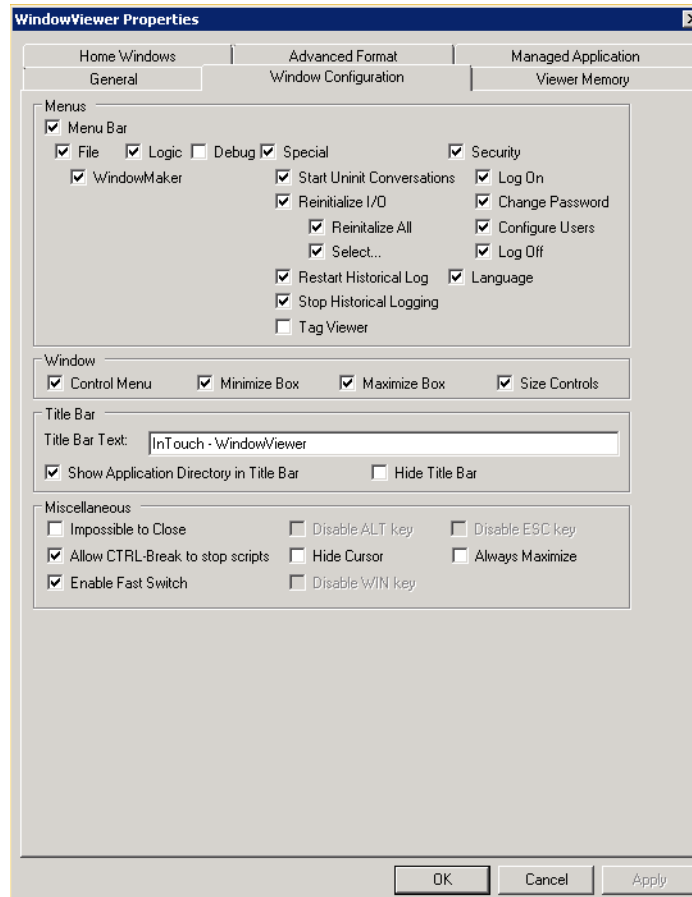
You can configure properties that determine the visual characteristics of WindowViewer as it runs an InTouch application. These properties determine the menus, commands, and standard controls that appear on a WindowViewer window.

To configure WindowViewer visual characteristics

- 1** Open WindowMaker.
- 2** On the **Special** menu, point to **Configure**, and then click WindowViewer. The WindowViewer **Properties** dialog box appears.

3 Click the **Window Configuration** tab.

Select the check boxes for the visual characteristics.

**4** Restart WindowViewer.

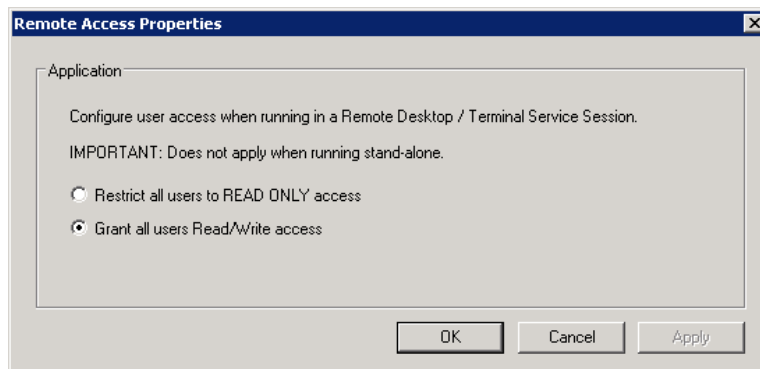
Configuring User Access to Applications Running in Remote Sessions

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.

To configure user access to applications running on remote nodes

- 1 Open the application in WindowMaker.
- 2 Click **Special**, and then click **Configure** to show a list of configuration options.
- 3 Click **Remote Access** from the list of configuration options.

The **Remote Access Properties** dialog box 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 click **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 ArchestrA graphics is available only in Managed or Published InTouch applications. This capability is disabled in Native 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 the *Creating and Managing ArchestrA Graphics User's Guide*.

Configuring Memory Usage for WindowViewer Windows

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

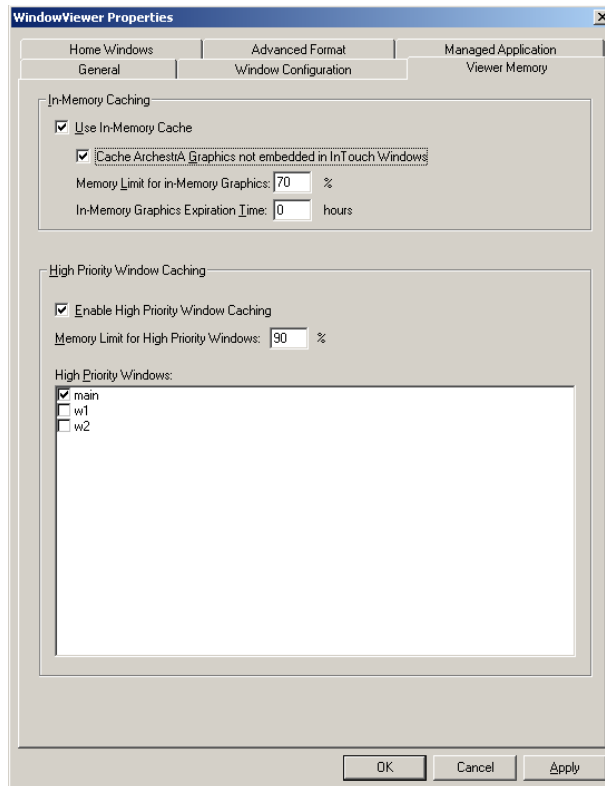
Reopening closed windows that have been cached retrieves them from memory rather than loading them from disk in certain conditions.

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.

To set the memory properties

- 1 On the **Special** menu, point to **Configure** and then click **WindowViewer**. The **WindowViewer Properties** dialog box appears.
- 2 Click the **Viewer Memory** tab.



- 3 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 ArchestrA Graphics not embedded in InTouch Windows** check box to enable ArchestrA graphics symbol caching.

InTouch windows and ArchestrA graphics will share the in-memory cache.

If the memory limit set in step 3c is exceeded, the system automatically removes the oldest in-memory graphic from the cache.

- c** 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.

- d** 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.

The memory limit or the expiration time limit is applied depending on which limit is reached first.

4 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.

5 Click **OK**.

Configuring 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 "Configuring Memory Usage for WindowViewer Windows" on page 206.

To 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.

Configuring 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 SMC 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:

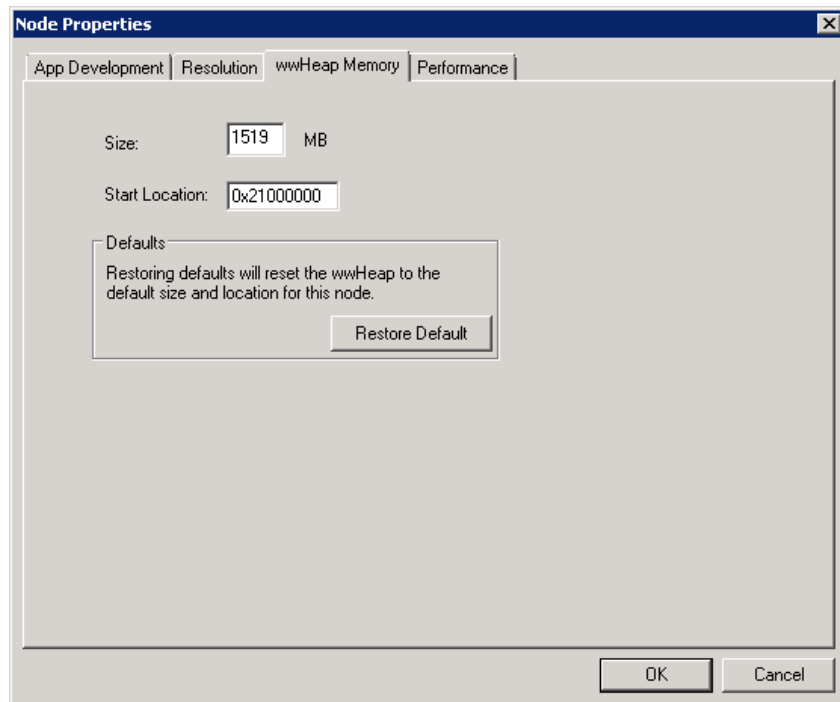
Operating System	Default Size
32-bit	1519 MB
32-bit with the /3GB switch enabled	2048 MB
64-bit	2048 MB

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 0x7FFFEFFF
32-bit with the /3GB switch enabled	0x40000000	0x00010000 to 0xBFFFEFFF
64-bit	0x80000000	0x00010000 to 0xFFFEFFFF

To configure wwHeap Memory settings

- 1 Start **Application Manager**.
- 2 On the **Tools** menu, click **Node Properties**. The **Node Properties** dialog box appears.
- 3 Click the **wwHeap Memory** 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, click **Restore Default**.
- 6 Click **OK**.

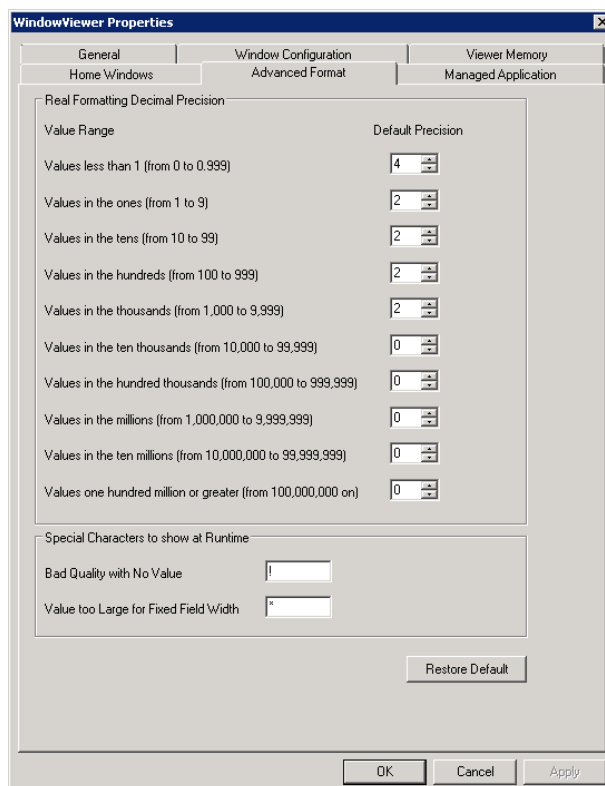
Setting Advanced Formatting Properties

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 also 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.

To set the advanced formatting properties

- 1 On the **Special** menu, point to **Configure** and then click **WindowViewer**. The **WindowViewer Properties** dialog box appears.
- 2 Click the **Advanced Format** tab.



- 3 In the **Real Formatting Decimal Precision** area, enter the number of decimal places that you want to be shown at run time for each real type number range.

- 4 In the **Special Characters to Show at Runtime** area, 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.
- 5 To restore the default values, click **Restore Default**

Configuring 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.

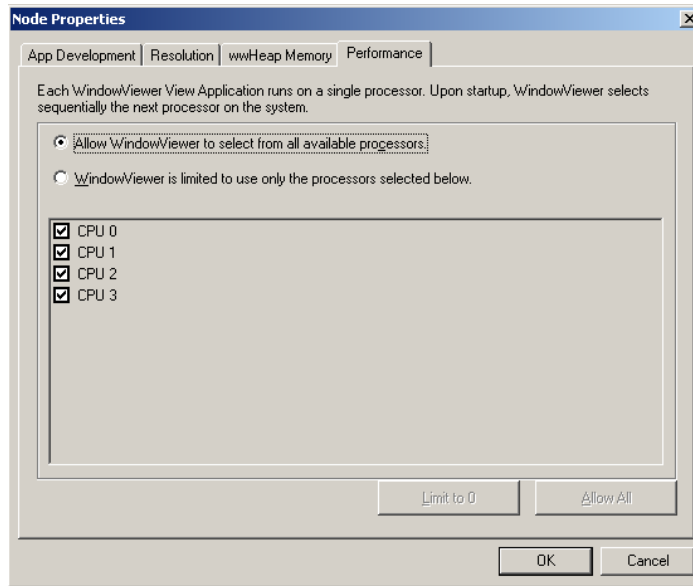
When WindowViewer starts, the system checks if the computer has multiple processors and which processors are allowed to run WindowViewer instances. WindowViewer then checks the processors sequentially and starts running on the processor that has the least number of View instances running on it.

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.

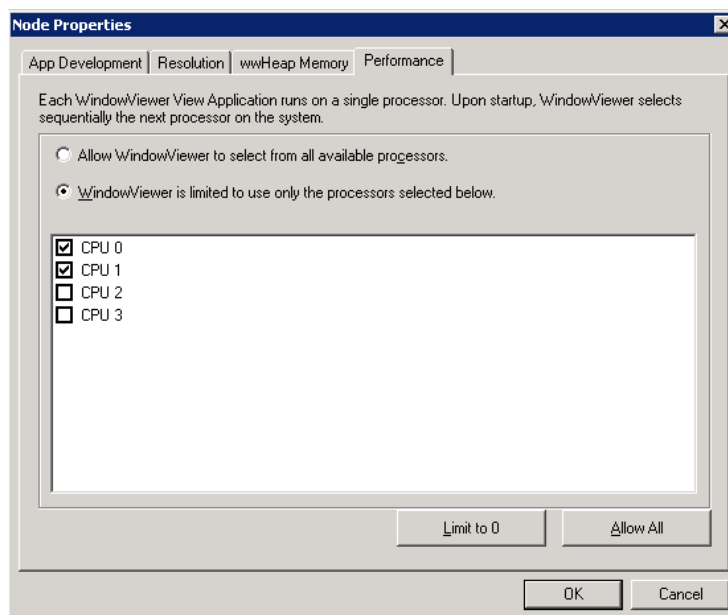
To configure the processor “pool”

- 1 Start **Application Manager**.
- 2 On the **Tools** menu, click **Node Properties**. The **Node Properties** dialog box appears.



- 3 Click the **Performance** tab.
- 4 To allow WindowViewer to use any available processor, click **Allow WindowViewer to select from all available processors**.
- 5 To restrict the processors that WindowViewer can use, click **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.
 - Click **Limit to 0** to only allow WindowViewer to run on processor 0. When you click this button, the **CPU 0** check box is automatically selected.

- Click **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 Click **OK**. WindowViewer starts on the next CPU based on the other View sessions.

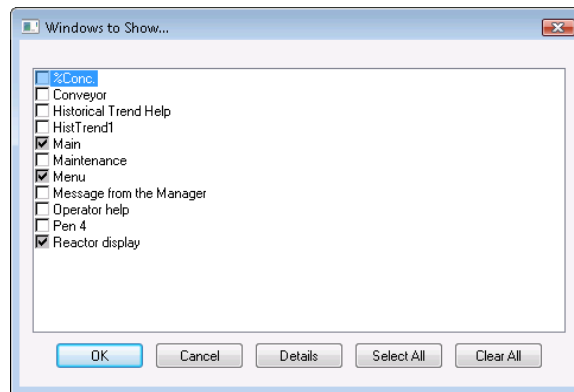
Working with WindowViewer Windows

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 Configuration** tab on the WindowViewer **Properties** dialog box, operators can run standard commands from the WindowViewer **File** menu to open and close windows.

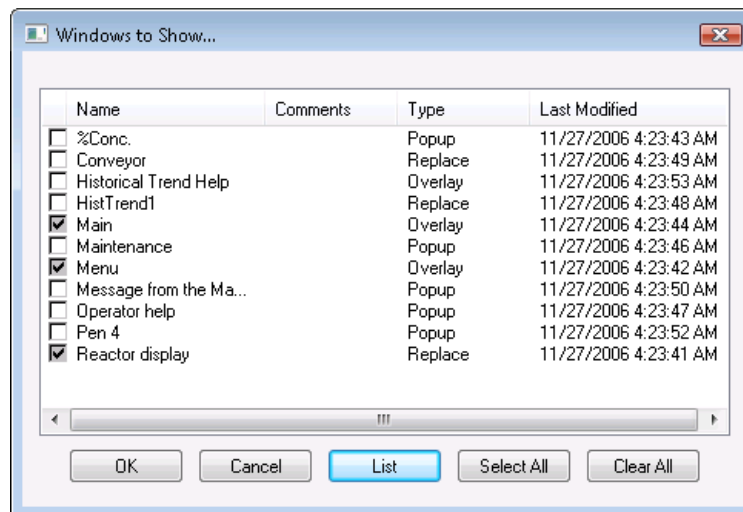
Common Dialog Box Features

If you configured WindowViewer to show the **File** menu, operators can open or close InTouch application windows. When operators click 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** dialog box appears after clicking on the **Open Window** command.



Click **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 clicking on any portion of its row, not just the check box. The entire row is highlighted when selected.

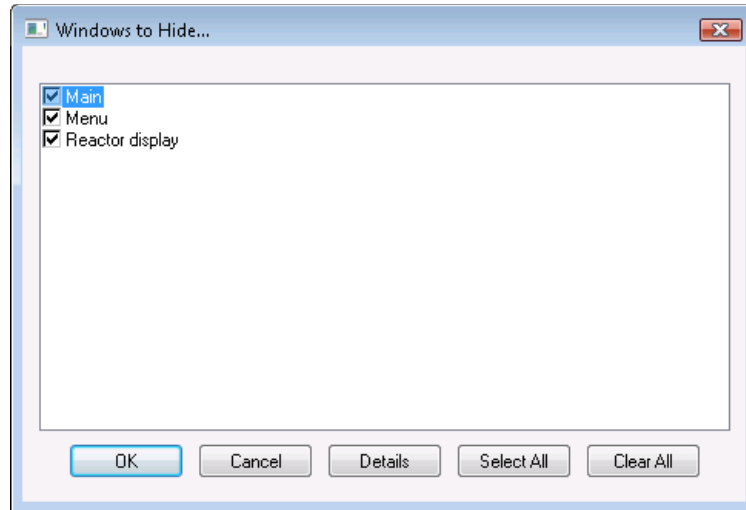
- To open selected windows click **OK**.
- To cancel your selections and close the dialog box, click **Cancel**.
- To return the dialog box to list view, click **List**.
- To select all listed windows, click **Select All**.
- To clear all selected windows, click **Clear All**.
- To sort the list in ascending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

Opening Windows from WindowViewer

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

To open windows from WindowViewer

- 1 On the **File** menu, click **Open Window**. The **Windows to Show** dialog box appears.



- 2 Click the check box next to the name of each window that you want to open.
- 3 Click **OK** to close the dialog box and open windows you selected.

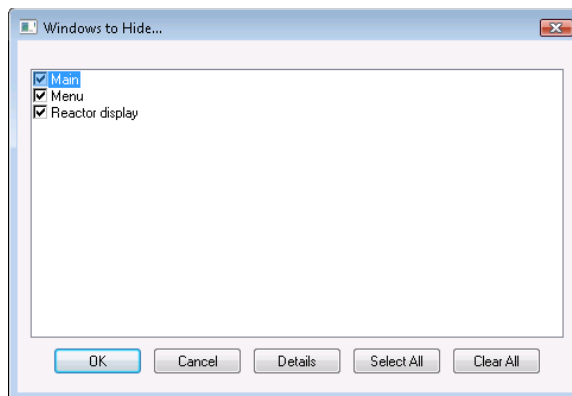
Note: If a “Replace” type window is selected, it closes any windows that it intersects.

Closing Windows from WindowViewer

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

To close open windows

- 1 On the **File** menu, click **Close Window**. The **Windows to Hide** dialog box appears.



- 2 Click the check box next to the name of one or more windows that you want to close.
- 3 Click **OK** to close the windows you selected.

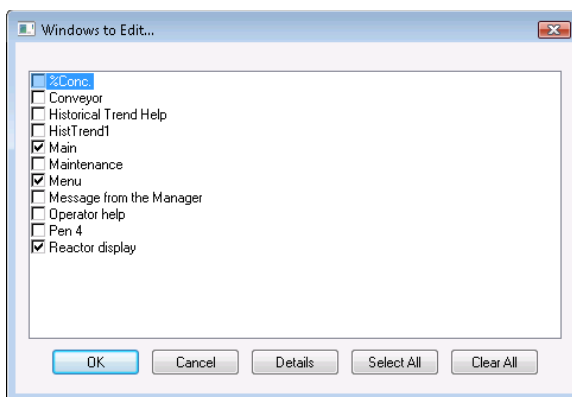
Transferring from WindowViewer to WindowMaker

When you develop an InTouch application, you can easily transfer between WindowMaker and WindowViewer by clicking 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.

To transfer from WindowViewer to WindowMaker

- 1 On the **File** menu, click WindowMaker. The **Windows to Edit** dialog box appears.



- 2 Click the check box next to the name of each window that you want to open when you transfer to WindowMaker.
- 3 Click **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.

About InTouchView Applications

InTouchView applications show visual interfaces designed specifically for use in an ArchemstrA Application Server environment. InTouchView applications run in WindowViewer, with Application Server providing most of the 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 ArchemstrA Application Server Galaxy.
- Cannot generate alarms. However, you can display and acknowledge alarms from remote alarm providers, such as ArchemstrA objects.
- Do not log application data or events. An InTouchView application generates only SYS and USER-related events.
- Can be secured only with ArchemstrA security.
- Cannot act as a server to other InTouch applications, InTouchView applications, or clients such as WWClient.

You develop an InTouchView application with WindowMaker. The following lists show which WindowMaker commands and Tagname Dictionary options are unavailable when creating InTouchView applications.

- Unavailable **Special** menu 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**

You run an InTouchView application as you would an InTouch application. Simply start the application in WindowViewer.

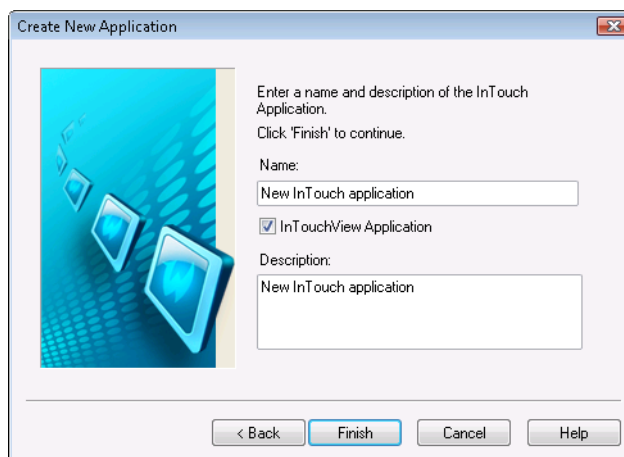
Creating a New InTouchView Application

You identify an InTouchView application by setting an option from Application Manager when you create the application.

To create a new InTouchView application

- 1** Start Application Manager.
- 2** On the **File** menu, click **New**. The **Create New Application** wizard appears.

- 3 Follow the steps of the **Create New Application** wizard. The wizard includes a dialog box with options to enter the application name, provide a description, and specify it as an InTouchView application.



- 4 Type the name and a description of the InTouchView application.
- 5 Select the InTouchView **Application** check box.
- 6 Click **Finish**. The system creates a new InTouchView application and sets the security to ArchestrA mode.

Converting an Application Between InTouch and InTouchView

You can convert an InTouch application to an InTouchView application and vice versa. A full InTouch license is required to run applications converted from InTouchView to InTouch. You can convert an InTouch application to InTouchView and still run the application with the InTouch license.

You cannot convert an application that is currently running in WindowViewer.

Converting an InTouchView Application to an InTouch Application

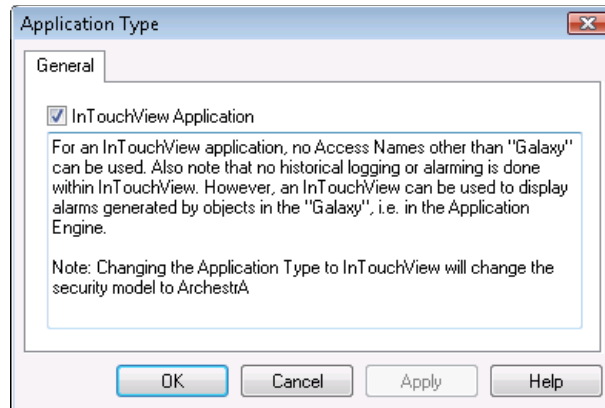
You can convert an InTouchView application to an InTouch application. For example, you would do this if you need the application to begin accessing tags from data sources other than the Application Server Galaxy.

After you convert the application, you can access all of the WindowMaker functions that are unavailable for an InTouchView application.

The converted application still uses ArchestrA security, but you can change the security type.

To convert an InTouchView application to an InTouch application

- 1 Open the InTouchView application in WindowMaker.
- 2 On the **Special** menu, click **Application Type**. The **Application Type** dialog box appears with the InTouchView **Application** box selected.



- 3 Clear the InTouchView **Application** check box and click **OK**.

To change the security type

- 1 Open the InTouch application in WindowMaker.
- 2 On the **Special** menu, point to **Security**, and then point to **Select Security Type** to show a list of supported InTouch security types.
- 3 Select the type of security for your InTouch application.

For more information about security, see "Securing InTouch" on page 131.

Converting an InTouch Application to an InTouchView Application

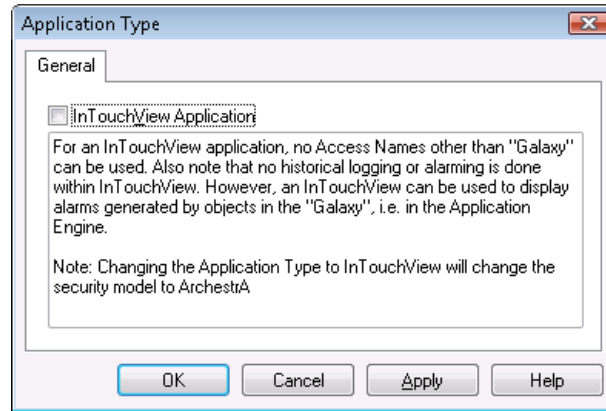
You can convert an InTouch application to InTouchView if the application only needs to connect to an Application Server.

After you convert the application to InTouchView, some WindowMaker functions are unavailable.

Important: 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.

To convert an InTouch application to an InTouchView application

- 1 Open the InTouch application in WindowMaker.
- 2 On the **Special** menu, click **Application Type**. The **Application Type** dialog box appears.



- 3 Select the InTouchView **Application** check box.
- 4 Click **OK**. When a message appears, click **OK**.

InTouchView Licensing

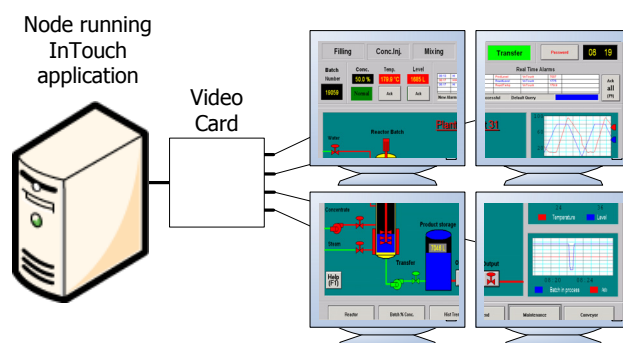
When an InTouchView application starts, WindowViewer requests a specific InTouchView license from the License Manager. If an InTouchView license is not available, License Manager requests a full-featured InTouch license. If the InTouch line is successfully read, WindowViewer starts the application. Otherwise, WindowViewer indicates the InTouchView license is unavailable and gives a brief description of the problem. WindowViewer offers options to quit, retry, or run the InTouchView application in demonstration mode.

Chapter 8

Setting 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

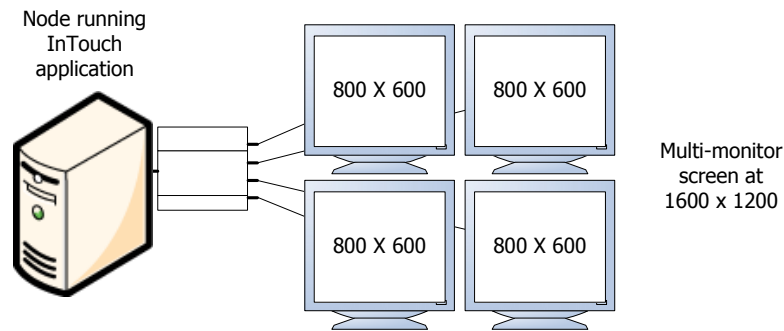
You can use two basic multi-monitor configurations.

- 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.

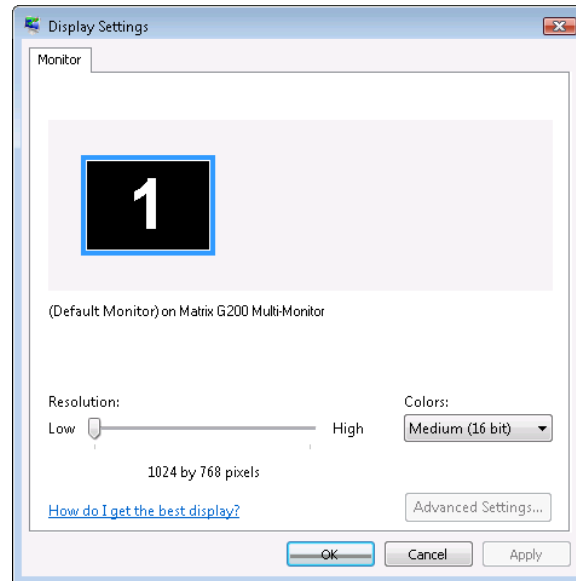
Characteristics of a Single Card Configuration

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

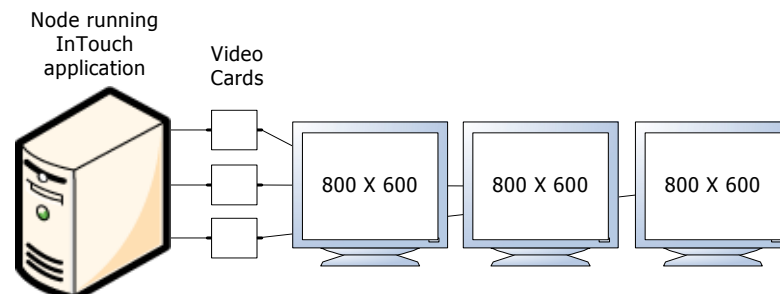
The following figure shows the Windows **Display Properties** dialog box to configure the driver for all monitors connected to a single video card with multiple output ports.



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

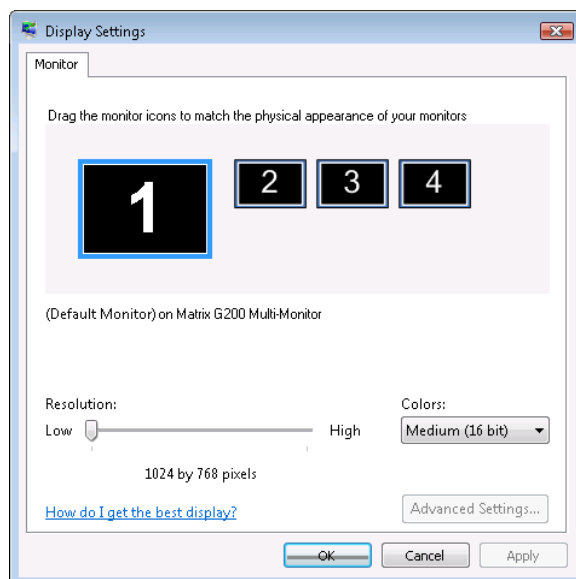
Dynamic Resolution Conversion (DRC) works with other distributed features to provide independence from screen resolution restrictions. In a NAD architecture, you create and maintain an InTouch application on a development node and then copy it to several View nodes. DRC allows all view nodes to show 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.



You click 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.

Planning a Multi-Monitor Application

To set up multiple monitors for your application, you must:

- 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

Choosing a Multi-Monitor Video Card

Wonderware 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?

Determining 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.

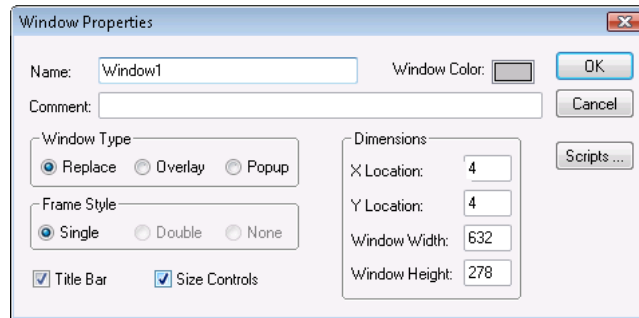
Create a drawing that shows the overall monitor configuration. The drawing should show the resolution of each monitor and the combined resolution of all the monitors together. This drawing helps you visualize the horizontal and vertical pixel range for each monitor.

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.

Determining 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 **Window Properties** dialog box to modify the characteristics of a window. You right-click on the name of the window listed in WindowMaker's **Windows & Scripts** pane, and then click **Properties** in the shortcut menu to show the **Window Properties** dialog box.



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.

Determining the Placement of Application Windows

You can use several different configurations when developing InTouch windows for a multi-monitor environment.

Windows Show 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 Manually Moved

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.

Developing 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.

Configuring Multi-Monitor Parameters

To enable multi-monitor support, you 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.

To 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
```

Configuring 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.

To 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.

Deploying the Application and Verifying 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. Simply click **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.

Verifying Multi-Monitor Support During Run Time

You can download an optional script function from the Wonderware 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.

Chapter 9

Using InTouch on a Tablet PC

Wonderware offers a line of portable Tablet PCs with Windows XP Tablet PC Edition and InTouch pre-installed. 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 Wonderware 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.

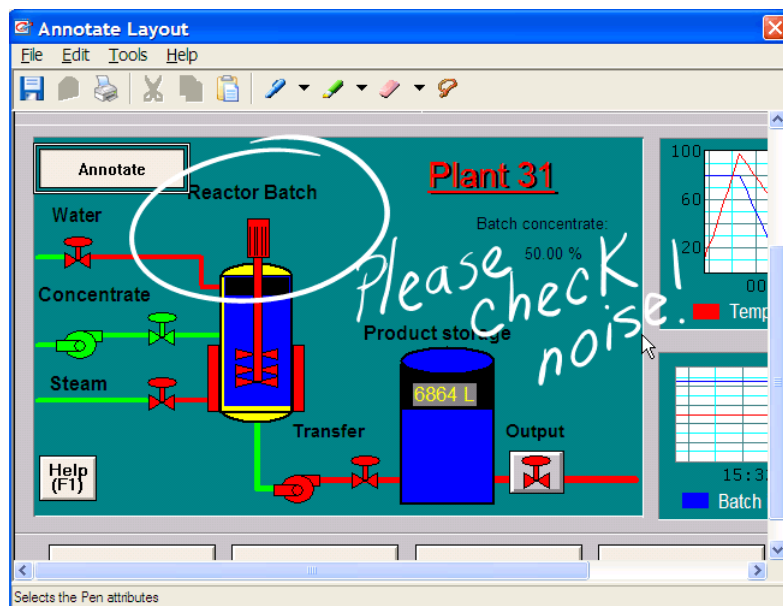


Operators can write notes and annotate a running InTouch application with direct observations about their actual plant processes.

Annotating and Sending 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.

Making Window Annotations

To make annotations to the window, use 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, click the downward arrow next to each tool's icon and then click the command for the option.
- To restore these options to their default settings, on the **Tools** menu, click **Restore Defaults**.

Selecting, Copying, and Deleting Window Annotations

You can select, copy, and delete annotations that you make in the window.

To select annotations



- 1 Click 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.

To cut, copy, and paste annotations

- ◆ Use the standard Windows Cut, Copy, and Paste commands.

To delete annotations

- ◆ Do any of the following:
 - To delete all annotations on a window, on the **Edit** menu, point to **Clear** and then click **All**.
 - To delete annotations that you selected using the lasso, on the **Edit** menu, point to **Clear** and then tap **Selection**.

Saving, Printing, and E-Mailing 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.

To save an annotated window

- 1 On the **File** menu, click **Save**. A standard Windows **Save As** dialog box appears.
- 2 Enter a name and format for the file and click **OK**.

To print an annotated window

- 1 On the **File** menu, click **Print**. A standard Windows Print dialog box appears.
- 2 Specify any printing options and click **OK**.

To send an annotated window as an e-mail attachment

- 1 On the **Edit** menu, click **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. Click **OK**.
- 3 On the **File** menu, click **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 Click **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).

Changing 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.

Appendix A

Customizing 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.
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.
LoopTimeOut	Sets the time out period of FOR-NEXT loop processing in an InTouch script.
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.

Setting 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.

Setting 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.

You 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
```

Logging 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
```

Disabling 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
```

Setting 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.

Adding 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
```

Scaling 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
```

Setting 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
```

Setting 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.

Resizing 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.

Resizing 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.

Resolving 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.

To solve these problems, do the following:

- 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.

Index

Symbols

\$AccessLevel system tag 159, 164, 169, 176
 \$ApplicationChanged system tag 49
 \$ApplicationVersion system tag 50
 \$ChangePassword system tag 155, 160, 176
 \$ConfigureUsers system tag 154, 161, 164, 176
 \$InactivityTimeout system tag 132, 134, 176
 \$InactivityWarning system tag 132, 134, 176
 \$Language system tag 184, 185
 \$LogicRunning system tag 139
 \$Operator system tag 59, 164, 169, 175, 176
 \$OperatorDomain system tag 174–175, 176
 \$OperatorDomainEntered system tag 169, 176
 \$OperatorEntered system tag 164, 167, 176
 \$OperatorName system tag 174, 176
 \$PasswordEntered system tag 164, 168, 176
 \$VerifiedUserName system tag 175, 176

Numerics

16PenTrendDrawMode parameter 244

A

Access Name

advising only active items 71
 restrictions with InTouchView applications 35
 selecting to convert placeholder tags 118
 selecting to convert placeholder tags to remote references 127

Access Names

specifying in DBLoad file 82
 AddPermission() function 142, 157, 176

AlarmGroup keyword 84–86

AnnotateLayout() function 234, 236

Application Manager

creating an InTouch application 18–20
 customizing the window 22
 finding applications 22–23
 modifying an application 21
 opening an application with WindowMaker 20
 opening an application with WindowViewer 20
 starting 16
 starting the ArchestrA IDE 17

Application Publisher

configuring package details 27
 contents of published file 25
 description 23

- package information 24
 - applications
 - configuring for run-time language switching 180–181
 - creating 18–20
 - creating a backup copy before migrating an application 29
 - creating for InTouchView 219–220
 - default folder 16
 - deleting from Application Manager 22
 - determining screen resolution for a multi-monitor system 227
 - editing lock 52
 - exporting alarm comments for translation 191–193
 - exporting alarm comments to an existing alarm comment file 193–194
 - exporting text for offline translation 186–187
 - exporting text to an existing dictionary file 187
 - extending overview 14
 - finding 22
 - importing translated alarm comments 196
 - logging on or off 162–163
 - logging remote referenced tags 241
 - managed 15
 - migrating to the current version of InTouch 29
 - modifying 21
 - opening in WindowMaker 20
 - opening in WindowViewer 20
 - publishing to remote nodes 23–28
 - renaming 21
 - scaling windows to different screen resolutions 243
 - securing in a Terminal Services environment 59
 - set mouse behavior 242
 - setting logging frequency 241
 - setting the length of the print waiting period 243
 - task management overview 13
 - Archestra
 - logging on to secured applications 163
 - managing InTouch applications through the IDE 15
 - master user account 68
 - setting up security 157
 - starting the IDE from the Application Manager 17
 - architectures
 - client-based 32
 - Network Application Development 34
 - server-based 33
 - AttemptInvisibleLogon() function 166, 176
 - authentication
 - using Archestra-based security 143
 - using InTouch-based security 141
 - using operating system-based security 142
 - authorization
 - restricting access to InTouch functionality 169
 - setting up for InTouch security 154–155
 - setting up for operating system-based security 156–157
- ## B
- beep sound for objects 202
 - blinking speeds 202
- ## C
- ChangePassword() function 158, 176
 - client-based architecture 32
 - closing WindowViewer on transfer to WindowMaker 201
 - closing/opening windows 216
 - commands
 - hiding in WindowViewer 139
 - Net Start view 68
 - Net Stop view 68
 - Notify Clients 34, 51
 - RESET 112
 - Terminal Services Client 57
 - unavailable from InTouchView 219
 - CommentRetentive parameter 243
 - core affinity 212
- ## D
- DBDump
 - description 74
 - viewing contents of exported file 75
 - DBLoad
 - AlarmGroup keyword 84–86
 - creating input file 77
 - creating SuperTag instances 113
 - description 74

input file 76
 IOAccess keyword 82, 82–84
 mode keyword 79–82
 specifying Access Names 82–84
 specifying alarm groups 84–86
 debugging scripts 202
 Distributed History System 40
 documentation conventions 11
 DRC, *See* Dynamic Resolution Conversion
 Dynamic Resolution Conversion 53
 configuring an application 54
 description 53

E

EnableDisableKeys() function 135, 137, 176
 event logs
 logging remote referenced tags 241

F

Fast Switch 201
 files, INTOUCH.ini 18
 ForceLogCurrentValue parameter 241
 ForceLogging parameter 241
 functions
 AddPermission() function 142, 157
 AnnotateLayout() function 234, 236
 AttemptInvisibleLogon() function 166
 ChangePassword() 158
 EnableDisableKeys() function 135, 137
 GetAccountStatus() function 171
 GetNodeName() function 59
 HTSetPenName() function 44
 HTUpdateToCurrentTime() function 40
 InvisibleVerifyCredentials() function 170
 IOReinitialize() function 59
 IOSetAccessName() function 36
 IsAssignedRole() function 172
 Logoff() function 166
 LogonCurrentUser() function 165
 PostLogonDialog() function 162, 164
 QueryGroupMembership() function 173
 ReloadWindowViewer() function 51
 RestartWindowViewer() function 51
 SwitchDisplayLanguage() function 184
 TseGetClientId() function 59, 62
 TseGetClientNodeName() function 62
 TseQueryRunningOnClient() function 63
 TseQueryRunningOnConsole() function 63

WWMultiMonitorNode() function 232

G

Galaxy
 defining ArchestrA security 157
 restrictions for InTouchView
 applications 35, 218
 GetAccountStatus() function 171, 177
 GetNodeName() function 59
 GroupVar keyword 108

H

health check
 memory 208
 historical logging
 restrictions for InTouchView
 applications 218
 historical logs
 setting logging frequency 241
 history provider, dynamically configuring 44
 HistoryTrend keyword 109
 HTSetPenName() function 44
 HTUpdateToCurrentTime() function 40

I

inactivity time-out 132–134
 IndirectAnalog keyword 110
 IndirectDisc keyword 110
 IndirectMsg keyword 111
 INTOUCH.ini file
 copying to an application folder 18
 creating 18
 custom parameters 240–245
 InTouchView
 converting from an InTouch application 222
 converting to an InTouch application 221
 creating an application 219–220
 description 199
 WindowMaker restrictions 219
 InvisibleVerifyCredentials() function 170, 177
 IOAccess keyword 82, 82–84
 IODisc keyword 95
 IOInt keyword 99–101
 IOMsg keyword 107–108
 IOReal keyword 104–106
 IOReinitialize() function 59
 IOSetAccessName() function 36
 IsAssignedRole() function 172, 177

K

keywords

- GroupVar keyword 108
- HistoryTrend keyword 109
- IndirectAnalog keyword 110
- IndirectDisc keyword 110
- IndirectMsg keyword 111
- IODisc keyword 95
- IOInt keyword 99–101
- IOMsg keyword 107–108
- IOReal keyword 104–106
- MemoryDisc keyword 95
- MemoryInt keyword 96–99
- MemoryMsg keyword 107
- MemoryReal keyword 102–104
- TagID keyword 109
- using default values 112

L

language switching

- adding run-time switching 182–184
- changing font settings 181–182
- configuring 180–181
- exporting alarm comments for translation 192–193
- exporting text for offline translation 186–187
- exporting text to an existing dictionary file 187–188
- importing translated alarm comments 196
- importing translated dictionary file 190–191
- translating an exported dictionary file 188–190

logging off 163

logging on 162

Logoff() function 166, 177

LogonCurrentUser() function 165, 177

LoopTimeout parameter 242

M

memory

- health check 208
- WindowViewer 206

MemoryDisc keyword 95

MemoryInt keyword 96–99

MemoryMsg keyword 107

MemoryReal keyword 102–104

Mode keyword 79–82

multi-monitor system

- choosing a video card 227
- configuring screen resolution conversion 231
- configuring with parameters 230
- deploying an application 231–232
- description 223
- determining screen resolution 227
- determining the number of monitors to display the application 228
- determining the placement of application windows 229
- multiple video card configuration 225–226
- planning 227
- single video card configuration 224–225
- verifying support during run time 232

MultiScreen parameter 230

MultiScreenHeight parameter 230

MultiScreenWidth parameter 230

N

NAD, See Network Application Development

Net Start view command 68

Net Stop view command 68

Network Application Development

- changes to an application during an update 52–53
- configuring automatic updates 48
- configuring UNC paths for files 39
- description 34
- distributing an application 41
- I/O data access 35
- I/O data sources 35
- local addresses to I/O data sources 36
- manually updating applications 49
- planning considerations 35
- using global I/O addresses 35–36
- using local I/O addresses 36–37
- using the distributed history system 41

network architectures

- server-based 33
- single computer 32
- supported types 31

NoKeyboardResize parameter 244

Notify Clients command 34, 51

O

objects
 blink speed 202
 objects, blink speed 202
 oldrightmousebehavior parameter 242
 opening/closing windows 216

P

parameters
 16PenTrendDrawMode 244
 CommentRetentive 243
 ForceLogCurrentValue 241
 ForceLogging 241
 LoopTimeout 242
 MultiScreen 230
 MultiScreenHeight 230
 MultiScreenWidth 230
 NoKeyboardResize 244
 oldrightmousebehavior 242
 PrintScreenWait 243
 PrintWindow 243
 Resizable InputLink 245
 ScaleForResolution 231, 243
 PostLogonDialog() function 162, 164, 177
 PrintScreenWait parameter 243
 PrintWindow parameter 243
 processors 212

Q

QueryGroupMembership() function 173, 177

R

RDP, See Remote Desktop Protocol
 ReloadWindowViewer() function 51
 Remote Desktop Protocol 57, 144
 remote history provider
 creating list 42–43
 description 42
 RESET command 112
 Resizable InputLink parameter 245
 RestartWindowViewer() function 51
 run time
 customizing 199
 Fast Switch 201
 run time, customizing for InTouchView 199

S

ScaleForResolution parameter 231, 243
 scripts
 adding user permissions 157
 change user password 158
 determine user group membership 172
 log on a user to InTouch automatically 166
 managing user accounts 157–161
 retrieving information about the current
 logged on user 174
 setting loop timeout limit 242
 show the InTouch Logon dialog box 164
 stopping in run time 139, 140
 verifying user credentials 170
 security
 adding user permissions with a script 157
 authentication 141
 authorization 141
 change user password with a script 158
 changing a password at run time 155
 creating a custom logon window 164
 determine user group membership with a
 script 172
 disable animation link 170
 hiding menu items at run time 138–140
 inactivity time-out feature 132
 InTouch-based authentication 141
 locking system keys 135–137
 log on a user to InTouch automatically with
 a script 166
 logging a user on or off with ArchestrA-
 based security 163
 logging user on or off with InTouch-based
 security 162
 overview 131
 restricting access to InTouch
 functionality 169
 retrieving information about the current
 logged on user with a script 174
 setting InTouch security authentication and
 authorization 154–155
 setting up ArchestrA-based 157
 setting up operating system-based
 authorization 156–157
 show InTouch Logon dialog box with a
 script 164
 using ArchestrA-based authentication 143
 using operating system-based
 authentication 142

- verifying user credentials with a script 170
- server-based architecture 33
- services
 - configuring a user account 68–69
 - configuring WindowViewer to start as a service 67
 - description 65
 - InTouch component registry keys 71
 - manually starting 67
 - starting WindowViewer 66
 - stopping using a command 68
 - stopping using the Control Panel 68
 - troubleshooting 69
 - troubleshooting user account problems 70
 - viewing error messages 70
- SuiteLink, description 37
- SwitchDisplayLanguage() function 184, 185
- system key locking 135–137
- system tags
 - \$AccessLevel system tag 159, 164
 - \$ApplicationChanged system tag 49
 - \$ApplicationVersion system tag 50
 - \$ChangePassword system tag 155, 160
 - \$ConfigureUsers system tag 154, 161, 164
 - \$InactivityTimeout system tag 132, 134
 - \$InactivityWarning system tag 132, 134
 - \$Language system tag 184, 185
 - \$LogicRunning system tag 139
 - \$Operator system tag 59, 164, 175
 - \$OperatorDomainEntered system tag 169
 - \$OperatorEntered system tag 164, 167
 - \$OperatorName system tag 174
 - \$PasswordEntered system tag 164, 168
 - \$VerifiedUserName system tag 175

T

- Tablet PC
 - changing screen orientation 237
 - cut, copy, and paste annotations 235
 - deleting annotations 235
 - description 233
 - making window annotations 234–235
 - printing a window 236
 - selecting annotations 235
 - sending an annotated window as an e-mail attachment 236
- TagID keyword 109
- Tagname Dictionary

- creating DBLoad input file 77
- exporting contents with DBDump 74
- formatting an input file 76
- importing tags with DBLoad 114
- tags
 - \$InactivityTimeout system tag 132
 - DBLoad keywords 87
 - exporting contents of Tagname Dictionary 74
 - importing into Tagname Dictionary with DBLoad 114
 - keyword attributes 88–94
 - logging remote referenced 241
 - SuperTag instances 113
- technical support, contacting 12
- Terminal Services
 - defining Read Only applications in remote sessions 205
 - deploying InTouch applications with 58
 - description 57, 57
 - limitations 61
 - planning considerations 58
 - running scripts 60
 - securing applications 59
 - starting an I/O Server 60
 - working with Distributed Alarm System 58–59
- Terminal Services Client command 57
- tick interval 202
- transferring to WindowMaker from WindowViewer 218
- TseGetClientId() function 59, 62
- TseGetClientNodeName() function 62
- TseQueryRunningOnClient() function 63
- TseQueryRunningOnConsole() function 63

V

- Value Time Quality (VTQ) 37

W

- WindowMaker
 - converting windows to ArchestraA symbols 120–124
 - disabling transfer from WindowViewer 139
 - editing lock 52
 - opening an application 20
 - restrictions when developing InTouchView applications 219
 - setting mouse behavior 242

- windows
 - converting placeholder tags 117–118
 - converting to ArchestrA symbols 120–124
 - exporting 118–119
 - importing 115–117
- WindowViewer
 - closing all open windows on transfer to WindowMaker 201
 - closing on transfer to WindowMaker 201
 - configuring to start as a Windows service 67
 - copying an application 52
 - customizing 199
 - disabling transfer to WindowMaker 139
 - hiding menu items at run time 138–140
 - Logic menu 139
 - memory properties 206
 - setting 206
 - opening an application 20
 - resizing the numeric keypad 244
 - running as a service 66
 - setting drawing mode of 16-Pen Trend 244
 - setting option to dynamically change screen resolution 54
 - setting print waiting period 243
 - setting script looping timeout interval 242
 - starting as a service 66
- Wonderware Historian
 - configuring as a history provider 43
- WWMultiMonitorNode() function 232

