Technical Reference
# Contents

## INTRODUCTION

- Conventions used in this documentation
- About this software
- Internal structure and data flow
- Executing and switching modules
- Executing and switching the Background Task

## INSTALLATION

- Comparison of InduSoft Web Studio software components
- Install the full InduSoft Web Studio software
- Install EmbeddedView or CEView on a target device

## LICENSING

- License Settings
- Product Versions
- Execution modes
- About hardkey licenses
  - Install a new hardkey license
  - Upgrade an existing hardkey license for InduSoft Web Studio
  - Upgrade an existing hardkey license for EmbeddedView or CEView
- About softkey licenses
  - Install or upgrade a softkey license for InduSoft Web Studio
  - Install or upgrade a softkey license for EmbeddedView or CEView (Local)
  - Install or upgrade a softkey license for EmbeddedView or CEView (Remote)
- About License Server
  - Configure CodeMeter Runtime to serve licenses
  - Configure Protection Manager to get license

## THE DEVELOPMENT ENVIRONMENT

- Application button
- Recent Projects
- New
- Open Project
- Open
- Save
- Save As
- Save All
- Save All as HTML
- Save as HTML
- Save Screen Group as HTML
- Print
- Print Preview
- Print Setup
- Close
- Close All
- Exit
CREATING A NEW PROJECT .............................................................................. 89

Creating a new project .................................................................................. 90
  About target platforms and product types .................................................. 91
  Changing the product type of an existing project ...................................... 93
Configuring additional project settings ......................................................... 94
  Information tab .......................................................................................... 94
  Options tab ................................................................................................. 95
  Viewer tab ................................................................................................ 103
  Communication tab .................................................................................... 106
  Preferences tab .......................................................................................... 108
Configuring your project's default email settings ..................................... 110
Configuring your project's default FTP settings ........................................ 111
Execution Tasks dialog ............................................................................... 112
Run a project as a Windows service .............................................................. 114

TAGS AND THE PROJECT DATABASE ................................................................. 119

About Tags and the Project Database ............................................................ 120
  Project Tags Folder .................................................................................. 121
  About classes .......................................................................................... 122
  Shared Database Folder .......................................................................... 125
  System Tags Folder .................................................................................. 125
Designing a Tag ............................................................................................. 126
  Understanding the Tag Name Syntax ....................................................... 126
  Choosing the Tag Type .......................................................................... 126
  Choosing the Tag Data Type ................................................................... 128
  Choosing the Tag Scope ......................................................................... 128
Creating Database Tags ................................................................................ 130
  Adding Tags to the Datasheet ................................................................. 130
  Creating Tags "On-the-Fly" ..................................................................... 131
  Editing Tags ............................................................................................. 131
About classes ................................................................................................. 133
Contents

About tag properties........................................................................................................................................................................ 135
Tag Properties: Parameters............................................................................................................................................................... 136
Tag Properties: Alarms....................................................................................................................................................................... 137
Tag Properties: History.................................................................................................................................................................... 139
List of tag properties....................................................................................................................................................................... 140
Change how out-of-range tag values are handled......................................................................................................................... 144
Using Tags in Your Project.............................................................................................................................................................. 146
Deleting a tag from the project database........................................................................................................................................ 147
Sort or filter the rows in a worksheet............................................................................................................................................. 148
Using the Tags Toolbar........................................................................................................................................................................... 150
Global Replace Tool............................................................................................................................................................................. 150
Replacing project tags in a document or screen object................................................................................................................... 150
Removing unused tags from the project database......................................................................................................................... 151
Reset Tags Database......................................................................................................................................................................... 152
Tagname Text Box.............................................................................................................................................................................. 153
Object Finder Tool.............................................................................................................................................................................. 153
Cross Reference Tool....................................................................................................................................................................... 154
Tag Properties Tool.......................................................................................................................................................................... 154
Import Wizard..................................................................................................................................................................................... 155
Importing from Other Project Databases...................................................................................................................................... 158
Importing from OPC Server Databases.......................................................................................................................................... 160
Importing from CSV Databases...................................................................................................................................................... 161
Importing from ODBC Databases................................................................................................................................................... 162
Importing from RSLogix 5000 CSV Databases............................................................................................................................... 162
Importing from PanelBuilder32 Databases..................................................................................................................................... 163
Importing PanelMate programs.......................................................................................................................................................... 164
Importing from TwinCAT PLC Databases....................................................................................................................................... 165
Importing from OMRON CX Programmer Databases.................................................................................................................... 166
Import a Studio XML Screen............................................................................................................................................................ 167
About tag integration.......................................................................................................................................................................... 170
Edit the tags database during run time........................................................................................................................................... 171

SCREENS AND GRAPHICS ........................................................................................................................................................................ 173

Graphics tab....................................................................................................................................................................................... 174
Screen folder....................................................................................................................................................................................... 175
Screen Group Folder......................................................................................................................................................................... 178
Layout Tool...................................................................................................................................................................................... 179
Using Screen Objects and Animations........................................................................................................................................... 180
Editing............................................................................................................................................................................................... 180
Shapes......................................................................................................................................................................................................... 182
Active Objects.................................................................................................................................................................................. 191
Libraries......................................................................................................................................................................................................... 214
Applying animations to screen objects........................................................................................................................................ 234
Format tab......................................................................................................................................................................................................... 246
Change the properties of multiple screen objects......................................................................................................................... 246
Set the tab order of screen objects.................................................................................................................................................... 248
Bring to front / Send to back.............................................................................................................................................................. 249
Group and ungroup screen objects................................................................................................................................................... 250
Align, Center and Distribute Tools.................................................................................................................................................. 251
Rotate Tool........................................................................................................................................................................................... 254
Resize Tools......................................................................................................................................................................................... 255
Fill Color Tool...................................................................................................................................................................................... 255
Line Color Tool................................................................................................................................................................................... 255
Fonts Tool........................................................................................................................................................................................ 256
About using Multi-Touch in your project................................................................. 257
  About the Multi-Touch settings for project screens.............................................. 258
  About the different types of multi-touch gestures.............................................. 263
  About Touch Events............................................................................................. 269
Import a Studio XML Screen................................................................................. 277

ALARMs, EVENTs, AND TRENDs............................................................................ 279
  Alarm worksheet.................................................................................................. 280
    Alarm Worksheet Header................................................................................... 281
    Alarm Worksheet Body..................................................................................... 284
    Saving your alarm history / event log to an external database....................... 288
    Format of the alarm history............................................................................ 289
  About events and event logging......................................................................... 292
    Enabling the event logger................................................................................ 292
    Saving your alarm history / event log to an external database....................... 293
    Format of the event log................................................................................... 294
  Alarm/Event Control object................................................................................ 297
  Trend worksheet.................................................................................................. 305
    Sort or filter the rows in a worksheet.............................................................. 307
    Connect to a Wonderware Historian database................................................. 310
    Converting Trend History Files from Binary to Text....................................... 311
    Converting Trend History Files from Text to Binary....................................... 312
    Creating Batch History................................................................................... 312
  Trend Control object........................................................................................... 316
    About the trend control runtime interface...................................................... 316
    Object Properties: Trend Control dialog......................................................... 318
    Using the Data Source Text File...................................................................... 336
    Using the Data Source Database..................................................................... 339
    Grid object........................................................................................................ 344
    Data dialog....................................................................................................... 345
    Columns dialog............................................................................................... 347
    Advanced dialog.............................................................................................. 350

BACKGROUND TASKS............................................................................................ 353
  Alarm worksheet.................................................................................................. 354
  Trend worksheet.................................................................................................. 356
  Recipe worksheet................................................................................................. 359
  Report worksheet................................................................................................. 361
  ODBC worksheet............................................................................................... 363
  Math worksheet................................................................................................... 366
  Script worksheet.................................................................................................. 367
    Startup Script worksheet................................................................................. 368
  Scheduler worksheet........................................................................................... 369
  Database/ERP worksheet.................................................................................... 371
  Sort or filter the rows in a worksheet................................................................. 375

COMMUNICATION WITH OTHER DEVICES............................................................. 377
ABOUT COLLABORATION AND SOURCE CONTROL

PROJECT SECURITY

PROJECT LOCALIZATION
Trigonometric functions............................................................................................................................................................. 704
ACos................................................................................................................................................................................... 704
ASin.................................................................................................................................................................................... 704
ATan.................................................................................................................................................................................... 705
Cos......................................................................................................................................................................................705
Cot...................................................................................................................................................................................... 705
Pi.........................................................................................................................................................................................706
Sin.......................................................................................................................................................................................707
Tan...................................................................................................................................................................................... 707

Screen functions........................................................................................................................................................................709
Close................................................................................................................................................................................... 709
Open................................................................................................................................................................................... 710
OpenPrevious..................................................................................................................................................................... 713
ShowInplaceInput............................................................................................................................................................... 714
ShowMessageBox.............................................................................................................................................................. 715

Security functions...................................................................................................................................................................... 717
BlockUser............................................................................................................................................................................717
CheckESign........................................................................................................................................................................ 718
CheckSecurityLevel............................................................................................................................................................ 718
CreateUser..........................................................................................................................................................................719
ExportSecuritySystem.........................................................................................................................................................721
GetSecuritySystemStatus...................................................................................................................................................722
GetUserFullName............................................................................................................................................................... 723
GetUserNames................................................................................................................................................................... 724
GetUserPwdAging.............................................................................................................................................................. 725
GetUserState...................................................................................................................................................................... 725
ImportSecuritySystem.........................................................................................................................................................726
RemoveUser.......................................................................................................................................................................727
SetPassword....................................................................................................................................................................... 728
UnblockUser....................................................................................................................................................................... 730
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Activity functions</td>
<td>732</td>
</tr>
<tr>
<td>AppActivate</td>
<td>732</td>
</tr>
<tr>
<td>AppIsRunning</td>
<td>733</td>
</tr>
<tr>
<td>AppPostMessage</td>
<td>734</td>
</tr>
<tr>
<td>AppSendKeys</td>
<td>735</td>
</tr>
<tr>
<td>CleanReadQueue</td>
<td>735</td>
</tr>
<tr>
<td>CloseSplashWindow</td>
<td>736</td>
</tr>
<tr>
<td>DisableMath</td>
<td>736</td>
</tr>
<tr>
<td>EnableMath</td>
<td>737</td>
</tr>
<tr>
<td>EndTask</td>
<td>737</td>
</tr>
<tr>
<td>ExitWindows</td>
<td>738</td>
</tr>
<tr>
<td>IsScreenOpen</td>
<td>738</td>
</tr>
<tr>
<td>IsTaskRunning</td>
<td>739</td>
</tr>
<tr>
<td>IsViewerInFocus</td>
<td>740</td>
</tr>
<tr>
<td>KeyPad</td>
<td>740</td>
</tr>
<tr>
<td>LogOff</td>
<td>741</td>
</tr>
<tr>
<td>LogOn</td>
<td>742</td>
</tr>
<tr>
<td>Math</td>
<td>743</td>
</tr>
<tr>
<td>PostKey</td>
<td>743</td>
</tr>
<tr>
<td>Recipe</td>
<td>744</td>
</tr>
<tr>
<td>Report</td>
<td>745</td>
</tr>
<tr>
<td>RunGlobalProcedureAsync</td>
<td>746</td>
</tr>
<tr>
<td>RunGlobalProcedureAsyncGetStatus</td>
<td>747</td>
</tr>
<tr>
<td>RunGlobalProcedureOnFalse</td>
<td>749</td>
</tr>
<tr>
<td>RunGlobalProcedureOnServer</td>
<td>750</td>
</tr>
<tr>
<td>RunGlobalProcedureOnTrigger</td>
<td>751</td>
</tr>
<tr>
<td>RunGlobalProcedureOnTrue</td>
<td>753</td>
</tr>
<tr>
<td>RunVBScript</td>
<td>754</td>
</tr>
<tr>
<td>SecureViewerReload</td>
<td>755</td>
</tr>
<tr>
<td>SendKeyObject</td>
<td>755</td>
</tr>
<tr>
<td>SetAppPath</td>
<td>757</td>
</tr>
<tr>
<td>SetViewerInFocus</td>
<td>758</td>
</tr>
<tr>
<td>SetViewerPos</td>
<td>758</td>
</tr>
<tr>
<td>ShutDown</td>
<td>759</td>
</tr>
<tr>
<td>StartTask</td>
<td>760</td>
</tr>
<tr>
<td>ViewerPostMessage</td>
<td>760</td>
</tr>
<tr>
<td>WinExec</td>
<td>761</td>
</tr>
<tr>
<td>WinExecIsRunning</td>
<td>763</td>
</tr>
</tbody>
</table>
Contents

File functions..................................................................................................................................................................765
  DeleteOlderFiles..................................................................................................................................................765
  DirCreate............................................................................................................................................................765
  DirDelete............................................................................................................................................................766
  DirLength...........................................................................................................................................................767
  DirRename...........................................................................................................................................................767
  FileCopy..............................................................................................................................................................768
  FileDelete...........................................................................................................................................................769
  FilenLength..........................................................................................................................................................769
  FileReadFields..................................................................................................................................................770
  FileReadMessage.............................................................................................................................................771
  FileRename........................................................................................................................................................772
  FileWrite............................................................................................................................................................772
  FileWriteFields................................................................................................................................................774
  FileWriteMessage............................................................................................................................................775
  FindFile..............................................................................................................................................................776
  FindPath..............................................................................................................................................................778
  GetFileAttributes.............................................................................................................................................778
  GetFileTime.........................................................................................................................................................779
  GetHstInfo..........................................................................................................................................................779
  GetLine...............................................................................................................................................................780
  Hst2Txt.................................................................................................................................................................782
  HST2TXTIsRunning...........................................................................................................................................784
  ImportXML..........................................................................................................................................................784
  LookupContains...............................................................................................................................................786
  LookupGet..........................................................................................................................................................787
  LookupLoad.........................................................................................................................................................787
  PDFCreate...........................................................................................................................................................788
  Print.....................................................................................................................................................................789
  RDFilen.................................................................................................................................................................789
  WebGetFile........................................................................................................................................................790

Graphic functions....................................................................................................................................................792
  AutoFormat.........................................................................................................................................................792
  ExecuteCustomCall.........................................................................................................................................792
  GetScrInfo........................................................................................................................................................793
  PrintSetup........................................................................................................................................................795
  PrintWindow.......................................................................................................................................................795
  ResetDecimalPointsTable...............................................................................................................................797
  RGBColor.........................................................................................................................................................797
  RGBComponent.................................................................................................................................................797
  SaveScreenShot...............................................................................................................................................798
  SetDecimalPoints............................................................................................................................................799
  SetDisplayUnit................................................................................................................................................800
  SetTagDisplayUnit.......................................................................................................................................801

Translation functions...........................................................................................................................................803
  Ext.....................................................................................................................................................................803
  SetLanguage......................................................................................................................................................803
  SetTranslationFile..........................................................................................................................................804
  TranslationLoad...............................................................................................................................................805
  TranslationLookupClose.................................................................................................................................806
  TranslationLookupGet......................................................................................................................................806
  TranslationLookupLoad.....................................................................................................................................807

Multimedia functions..........................................................................................................................................809
  Play.....................................................................................................................................................................809
System Info functions

System Info functions

DBVersion................................................................................................................................................................810
GetAppHorizontalResolution...........................................................................................................................................810
GetAppPath....................................................................................................................................................................810
GetAppVerticalResolution............................................................................................................................................. 811
GetComputerIP...............................................................................................................................................................811
GetComputerName.......................................................................................................................................................812
GetCursorX....................................................................................................................................................................812
GetCursorY....................................................................................................................................................................813
GetDisplayHorizontalResolution........................................................................................................................................813
GetDisplayVerticalResolution........................................................................................................................................813
GetFreeMemoryCE........................................................................................................................................................814
GetHardKeyModel........................................................................................................................................................814
GetHardKeySN..............................................................................................................................................................815
GetIPAll........................................................................................................................................................................815
GetLicenseGracePeriod................................................................................................................................................816
GetMemoryCE...............................................................................................................................................................817
GetNetMACID...............................................................................................................................................................817
GetOS............................................................................................................................................................................ 818
GetPerformanceMetric................................................................................................................................................819
GetPrivateProfileString...............................................................................................................................................822
GetProductPath............................................................................................................................................................823
GetRegValue...............................................................................................................................................................823
GetRegValueType........................................................................................................................................................824
GetServerHostName...................................................................................................................................................825
GetTickCount...............................................................................................................................................................825
InfoAppAirDir..............................................................................................................................................................825
InfoAppHstDir..............................................................................................................................................................826
InfoDiskFree...............................................................................................................................................................826
InfoResources.............................................................................................................................................................827
IsActiveXReg.............................................................................................................................................................827
IsAppChangedOnServer................................................................................................................................................828
NoInputTime...............................................................................................................................................................829
ProductVersion...........................................................................................................................................................830
RegSaveCE.................................................................................................................................................................830
ReloadAppFromServer................................................................................................................................................831
SaveAlarmFile.............................................................................................................................................................831
SetAppAlarmPath.........................................................................................................................................................832
SetAppHstPath...........................................................................................................................................................832
SetDateFormat............................................................................................................................................................833
SetKeyboardLanguage................................................................................................................................................833
SetRegValue...............................................................................................................................................................834
SetWebConfig............................................................................................................................................................835
SNMPGet....................................................................................................................................................................837
SNMPSet....................................................................................................................................................................838
WritePrivateProfileString.............................................................................................................................................839
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBC functions</td>
<td>872</td>
</tr>
<tr>
<td>ODBCBeginTrans</td>
<td>872</td>
</tr>
<tr>
<td>ODBCBindCol</td>
<td>872</td>
</tr>
<tr>
<td>ODBCCanAppend</td>
<td>873</td>
</tr>
<tr>
<td>ODBCCanTransact</td>
<td>874</td>
</tr>
<tr>
<td>ODBCCanUpdate</td>
<td>874</td>
</tr>
<tr>
<td>ODBCClose</td>
<td>875</td>
</tr>
<tr>
<td>ODBCCommitTrans</td>
<td>875</td>
</tr>
<tr>
<td>ODBCDelete</td>
<td>876</td>
</tr>
<tr>
<td>ODBCExecuteSQL</td>
<td>876</td>
</tr>
<tr>
<td>ODBCInsert</td>
<td>877</td>
</tr>
<tr>
<td>ODBCisBOF</td>
<td>877</td>
</tr>
<tr>
<td>ODBCisDeleted</td>
<td>878</td>
</tr>
<tr>
<td>ODBCisEOF</td>
<td>878</td>
</tr>
<tr>
<td>ODBCisFieldNULL</td>
<td>879</td>
</tr>
<tr>
<td>ODBCisFieldNullable</td>
<td>879</td>
</tr>
<tr>
<td>ODBCMove</td>
<td>880</td>
</tr>
<tr>
<td>ODBCMoveFirst</td>
<td>881</td>
</tr>
<tr>
<td>ODBCMoveLast</td>
<td>881</td>
</tr>
<tr>
<td>ODBCMoveNext</td>
<td>882</td>
</tr>
<tr>
<td>ODBCMovePrev</td>
<td>882</td>
</tr>
<tr>
<td>ODBCopen</td>
<td>883</td>
</tr>
<tr>
<td>ODBCQuery</td>
<td>883</td>
</tr>
<tr>
<td>ODBCRollBack</td>
<td>884</td>
</tr>
<tr>
<td>ODBCSetFieldNULL</td>
<td>885</td>
</tr>
<tr>
<td>ODBCSetFilter</td>
<td>885</td>
</tr>
<tr>
<td>ODBCSetSort</td>
<td>886</td>
</tr>
<tr>
<td>ODBCUnbindCol</td>
<td>886</td>
</tr>
<tr>
<td>ODBCUpdate</td>
<td>887</td>
</tr>
<tr>
<td>Email functions</td>
<td>888</td>
</tr>
<tr>
<td>CnfEmail</td>
<td>888</td>
</tr>
<tr>
<td>GetStatusSendEmailExt</td>
<td>890</td>
</tr>
<tr>
<td>SendEmail</td>
<td>890</td>
</tr>
<tr>
<td>SendEmailExt</td>
<td>891</td>
</tr>
<tr>
<td>Dial-up functions</td>
<td>894</td>
</tr>
<tr>
<td>DialError</td>
<td>894</td>
</tr>
<tr>
<td>DialGetClientIP</td>
<td>899</td>
</tr>
<tr>
<td>DialGetServerIP</td>
<td>900</td>
</tr>
<tr>
<td>DialStatus</td>
<td>901</td>
</tr>
<tr>
<td>DialUp</td>
<td>903</td>
</tr>
<tr>
<td>DialUpToCE</td>
<td>904</td>
</tr>
<tr>
<td>FindAllDevices</td>
<td>906</td>
</tr>
<tr>
<td>FindModem</td>
<td>906</td>
</tr>
<tr>
<td>HangUp</td>
<td>907</td>
</tr>
<tr>
<td>PhoneDialUp</td>
<td>908</td>
</tr>
<tr>
<td>PhoneDisableListen</td>
<td>908</td>
</tr>
<tr>
<td>PhoneEnableListen</td>
<td>909</td>
</tr>
<tr>
<td>PhoneHangUp</td>
<td>910</td>
</tr>
<tr>
<td>PhoneStatus</td>
<td>910</td>
</tr>
<tr>
<td>ActiveX and .NET Control functions</td>
<td>912</td>
</tr>
<tr>
<td>XGet</td>
<td>912</td>
</tr>
<tr>
<td>XRun</td>
<td>912</td>
</tr>
<tr>
<td>XSet</td>
<td>913</td>
</tr>
<tr>
<td>Event Logger functions</td>
<td>915</td>
</tr>
<tr>
<td>SendEvent</td>
<td>915</td>
</tr>
<tr>
<td>FTP functions</td>
<td>917</td>
</tr>
<tr>
<td>CnfFTP</td>
<td>917</td>
</tr>
<tr>
<td>FTPGet</td>
<td>918</td>
</tr>
<tr>
<td>FTPPut</td>
<td>919</td>
</tr>
<tr>
<td>FTPStatus</td>
<td>920</td>
</tr>
</tbody>
</table>
OVERVIEW OF VBSCRIPT

VBScript Interfaces in the Software

Global Procedures
Graphic Module
Background Task

Language Reference

Operators
Constants
Objects and Collections
Properties
Statements
Methods
Functions
Keywords
Errors

Tips & Tricks

VBScript Editor IntelliSense
VBScript Compared to VBA
Screen Events
MsgBox and InputBox Functions
VBScript Procedures
Creating Constants
Declaring Variables
Scope and Lifetime of Variables
Boolean tags and Boolean variables
Writing Real Values to Integer Tags
Precedence of VBScript Operators
Logical Operator NOT
Using Conditional Statements
Looping Through Code
Support for ActiveX Controls
Support for VBScript on Windows Embedded
Introduction

This User Guide and Technical Reference was designed to help you get the best results from your InduSoft Web Studio™ software. This document provides technical information and step-by-step instructions for all the tasks you need to create Web-enabled HMI/SCADA programs.

Who should read this

This User Guide and Technical Reference is a comprehensive document designed to provide useful information for both novice and advanced users of IWS.

- **New Users**: This publication uses a step-by-step, hands-on approach to the project development process. Be sure to read the introductory chapters describing the product’s features and development environment.

- **Experienced Users**: This publication offers advanced instructions, tips, and troubleshooting information to help you get the most out of your projects.

**Note**: We assume you are familiar with working in a Windows environment, and we do not attempt to explain Windows navigation, file management, and so forth. If you are unfamiliar with any of these procedures, we recommend using the Windows Help feature (Start > Help) or consulting your Microsoft Windows documentation.

Contents

The information in this document is organized into the following sections:

**Introduction**

This section, which provides an overview of the features and architecture of InduSoft Web Studio.

**Installation**

Step-by-step instructions for installing and uninstalling the full InduSoft Web Studio software for Windows and the EmbeddedView or CEView runtime for Windows Embedded.

**Licensing**

Describes the licensing scheme for InduSoft Web Studio, as well as how to install and upgrade licenses on different platforms.

**The Development Environment**

A tour of the IWS development environment. Also, some basic skills and techniques you should understand before you create a new project.

**Creating a New Project**

Provides step-by-step instructions for creating and configuring a new project.

**Tags and the Project Database**

Explains basic concepts about the product database, tag types (arrays, classes, and pointers), tag values and parameters. Following the concepts discussion, this chapter provides instructions for creating and editing tags for your projects.

**Screens and Graphics**

Explains how to use the different IWS development tools to create your project screens and graphics.

**Alarms, Events, and Trends**

Explains how to create and configure task worksheets and screen objects to save and display historical data.

**Background Tasks**

Explains how to create and configure task worksheets for the other major background tasks.

**Communication with Other Devices**

Describes how to configure your project to read from and write to a device’s registers. The information includes instructions for configuring drivers and OPC, TCP/IP, and DDE communication.
Collaboration and Source Control
Explains how to collaborate with others on project development using Team Foundation Server.

Project Security
Explains how to set-up and manage a security system for your projects.

Project Localization
Explains how to use the Translation Tool to translate the text in your projects from one language to another.

Testing and Debugging
Discusses how to test and debug projects using tools such as the Database Spy and Output windows. This chapter includes a list of possible error messages and methods for correcting those errors.

Remote Management
Explains how to download, monitor, and debug projects from a remote runtime workstation.

Thin Clients and Mobile Access
Explains how configure and run your projects on the Web.

Database Interface
Explains how to connect IWS to compatible databases.

Troubleshooting
Provides instructions for verifying projects, describes some common development errors, and explains what to do if you need to contact a support representative.

Scripting Languages
Describes IWS’s built-in scripting language, as well as the support for VBScript in IWS.

Related documentation
You may want to review the following manuals in addition to this Technical Reference:

- **Quickstart Guide**: Provides basic information about InduSoft Web Studio, including a systematic tutorial that allows you to develop a single project and become familiar with the product in a short time.

- **Tutorial Manual**: Describes how to build a project, step-by-step, with the main product features. You can use this document as a self-training manual.

- **Drivers User Guides**: Explain how to configure individual direct communication drivers, according to their unique protocol characteristics. One customized user guide is included with each driver.

**Note:** All manuals are located in the *Documentation* folder on the IWS installation CD. IWS installs the Drivers User Guides in the `\Drv` folder in the program directory. You also can access technical information from the *Help* menu.
Conventions used in this documentation

This documentation uses standardized formatting and terminology to make it easier for all users to understand.

Text conventions
This documentation uses special text formatting to help you quickly identify certain items:

- Titles, labels, new terms, and messages are indicated using italic text (for example, Object Properties).
- File names, screen text, and text you must enter are indicated using monospace text (for example, D:\Setup.exe).
- Buttons, menu options, and keyboard keys are indicated using a bold typeface (for example, File menu).

In addition, this documentation segregates some text into Tip, Note, and Caution boxes:

- Tips provide useful information to save development time or to improve the project performance.
- Notes provide extra information that may make it easier to understand the nearby text, usually the text just before the note.
- Cautions provide information necessary to prevent errors that can cause problems when running the project, and may result in damage.

Mouse and selection conventions
Because most PCs used for project development run a version of Microsoft Windows with a mouse, this documentation assumes you are using a mouse. Generally, a PC mouse is configured for right-handed use, so that the left mouse button is the primary button and the right mouse button is the secondary button.

This documentation uses the following mouse and selection conventions:

- Click and Select both mean to click once on an item with the left mouse button. In general, you click buttons and you select from menus and lists.
- Double-click means to quickly click twice on an item with the left mouse button.
- Right-click means to click once on an item with the right mouse button.
- Select also means you should use your pointing device to highlight or specify an item on the computer screen. Selecting an item with a touchscreen is usually the same as selecting with a mouse, except that you use your finger to touch (select) a screen object or section. To select items with your keyboard, you typically use the Tab key to move around options, the Enter key to open menus, and the Alt key with a letter key to select an object that has an underlined letter.
- Drag means to press down the appropriate mouse button and move the mouse before releasing the button. Usually an outline of the item will move with the mouse cursor.

Windows conventions
This documentation uses the following Windows conventions:

- Dialogs are windows that allow you to configure settings and enter information.
- Text boxes are areas in dialogs where you can type text.
- Radio buttons are white circles in which a black dot appears or disappears when you click on the button. Typically, the dot indicates the option is selected or enabled. No dot indicates the option is cleared or disabled.
- Check boxes are white squares in which a check (√) appears or disappears when you click on it with the cursor. Typically, a check √ indicates the option is selected or enabled. No check ☐ indicates the option is cleared or disabled.
- Buttons are icons in boxes appear “pressed” when you click on them.
- Lists are panes (white boxes) in windows or dialogs containing two or more selectable options.
- Combo boxes have arrows that, when clicked, show part or all of an otherwise concealed list.
- **Dockable windows** are windows that you can drag to an edge of the interface and merge with that edge.
About this software

InduSoft Web Studio is powerful software for developing HMI, SCADA, and OEE/Dashboard projects that can be deployed anywhere.

Each IWS project consists of:

- A project tags database to manage all run-time data, including both internal variables and I/O data;
- Configurable drivers to communicate in real-time with programmable logic controllers (PLCs), remote I/O devices, and other data-acquisition equipment;
- Animated HMI screens and OEE dashboards; and
- Optional modules such as alarms, events, trends, recipes, reports, scriptable logic, schedulers, a project security system, and a complete database interface.

After you develop your project, you can either run it locally on your development workstation or download it to a remote computer and run it there. The project runtime server processes I/O data from connected devices according to your project parameters and then reacts to, displays, and/or saves the data.

Product features

ActiveX and .NET

Use third-party controls to enhance your project. IWS is a container for ActiveX and .NET controls. Add functionality such as browsers, media players, charting, and other tools that support the ActiveX and .NET interface standards.

Alarms

In addition to all the alarm functions you’d expect, IWS v7.1+SP3 also sends alarms using multi-media formats like PDF. Use remote notification to have alarms sent right to your inbox, a printer, or a smartphone! Alarms are real-time and historical, log data in binary format or to any database.

Animation

IWS gives you great command over graphics. Paste images, and even rotate them dynamically. Fill bar graphs with color, or adjust the scale of objects with easy-to-use configuration. Other animations include “command” (for touch, keyboard and mouse interaction), hyperlink, text data link, color, resize (independent height and width), position, and rotation (with custom rotation point).

Database

Connect to any SQL database (MS SQL, MySQL, Sybase, Oracle), or MS Access or Excel, and ERP/MES systems (including SAP), even from Windows Embedded. Flexible enough to have a built-in interface without the need for knowing SQL (for trends, alarms/events, grid and other objects), or use any SQL statement you need anywhere you need it.

Drivers

IWS v7.1+SP3 contains over 240 built-in drivers for most PLCs, temperature controllers, motion controllers, and bar code/2D/RFID readers. InduSoft driver toolkits allow you the flexibility to build your own drivers. Use these built in drivers without the need for OPC servers (but are an optional connection method).

Email

Send email using SMTP to desktop, email enabled phone, or any enabled device. Get real-time information on alarms, process values, and other events. InduSoft Web Studio v7.1+SP3 supports SSL encryption allowing the use of third-party providers such as Gmail.

Events

IWS v7.1+SP3 offers traceability for operator initiated actions or internal system activity. Log events such as security system changes (user logon or off), screen open/close, recipe/report
operations, custom messages and system warnings. Also any tag value changes including custom messages.

**FDA Traceability**
Take advantage of built-in functionality to create 21 CFR part 11 compliant projects with traceability and e-signatures. These features are often used for pharmaceutical and food applications, but also for any application where traceability is a must.

**FTP**
Automatically upload or download files during runtime to/from remote storage locations using FTP protocol and flexible scripting functions. Configure FTP via scripting or the included configuration interface.

**Graphics and Design Tools**
Create powerful screens to meet any application need using the improved tools in our graphic interface. Combine built-in objects to create any functionality required. Store graphics in the library for future use, or easily make project across a product line share a consistent “look and feel”.

**Historical Performance**
We have optimized the trend history module, and designed it to load millions of values from SQL Relational Databases with high performance, with built-in data decimation in the Trend Control. Easy to use tools provide quick access to Statistical Process Control (SPC) values without any need for programming.

**Intellectual Property Protection**
Screens, documents, scripts and even math worksheets can be individually password protected. This prevents unauthorized viewing or editing of your corporate custom functionality. Protect the entire project with just a few mouse clicks.

**Multi-Language**
Develop your project in one of many development languages, including English, Portuguese, German, and French.

**OPC**
Drivers for all major brands of PLCs are built in, but any OPC server may optionally be used. IWS supports OPC DA (Server/Client), OPC HDA(Server), UA (Client) and OPC .NET 3.0 (Client). InduSoft Web Studio v7.1+SP3 also supports OPC XML as an additional add-on.

**PDF Export**
Send Alarms, Reports, or any file (including .doc or .txt) to a production supervisor, quality manager, or maintenance staff using the included PDF writer.

**Recipes**
Save time and maintain consistency by automating part parameters or productions quantities with any triggering event.

**Redundancy**
For critical applications where data is vital, IWS v7.1+SP3 supports web server, database and overall system redundancy.

**Reports**
Create clear, concise reports in text format, graphical RTF, XML, PDF, HTML, and CSV or integrate with Microsoft Office. Get the data you need, in the format you need it, to make informed decisions, fast.

**Scalable**
Develop once and deploy anywhere, on any currently supported version of Microsoft Windows.
Scheduler
Schedule custom tag changes on date/time, frequency, or any trigger. Use this for simulation, to trigger reports or other functionality at a particular time of day, or even to trigger driver worksheets to read/write at a scan rate you choose.

Scripting
Two powerful scripting languages are supported. Use built-in functions or use standard VBScript to take advantage of widely available resources. Both can be used simultaneously to give you the functionality you need.

Security
IWS includes support for group and user accounts, e-signatures, and traceability, as well as support for the ADAM Server, in addition to standard LDAP Servers. Integrate your project to the Active Directory (Users and Groups).

SSL Support for Emails
Native support for Secure Socket Layer (SSL), makes it easy and secure to send emails from InduSoft Web Studio using third-party tools such as Gmail!

Standards
Take advantage of common industry standards to develop projects that are compatible with any format. TCP/IP, .Net, ActiveX, OPC (client and server), ADO/ODBC, COM/DCOM, OLE, DDE, XML, SOAP, and HTML are supported.

SNMP
Easily configure managed networked devices on IP networks (such as switches and routers) using incorporated SNMP configuration commands and an easy-to-use configuration interface.

Symbols
Included library features push buttons, pilot lights, tanks, sliders, meters, motors, pipes, valves and other common objects. Use the included symbols in your project, modify existing symbols to suit your needs, or create your own from scratch. Plus support for third-party symbol libraries and graphic tools.

Tag Database
IWS features an object oriented database with boolean, integer, real, strings, arrays, classes (structures), indirect tags and included system tags.

Thin Clients
Remotely view screens as web pages using the Internet Explorer browser or InduSoft Secure Viewer. Or use Mobile Access to access your project using mobile devices such as smartphones and tablets.

Trends
Real-time and Historical trends are supported. Log data in binary format or to any database locally and remotely. Color or fill trends with graphic elements to enhance clarity of data. Date/Time based or numeric (X/Y plot) trends give you the flexibility to display information that best suits your project.

Troubleshooting
Quickly debug and verify a project using local and remote tools for troubleshooting, including status fields, DatabaseSpy and LogWin. Capture screen open and close times, see communications in real-time, and messages related to OPC, recipes/reports, security, database errors and even custom messages. Quickly get your project finished using these powerful tools.
Internal structure and data flow

The InduSoft Web Studio project runtime runs on a Windows computer or embedded system and consists of the following modules or threads (program elements that can execute independently of other program elements):

**Background Task**
A supervisory task that executes other internal tasks (IWS worksheets). For example, the Background task executes scripts configured in the Math and Scheduler worksheets and manages parameters configured in the Alarm, Recipe, Report, and Trend worksheets.

**Database Spy**
A debugging tool that...
- Executes functions and/or expressions for testing purposes
- Reads data (such as tag values) from the tags database
- Writes data (such as tag values) to the Tags database

**DDE Client**
Manages DDE communication with a DDE Server (local or remote), according to parameters configured in the DDE Client worksheets.

**DDE Server**
Manages DDE communication with a DDE Client (local or remote).

**LogWin**
A debugging tool that traces messages generated from other modules/tasks.

**Driver Runtime**
Manages the read/write commands configured in the Driver worksheets.

**OPC Client**
Manages OPC communication with an OPC Server (local or remote), according to parameters configured in the OPC Client worksheets.

**OPC Server**
Manages OPC communication with an OPC Client (local or remote).

**ODBC Runtime**
Manages ODBC data communication with any SQL relational database, according to parameters configured in the ODBC worksheets.

**TCP/IP Client**
A "thick client" that manages TCP/IP communication with the TCP/IP Server module in another IWS project, according to parameters configured in the TCP/IP Client worksheets.

**TCP/IP Server**
Manages TCP/IP communication messages with both thick clients (i.e., the TCP/IP Client module in other IWS projects) and thin clients (i.e., the Viewer module).

**Viewer**
Executes all scripts (On Open, On While, On Close, Command, Hyperlink, and so forth) configured for project screens and updates the screen objects.

None of the preceding runtime modules exchange data directly with another module or task. Instead, runtime modules send data to and receive data from the tags database, which is the "heart" of InduSoft Web Studio.

The tags database manages the flow of data between modules. In addition, the tags database stores all tag values and the status of all properties associated with each tag (such as alarm conditioning, timestamp, quality, and so forth).
Note: Tags are variables (such as communication points in field equipment, calculation results, alarm points, and so on) that are used in screens and worksheets. For more information, see About Tags and the Project Database on page 120.

Architecture of the project runtime on Windows

Architecture of the project runtime on Windows Embedded
Each runtime module contains a virtual table of the tags that are relevant for that module at the current time. The tags database uses this table to determine which information must be updated in each module. For example, the Viewer contains a virtual table that lists all tags configured for all of the open project screens. If a tag value changes, the tags database sends a message to the Viewer, and then the Viewer updates the value in all objects where the tag is configured.

For example, if a driver reads a new value from the PLC, the driver updates the tag associated with this value in the tags database. Then, if this new information must display on the project screen, the tags database sends the new tag value to the Viewer task, and the Viewer updates the screen.

![An example of data flow](image)

Note that the driver does not send new tag values directly to the Viewer. In addition, there is no pooling between tasks — the tags database receives the updated information and immediately forwards it to all runtime tasks requiring that information.

**Note:** The Viewer module will update an object only when (at least) one of the object’s tag values change.

If you configure an object animation (e.g., Text Data Link) with a function that does not require a tag (e.g., NoInputTime), then the Viewer will not update the object because there is no tag associated with that object.

The architecture of InduSoft Web Studio significantly improves the internal data flow performance and makes it easy for you to add new internal tasks. Even though each task works independently, it can access information from any other task through the tags database.
Executing and switching modules

InduSoft Web Studio is a SCADA system composed of modules that must be executed simultaneously. Based on the multi-tasking concept, each runtime module (Viewer, Driver, and so forth) is a thread and the operating system switches from one thread to another automatically. It is a common misconception that you execute a SCADA system when you execute a PLC program. A PLC program contains a simple loop:

```
PLC Program Loop
```

However, in a SCADA system, there are several modules running simultaneously, and most of them can read and write data. Because a SCADA system modifies data (tag values) continuously during task execution, the preceding diagram is not applicable.

InduSoft Web Studio only has one run-time process: Studio Manager.exe on Windows or CEView.exe on Windows Embedded. When you run a project, this process starts the Tags database and all of the runtime modules configured for the project. You can specify which modules (such as Viewer and Driver) will start during run time.

Each process keeps a list of active threads for the operating system. Actually, each process activates and deactivates each thread during the runtime, according to the algorithm of each process. Also, when you create a thread you specify a priority value. The operating system continuously scans all currently active threads, and executes the threads according to their priority value — executing the higher-priority threads first. When threads with higher-priority values are active, the threads with lower-priority values are not executed at all. If there is more than one thread with the same priority value, and there are no other threads with higher-priority values, the operating system keeps switching between the threads with the same priority.

**Note:** All InduSoft Web Studio threads are set to priority 7, which is `THREAD_PRIORITY_NORMAL`. (Most programs contain this priority value.)

Real-time program (such as SoftPLCs and Device Drivers) threads are assigned a higher-priority value (`THREAD_PRIORITY_HIGHEST`); however, these programs must provide a mechanism to keep them inactive for some period of time or the threads with normal priority would never be executed.

InduSoft Web Studio uses the `UNICOMM.DLL` library for serial drivers. This library creates a `THREAD_PRIORITY_HIGHEST` thread that "sleeps" (remains inactive) until data arrives in the serial channel. When InduSoft Web Studio detects new data in the serial channel, the `THREAD_PRIORITY_HIGHEST` thread "wakes up" (becomes active) and transfers the data from the operating system buffer to the thread buffer, where it can be read by the Driver. This thread is the only highest-priority thread created by InduSoft Web Studio.

If you allowed threads to remain active all the time, the CPU usage would be 100% all the time, which must be avoided for performance reasons. Every program provides a mechanism to prevent threads from staying active all the time.

InduSoft Web Studio uses the following parameters to prevent threads from staying active continuously:
• **TimeSlice** (from operating system): Causes the operating system to switch automatically between active threads with the same priority value.

By default, the operating system executes each active thread for approximately 20ms and then switches to the next active thread. In other words, if there are multiple active threads with the same priority value waiting to be executed, the operating system will not execute any one active thread for more than 20ms.

• **TimeSlice** (from InduSoft Web Studio): Specifies how long each IWS thread can remain continuously active.

You use this parameter in addition to the operating system's TimeSlice parameter. You configure a TimeSlice value for each IWS thread (except the Background Task) and specify how long each thread can remain continuously active. As long as a thread is active, the operating system can switch to that thread.

• **Period** (from InduSoft Web Studio): Specifies the maximum amount of time each IWS thread (except the Background Task) can remain inactive.

---

**Note:** We strongly recommend that you do not change these default values unless it is absolutely necessary. Configuring these parameters incorrectly can cause the entire system to malfunction (for example, CPU usage will go to 100%) and/or cause some tasks to perform poorly.

If you must change the parameter defaults, note the values before making your changes so if a malfunction occurs you can return to the original settings.

To change the IWS TimeSlice and Period parameter default values:

1. From the InduSoft Web Studio installation directory (for example, C:\Program Files\installation folder\Bin), double-click BIN to open the folder.

2. Double-click the **Program Settings.INI** file to open the file in Microsoft® Notepad.

   The following is a list of all parameters contained in this .ini file and their default values (in milliseconds).

   ```ini
   [Period]
   DBSpy=1000
   UniDDEClient=200
   UniDDE=200
   Driver=20
   LogWin=100
   UniODBCRT=100
   OPCClient=20
   OPCServer=20
   TCPClient=100
   TCPServer=100
   Viewer=50

   [TimeSlice]
   UniDDEClient=100
   Driver=10
   OPCClient=10
   OPCServer=10
   TCPClient=200
   TCPServer=200
   Viewer=200
   ```

   **Note:** You may not see all of these parameters listed when you open your Program Settings.INI file. However, even if a parameter is not visible in your list, IWS still uses that parameter and its default value.

   • To change the default value of a displayed parameter: In Notepad, delete the default value and type the new value in its place.

   • To change the default value of a parameter that is not displayed in your list: In Notepad, type the parameter name exactly as shown in the following list, the equal sign, and then the new value.

3. Save the file (File > Save) and close Notepad (File > Exit).
The following figure illustrates how IWS executes a generic thread (such as the Viewer).

**Executing a Generic Thread**

Where:
- Signal 1 is the Period time period (set to 50ms for this example).
- Signal 2 shows when the thread is active for the operating system.
- Signal 3 is the TimeSlice time period (set to 30ms for this example).
- Signal 4 shows the execution of the thread itself.

In this example, IWS generates a Period message every 50ms (signal 1). When IWS generates this message, its thread becomes active and remains active until the specified TimeSlice time period (from IWS) expires. The thread then remains inactive until IWS generates the next Period message (signal 1).

While the thread is active, the operating system is responsible for executing that thread. However, just because a thread is active does not mean the operating system will execute it immediately — the operating system may be executing other threads, for example.

When the operating system executes the thread, the TimeSlice timer starts counting and the thread is executed for 20ms (TimeSlice from the operating system). After the 20ms period, the operating system automatically switches to the next active thread (such as the Driver), and so on.

In the above example, the TimeSlice time was set to 30ms, which means the operating system is not supposed to execute the thread more than once in each TimeSlice of IWS. However, if you specify higher values for the IWS TimeSlice time period, it is likely that the operating system will execute the same thread more than once in the same TimeSlice period.

In the next example, the Period and the TimeSlice values were changed as follows, but the default operating system TimeSlice period (20 ms) was not changed.

**Setting a Higher TimeSlice**

Where:
- Signal 1 is the Period time period (set to 100ms).
- Signal 2 shows when the thread is active for the operating system.
• Signal 3 is the IWS TimeSlice time period (set to 80ms).
• Signal 4 shows the execution of the thread itself.

Notice that the thread can be executed more than once in the same TimeSlice time period. When the IWS TimeSlice time period expires, the operating system interrupts the thread execution; however, even though the IWS Period and TimeSlice parameters are set to 100ms and 80ms respectively, the operating system will not execute this thread continuously for more than 20ms, because the operating system TimeSlice time period is set to 20ms.

When the operating system is not executing the Viewer thread, the CPU can execute any other thread or remain idle (if there are no other active threads to execute). Remember, the IWS Period and TimeSlice parameters were created to prevent all threads from being active at the same time to prevent 100% CPU usage.

During thread execution, the thread must handle its pending messages. For example, the Viewer module must update any related screen objects. If there are no messages pending, the thread deactivates itself and gives control back to the operating system. The operating system immediately switches to the next active thread. In other words, a thread can interrupt its own execution — even if the operating system TimeSlice time period has not yet expired (which occurs frequently in real-world applications).

**Note:** The Database Spy, DDE Server, LogWin, and ODBC Runtime modules do not have a TimeSlice parameter. Consequently, after each thread handles all of its pending messages, the threads become inactive until the next Period message for each one of the threads occurs.

The Background Task is the exception to the execution/switching process just discussed. The mechanism for executing/switching the Background Task is described in the next section.
Executing and switching the Background Task

The Background Task executes scripts from the Math and Scheduler worksheets (for example, messages from Alarm and Trend worksheets). In addition, the Background Task executes all Recipe and Report commands when the Recipe or Report functions are executed during the runtime.

Although the Alarm, Math, Scheduler, and Trend tasks are not threads, you can specify or change their Period time in the Program Settings.ini file located in the IWS program directory.

The Period default values (in milliseconds) are as follows:

```
[Period]
Math=100
Sched=50
Alarm=100
Trend=1000
```

These values mean that every 100ms, IWS generates a Period message to the Math task. Every 50ms, IWS generates a Period message to the Scheduler task, and so on.

**Caution:** We strongly recommend that you do not modify the Background Task default values unless it is absolutely necessary. Configuring any of these parameters incorrectly can cause your entire system to malfunction (for example, CPU usage will go to 100%) and/or cause some tasks to perform poorly.

If you must change the parameter defaults, note the values before making your changes so if a malfunction occurs you can return to the original settings.

Keep in mind that the Background Task thread has the same priority as other threads in IWS (Drivers, Viewer, and so forth), which means that the operating system will not execute this task continuously for more than 20ms.

The Background Task executes the Recipe and Report worksheets when the Recipe and Report functions are called, respectively. Because the Recipe and Report functions are synchronous, once the Background Task starts executing the functions, it will not switch to another task (Alarm, Math, Scheduler, or Trend) until it completely executes the functions. Executing a Recipe or Report function usually takes a few milliseconds.

The Background Task must switch between the Alarm, Math, Scheduler, and Trend tasks. When Background Task switches to the Scheduler task, it will not switch to another task (Alarm, Math, or Trend) until all Scheduler worksheets are executed. After executing all Scheduler worksheets, the Background Task will not execute the Scheduler again until it receives the next Period message for the Scheduler task.

The Background Task applies the same behavior when executing the Alarm and Trend tasks — when the Background Task switches to the Alarm or Trend task, it will not switch to another task until it handles all pending messages. So, the Background Task will not execute the Alarm or Trend task again, until IWS generates the next Period message for each of these tasks.

The Background Task typically executes the Alarm, Scheduler, and Trend tasks in a few milliseconds. However, it can take longer to execute the Math task because it usually contains loops and complex scripts. Consequently, the mechanism used to execute the Alarm, Scheduler, and Trend tasks cannot be applied to the Math task.

The Background Task executes the Math task for no more than 10ms continuously before switching to other task (such as the Scheduler). The Background Task cannot execute the Math task again for the next 50ms; however, the Background Task can execute other tasks (Alarm, Recipe, Report, Scheduler, or Trend) during this 50ms period. After the Background Task executes all of the Math worksheets, it will not begin a new scan of the Math worksheets until IWS generates a new Period message for the Math task.

It is important to re-emphasize that this process was created to prevent 100% CPU usage all the time.

**Caution:** We recommend caution when using the Math function in a Scheduler worksheet or for a screen object (such as the Command animation).

When the Scheduler task executes a Math function, no other task can be executed by the Background Task until the Scheduler executes the entire Math worksheet called by the Math
function. This process can take several milliseconds or even seconds, depending on how you configured the script in the Math worksheet (especially for loops).

If you configure a Math function for a screen object, the Viewer stops updating the screen until the Viewer executes the entire Math worksheet called by the Math function.

If you must use the Math function for the Scheduler task or a screen object, we recommend using the following procedure to prevent process delays:

1. Specify one auxiliary tag with the value 1 and the Scheduler or Viewer task will send a message to the Tags database to update this tag value.
2. Configure the tag in the Execution field of the Math worksheet to be executed. When the Background Task scans the Math worksheet, IWS will execute the worksheet.
3. Reset the tag in the last line of the Math worksheet (write the value 0 to the auxiliary tag).

As a result, the Background Task will not execute the Math worksheet in the next scan unless the auxiliary tag is set to the value 1 again.
Installation

This section provides instructions for installing, starting, and uninstalling InduSoft Web Studio, EmbeddedView, and CEView.
Comparison of InduSoft Web Studio software components

The InduSoft Web Studio software suite actually comprises several individual components that can be installed on different computers to perform different functions.

### Comparison of InduSoft Web Studio software components

<table>
<thead>
<tr>
<th>Component</th>
<th>Functions</th>
<th>Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>InduSoft Web Studio</td>
<td>• Project development&lt;br&gt;• Tag integration&lt;br&gt;• Remote management&lt;br&gt;• Project runtime server&lt;br&gt;• Project runtime client</td>
<td>• Windows&lt;br&gt;• Windows Server&lt;br&gt;• Windows Embedded Standard</td>
</tr>
<tr>
<td>EmbeddedView (including Remote Agent)</td>
<td>• Agent to allow remote management&lt;br&gt;• Project runtime server (limited tag count)&lt;br&gt;• Project runtime client</td>
<td>Windows Embedded Standard</td>
</tr>
<tr>
<td>CEView (including Remote Agent)</td>
<td>• Agent to allow remote management&lt;br&gt;• Project runtime server (limited tag count)&lt;br&gt;• Project runtime client</td>
<td>Windows Embedded Compact</td>
</tr>
<tr>
<td>Web Tunneling Gateway (WTG)</td>
<td>Enables a public-facing web server to pass data between your project runtime server and web thin clients, when your project runtime server is located on a secure, internal network.</td>
<td>Microsoft IIS for Windows</td>
</tr>
<tr>
<td>Mobile Access Runtime</td>
<td>Enables your project runtime server to deliver HTML5-enhanced project screens to tablets and smartphones.</td>
<td>Microsoft IIS for Windows</td>
</tr>
<tr>
<td>Secure Viewer (a.k.a. InduSoft Thin Client)</td>
<td>Project runtime client, as a standalone application.</td>
<td>• Windows&lt;br&gt;• Windows Server&lt;br&gt;• Windows Embedded Standard&lt;br&gt;• Windows Embedded Compact</td>
</tr>
<tr>
<td>Web Thin Client (a.k.a. ISSymbol)</td>
<td>Project runtime client, as a browser plug-in.</td>
<td>Internet Explorer for Windows</td>
</tr>
</tbody>
</table>
The architecture of your finished IWS project depends on which components you install, where you install them, and how you connect them to each other.

In most cases, you should first install the full InduSoft Web Studio software on your primary workstation. Not only does it set up the project development environment on your computer, it also unpacks the rest of the components so that they can be installed on other computers.

There are separate, downloadable installers for the thin clients.
Install the full InduSoft Web Studio software

Install the full InduSoft Web Studio software on your Windows computer in order to develop IWS projects, or to use the computer as a project runtime server and/or client.

To install and run the full InduSoft Web Studio software, you must have:

- A Windows-compatible computer with a standard keyboard, pointer input (e.g., mouse, trackpad, or touchscreen), and SVGA-minimum display;
- One of the following Windows operating systems:
  - Microsoft Windows XP Service Pack 3
  - Microsoft Windows Vista Service Pack 2
  - Microsoft Windows 7 Service Pack 1
  - Microsoft Windows 8 or 8.1 (excluding Windows RT)
  - Microsoft Windows Server 2003 Service Pack 2
  - Microsoft Windows Server 2008 Service Pack 2
  - Microsoft Windows Server 2008 R2 Service Pack 1
  - Microsoft Windows Server 2012
  - Microsoft Windows XP Embedded Service Pack 3
  - Microsoft Windows Embedded Standard 7
  - Microsoft .NET Framework 3.5.1 (see note below);
  - Microsoft Internet Explorer 6.0 or later;
  - 2 GB free hard drive space or non-volatile memory; and
  - An Ethernet or Wi-Fi network adapter, for TCP/IP networking.

We recommend the Home Premium, Professional, Enterprise, and Ultimate editions of Windows, because they include Microsoft Internet Information Services (IIS) as a pre-installed feature that can be turned on. You can use IIS to make your projects accessible to thin clients and mobile devices. We do not recommend the Starter and Home Basic editions because they do not include IIS, but you can still use them if you do not plan to use web-based features.

The following items are optional but recommended:

- A DVD-ROM drive, to install the software from disc.
  This is optional because you can also download the installer to your computer.
- A USB or parallel port, to be used with hardkey licensing of the software.
  This is optional because softkey licensing is also available.
- Serial COM ports and adapters, to be used for direct communication with PLCs and other devices.
  This is optional because many newer device protocols use TCP/IP communication instead of serial communication.
- Microsoft IIS installed and turned on, to make your projects accessible to mobile devices. For more information, see the description of Mobile Access Runtime below.
  This is optional because you may choose not to install the Mobile Access Runtime feature.
- Microsoft Visual Studio Team Explorer 2010 installed, to enable workgroup collaboration and source control. For more information, see the description of Collaboration below.
  This is optional because you may choose not to install the Collaboration feature.

Note:

- You must have Administrator privileges on the computer in order to install software.
• Only Windows 7, Windows 8, Windows Server 2008 R2, and Windows Server 2012 are under what Microsoft calls "mainstream support", which means they are actively maintained and additional service packs may be released for them in the future. The rest of the listed operating systems are under what Microsoft calls "extended support", which means they are no longer actively maintained.

• InduSoft Web Studio can be installed on Windows XP 64-bit, but we do not support it.

• Even though you can install the full InduSoft Web Studio software on a Windows XP Embedded or Windows Embedded Standard 7 computer, if you do not plan to do project development on that computer and will use it only as a project runtime server and/or client, you should consider installing EmbeddedView instead. EmbeddedView does not support as many project tags as the full software, but it requires fewer system resources and it can be installed and managed remotely.

• Microsoft .NET Framework 3.5.1 is pre-installed on newer versions of Windows, but it may not be turned on. To turn it on, use the Programs and Features control panel (Start > Control Panel > Programs > Programs and Features > Turn Windows features on and off).

On older versions of Windows, Microsoft .NET Framework 3.5.1 must be installed separately. The InduSoft Web Studio installer will attempt to do this for you (see Step 2 below), but depending on your computer's security settings, the installation might fail without notice. If you experience problems later, while trying to run InduSoft Web Studio, you should at least confirm that Microsoft .NET Framework 3.5.1 was installed successfully. To do this in Windows XP or Windows Server 2003, use the Add/Remove Programs control panel (Control Panel > Add/Remove Programs).

Please note that Microsoft .NET Framework 4 does not include Microsoft .NET Framework 3.5.1.

To install the full InduSoft Web Studio software:

1. Do one of the following:
   • Download the zipped installer to your computer, either from our website (www.indusoft.com) or from another location on your network where you have previously saved it. Extract the zip archive, open the resulting folder, and then locate and double-click setup.exe.
   • Insert the installation disc into your DVD-ROM drive. If it does not autorun, manually run D:\indusoft.htm. When the page opens in your browser, follow the instructions for Product Installation.

The installation wizard runs and asks you to select a language for the installation.

2. Select a language from the list, and then click OK.

   This selection determines what language the installation wizard will use, as well as what language the project development application's user interface will be in. You can change the language of the user interface later, after you have installed the software.

   The wizard prepares for installation. During this step, it automatically installs SafeNet's Sentinel drivers (a part of the software licensing mechanism) and Microsoft .NET Framework 3.5.1.

3. On the Welcome page of the wizard, click Next to proceed with the installation.

4. On the License Agreement page, click either Yes to accept the agreement and proceed or No to refuse the agreement and exit the wizard.

5. On the Customer Information page, enter your user name and company name, and then click Next.

6. On the Choose Destination Location page, select the folder where the software should be installed, and then click Next.

   By default, the software will be installed at: C:\Program Files\InduSoft Web Studio v7.1\

7. On the Select Features page, select which software features and components you want to install on your computer, and then click Next.

   The following features are available:

   **Program Files**
   The main program files for the project development application, the project runtime server, and the project runtime client. This feature cannot be deselected.

   **Demos**
   Premade projects that demonstrate the capabilities of the InduSoft Web Studio software.
Hardkey Support
Additional drivers to support the use of hardkey licenses.

OPC Components
Additional components to communicate with other OPC-compatible devices. This includes OPC DA (a.k.a. OPC Classic), OPC UA, OPC .NET (a.k.a. OPC Xi), and OPC XML-DA.

PDF Printing
Software for printing run-time reports directly to PDF files.

Security System Device Driver
An additional keyboard driver to control user input for project security.

Symbol Library
A library of premade but configurable screen objects such as pushbuttons, toggle switches, gauges, dials, indicator lights, and so on.

Windows CE Runtime
Also called CEView — project runtime software (server/client, but not development) for Windows Embedded Compact on a variety of processors. Check the documentation for your specific device to see what processor it uses.

Note: Selecting this feature will not actually install CEView on your computer. It will simply unpack the installation files and copy them to your program folder, so that you can later install CEView on a Windows Embedded Compact device.

Windows Embedded Runtime
Also called EmbeddedView — project runtime software (server/client, but not development) for Windows Embedded Standard computers.

Note: Selecting this feature will not actually install EmbeddedView on your computer. It will simply unpack the installation files and copy them to your program folder, so that you can later install EmbeddedView on a Windows Embedded Standard computer.

Windows Mobile
Additional runtime software for older Windows Mobile devices.

Mobile Access Runtime
Additional software for Microsoft IIS that makes your project runtime accessible to mobile devices such as tablets and smartphones.

This feature requires that you have Microsoft IIS turned on and configured with ASP, ASP.NET, and ISAPI Extensions enabled. The InduSoft Web Studio installation wizard will attempt to verify that you do, and if you do not, it will not install this feature.

For more information, see Turn on Microsoft IIS for thin client access on page 571.

You do not need to install this feature at this time. You can install it later, after you have turned on IIS, or you can install it on another computer that is acting as your project runtime server. There is a separate Mobile Access Runtime installer (MobileAccessSetup.exe) that is unpacked with the rest of the software.

Note: To use Mobile Access Runtime, your InduSoft Web Studio software license must have the necessary add-on. You can still develop projects that include Mobile Access features, even without the license add-on, but mobile devices will not be able to access them.

Collaboration
Additional tools for workgroup collaboration and source control within the development environment.

This feature requires that you have Microsoft Visual Studio Team Explorer 2010 installed on the same computer. The InduSoft Web Studio installation wizard will attempt to verify that you do, and if you do not, it will not install this feature.
The Team Explorer module is included in some versions of Microsoft Visual Studio 2010, so if you already have Visual Studio installed on your computer, you might be able to select and install the Collaboration feature. However, if you do not have Visual Studio installed, or if you are not sure that your version of Visual Studio includes the Team Explorer module, you can separately download and install Team Explorer 2010 for free.


**Note:** Team Explorer 2012 and Team Explorer 2013 are not supported at this time for the collaboration client.

To use the Collaboration feature, you should also have Microsoft Visual Studio Team Foundation Server 2010 or 2012 running somewhere on your network, but if you do not, it will not prevent you from installing the Collaboration feature now.

**Note:** Team Foundation Server 2013 is not supported at this time for the collaboration server.

To use the Collaboration feature, your InduSoft Web Studio software license must have the Collaboration add-on. To purchase the add-on, please contact your software vendor.

8. On the **Ready To Install** page, click **Install**. The installation proceeds.

9. On the final page, click **Finish**. The wizard closes.

When you have finished the installation, you should find the InduSoft Web Studio software in your **Start** menu at **Start** > **All Programs** > **InduSoft Web Studio v7.1**. It includes the following components:

**InduSoft Web Studio v7.1+SP3**

The project development application, project runtime server, and project runtime client.

**Register**

A utility that manages your InduSoft Web Studio software license.

**StartUp**

A shortcut that automatically starts the project runtime server and runs the most recent project.

**Remote Agent**

A utility that allows InduSoft Web Studio running on other computers to connect to your computer and send projects to it.

**Quick Start Guide**

A brief guide to installing and using the project development application, including a tutorial for building a simple project.

**Help Manual**

A complete technical reference and user guide for the InduSoft Web Studio software.

**Release Notes**

A list of changes in the the InduSoft Web Studio software.

There should also be a shortcut icon on your desktop.

**Note:** You might encounter the following error message during installation: "Error 1628: Failed to complete script based install." For more information about this error and how to resolve it, go to: consumerdocs.installshield.com/selfservice/viewContent.do?externalId=Q108464

To run the software, do one of the following:

- Double-click the shortcut icon on your desktop; or
• Click Start > All Programs > InduSoft Web Studio v7.1 > InduSoft Web Studio v7.1.
Install EmbeddedView or CEView on a target device

Install EmbeddedView on a Windows Embedded Standard computer, or install CEView on a Windows Embedded Compact device, to use it as a project runtime server and/or client.

You must have already installed the full InduSoft Web Studio software on your computer, either from the installation disc or from the downloadable installer, because the redistributable EmbeddedView and CEView software is included in the InduSoft Web Studio program folder.

To install and run EmbeddedView or CEView, you must have:

- A Windows Embedded-compatible computer or device (a.k.a. the target device);
- A Windows Embedded Standard or Windows Embedded Compact operating system that is currently supported by Microsoft, which at this time includes:
  - Microsoft Windows XP Embedded Service Pack 3
  - Microsoft Windows Embedded Standard 7
  - Microsoft Windows Mobile or Windows CE 5.x/6.x
  - Microsoft Windows Embedded Compact 7
- 100 MB of free hard drive space or non-volatile memory (flash card or hard disk) for the runtime. More space might be required depending on your project size.
- 32 MB of free RAM. More memory might be required depending on your project size.
- An Ethernet or Wi-Fi network adapter, for TCP/IP networking.

The following items are optional but recommended:

- A USB port; and
- Serial COM ports and adapters, to be used for direct communication with PLCs and other devices.

This is optional because many newer device protocols use TCP/IP communication instead of serial communication.

Installing EmbeddedView or CEView on a target device is actually a two-part procedure. First, you will copy the Remote Agent utility to the target device and then run it. Remote Agent allows you to connect from the project development application to the target device. And second, through this connection, you will install the rest of the EmbeddedView or CEView software.

Note: If EmbeddedView or CEView is pre-installed on the target device, you may skip this entire task. Many hardware manufacturers pre-install Remote Agent on their devices, as part of a larger InduSoft Web Studio package.

To install EmbeddedView or CEView:

1. Turn on the target device and make sure it is connected to your TCP/IP network.
   If Remote Agent is pre-installed on the target device, it will automatically run at start up, and you may skip the next step. Many hardware manufacturers pre-install Remote Agent on their devices, as part of a larger InduSoft Web Studio package.

2. Copy the Remote Agent utility to the target device and then run it:
   a) Locate the correct version of the Remote Agent utility (CEServer.exe) for the target device. All versions are stored in your InduSoft Web Studio program folder.

   Remote Agent for Windows Embedded Standard is located at:

   C:\Program Files\InduSoft Web Studio v7.1\Redist\WinEmbedded\Bin\CEServer.exe

   Remote Agent for Windows Embedded Compact is located at:

   C:\Program Files\InduSoft Web Studio v7.1\Redist\WinCE 5.0\processor\Bin\CEServer.exe
...where processor is the specific processor used by the target device. For more information, consult the manufacturer’s documentation.

b) Copy CEServer.exe to the target device by either downloading it over the network, transferring it on a USB flash drive, or syncing it with Microsoft ActiveSync. (ActiveSync is also known as Windows Mobile Device Center in Windows Vista or Zune Software in Windows 7.) You may save the file anywhere you want on the target device, as long as it is in permanent (i.e., non-volatile) memory and it is not in the root folder (i.e., C:\ or \).

c) Set Remote Agent to automatically run at start up.
   You can do this on most Windows Embedded Compact devices by creating a link to it in \Windows \Startup. If this does not work on your device, please refer to the manufacturer’s documentation.

d) Run Remote Agent.
   The Remote Agent window is displayed.

3. Configure the communication settings in Remote Agent:
   a) Click Setup.
      The Setup dialog box is displayed.

   b) Select the appropriate connection type: Serial Port or TCP/IP. If you select Serial Port, also select the COM port and review the advanced settings.
      If you are already connected to the target device via ActiveSync, you do not need to select another connection at this time. However, keep in mind how the target device will actually be used during project run time.

   c) Click OK.

   d) If you selected TCP/IP for the device connection, note the IP address.

   e) Make sure that you leave Remote Agent running on the target device.

4. Use the Remote Management tool to connect to the target device:
   a) Run the project development application on your computer.

   b) On the Home tab of the ribbon, in the Remote Management group, click Connect.
The *Remote Management* dialog box is displayed.

![Remote Management Dialog Box](image)

- Select the appropriate connection type for the target device: Network IP, Serial Port, or Microsoft ActiveSync. If you select Network IP, also type the IP address of the target device. If you select Serial Port, also select the COM port and verify the advanced serial communication settings.

- Click **Connect**.

If you are successfully connected, the connection status is shown in the **Status** box and the target device's specifications are shown in the **Platform** box.

If you are not connected, check both the connection settings and the physical connections. In particular, if you have selected Network IP as the connection type, check to make sure that port 4322 is open on any firewalls between your computer and the target station, including on the target station itself.

### Note:
In some cases, the Remote Management tool may not be able to connect via Microsoft ActiveSync to a device running Windows CE 6.0 or later. This is because of a problem in the default configuration of Windows CE 6.0. You can fix the problem by using a small utility that is included with InduSoft Web Studio. The utility is located at:

```
C:\Program Files\InduSoft Web Studio v7.1\Redist\ActiveSyncUnlock.exe
```

Copy this file to the device using the stand-alone version of Microsoft ActiveSync and then execute the file on the device. It does not matter where on the device the file is located. When this is done, try again to use the Remote Management tool to connect to the device.

If you still cannot connect via Microsoft ActiveSync, empty the device’s \Temp directory and try again.

5. Install the rest of the EmbeddedView or CEView software on the target device:
   - a) In *Remote Management*, click **Install system files**.

   When the installation is finished, the target device’s updated status is displayed in the **Status** box.

   With EmbeddedView or CEView installed on the target device, you can now use it as a project runtime server and/or client.

### Note:
The following run-time features are supported by the full InduSoft Web Studio software but not by EmbeddedView and CEView:

- DDE task worksheets.
- ODBC task worksheets and built-in functions. (ADO interface is supported.)
- Third-party .NET controls.
- OPC HDA Server.
- OPC UA and OPC .NET clients.
• Some minor graphic features, such as fill effects for shapes other than rectangle, picture formats other than BMP or JPG, and Ellipse style.
• Some built-in functions. For more information, see the individual function descriptions.

Please keep these limitations in mind as you develop your projects.
License Settings

Both hardkey and softkey licenses have the following settings:

![Example of license settings]

**Version**

The overall version of the InduSoft Web Studio software (e.g., 7.1). This does not include the service pack, if any.

**Drivers**

The number of direct communication drivers that can be simultaneously enabled.

**Product Type**

Specifies which features and restrictions are enabled for the application (e.g., the maximum number of tags per project). Consult your software vendor about which product types are available and which features are enabled for each type.

**Execution Mode**

Specifies one of the following options:

- **Engineering Only**: Can develop a IWS project and then run it for short-term testing only. You cannot use this license as a long-term run-time license.
- **Runtime Only**: Can run a IWS project for an unlimited period. You cannot use this license to develop or modify the project.
- **Engineering + Runtime**: Can develop a IWS project and then run it for an unlimited period.

For more information, see Execution modes on page 48.

**Importers**

The list of third-party applications that can be handled by the Import Wizard.

**Thin Clients**

The number of Web Thin Clients that can simultaneously connect to the project runtime server. One connection is included with every license. Contact your vendor to purchase additional connections.

**Secure Viewers**

The number of Secure Viewer thin clients that can simultaneously connect to the project runtime server. One connection is included with every license. Contact your vendor to purchase additional connections.

**SMA Clients**

The number of Mobile Access and Mobile Access Tabular thin clients that can connect simultaneously to the server. One connection is included with every license. Contact your vendor to purchase additional connections.

**Product Versions**

InduSoft Web Studio, EmbeddedView, and CEView should all have the same version number, which uses the syntax X.Y+SPWW (e.g., InduSoft Web Studio v7.1+SP3), where:

- X: Major version number
- Y: Minor version number
- SPWW: Service pack number
• X represents the **Family version**. The Family version changes only when major enhancements are added to the product technologies and concepts.

• Y represents the **Sub-version**. The Sub-version changes when minor enhancements and/or new features are added to the product.

• WW represents the **Service Pack**. The Service Pack version changes when you must install add-on packages to accomplish the following:
  - Upgrade files for the version previously installed
  - Fix bugs in the product (showstoppers and no-workarounds)
  - Provide minor enhancements before releasing the next version of the product

Each Service Pack release supersedes the previous Service Pack release. For example, SP2 includes all the contents of SP1 and all newly upgraded files, bug fixes, and enhancements. SP3 includes all the contents of SP2 and all new upgraded files, bug fixes, enhancements and so on.

InduSoft Web Studio, EmbeddedView, and CEView all can run projects that were developed in previous versions of the product. However, previous versions cannot run projects that were developed or modified in newer versions. Opening and modifying a project in a newer version will update the project to that version.

For example, you cannot run version 6.1 projects using IWS version 6.0 but you can run version 6.0 projects with IWS version 6.1.

**Important:** We issue each license for a specific **Family** version and **Sub-version** (X,Y), and the license is valid for that version (including Service Packs) only. The license is not valid for a newer **Family** version or **Sub-version** of the product. Therefore, if you install a new version of IWS, EmbeddedView, or CEView, then you must upgrade your license to the new version being installed. If you install a Service Pack only, then you do not need to upgrade your license.

### Execution modes

InduSoft Web Studio, EmbeddedView, and CEView support the following execution modes:

<table>
<thead>
<tr>
<th>Execution Mode</th>
<th>InduSoft Web Studio</th>
<th>EmbeddedView / CEView</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation Mode</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Demo Mode</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Licensed for Engineering Only</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Licensed for Runtime Only</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Licensed for Engineering + Runtime</td>
<td>#</td>
<td>#</td>
</tr>
</tbody>
</table>

**Evaluation Mode**

Enables all of the product’s engineering and runtime features.

The first time you install InduSoft Web Studio on a computer, the product runs for forty (40) hours in Evaluation Mode. This evaluation period includes any time you run a product module (engineering or runtime). You can use this evaluation period continuously or not; for example, 10 hours a day for 4 days, or 5 hours a day for 8 days, or 10 hours a day for 3 days plus 5 hours a day for 2 days, and so on.

After running for 40 hours in the Evaluation Mode, the evaluation period ends and the program automatically converts to Demo Mode until you apply a valid license (hardkey or softkey). You cannot reactivate Evaluation Mode, even if you reinstall the software on your computer.

**Note:** Each version of InduSoft Web Studio has an evaluation period that is independent of every other version. For example, if your InduSoft Web Studio v7.0 evaluation period has expired and you are running in Demo Mode because you have not installed a license, when you install InduSoft Web Studio v7.1+SP3 on the same computer, the newer version will begin its own 40-hour evaluation period and the older version will continue running in Demo Mode.
Demo Mode

Allows you to download projects to remote stations and to run projects for testing or demonstration purposes. You can execute runtime tasks and use the debugging tools (LogWin and Database Spy), but they shut down automatically after running for two hours continuously. You can restart the Demo Mode again and run for another two hours, and so on.

You cannot create or modify screens, worksheets, or project settings in Demo Mode.

Licensed for Engineering Only

Enables all development options for an unlimited time.

This mode also allows you to continuously run the runtime tasks and debugging tools (Database Spy, Output window, and LogWin module) for 72 hours. After that period, these tasks shut down, but you can restart them and run for another 72 hours, and so on. You can use this license for development and testing only.

Licensed for Runtime Only

Enables all runtime tasks and debugging tools (Database Spy, Output window, and LogWin module) for unlimited time, but you cannot create or modify screens and/or worksheets.

The menu options available in Runtime Only mode are the same as the options listed for Demo Mode (see previous table).

Licensed for Engineering + Runtime

Enables all development options, runtime tasks, and debugging tools (Database Spy, Output window, and LogWin module) for an unlimited time.

**Note:** The Remote Management tool is always available, regardless of the execution mode, so that you can upload files from or download files to remote stations.

To see which execution mode you are currently running, click About on the Help tab of the ribbon; the About dialog shows the execution mode, including the time remaining if you are in Evaluation Mode.
About hardkey licenses

An encapsulated chip that must be physically connected to the computer's parallel port (LPT1) or USB interface.

The software license resides in the hardkey, and you cannot share this license simultaneously with more than one other copy of software in the network. If you connect the hardkey to another computer, then you effectively transfer the license to that computer.

Using the parallel port hardkey does not prevent you from connecting another device — such as a printer — to the port. The hardkey should be electronically transparent to other devices connected to the parallel port. You simply connect the hardkey to the computer and then connect the printer cable to the hardkey. However, you may encounter problems if you install more than one hardkey (for different products) on the same parallel port.

**Note:** Be careful when installing or removing a hardkey from the computer’s parallel port. We strongly recommend that you turn off the computer and disconnect it from the power supply before installing or removing a hardkey.

On the other hand, while using the USB hardkey, the USB port cannot be shared with any other device.

**Install a new hardkey license**

Install a new hardkey license for InduSoft Web Studio, EmbeddedView, or CEView.

Before you begin this task, make sure the appropriate software has been installed on your computer or device. For more information, see Installation on page 34.

To install a new hardkey license:

1. On the computer or device where you have installed the software, connect the hardkey to the appropriate port (e.g., USB, LPT1).

   **Note:** If you are using a parallel port hardkey, we strongly recommend that you turn off the computer and disconnect it from the power supply before you connect or remove the hardkey. This is because when the computer is on, the parallel port carries enough power to damage the hardkey.

2. Run the software.

If the software recognizes the hardkey, it will run normally without any alert messages.

However, if it does not, try the following:

- For InduSoft Web Studio, use the Protection Manager utility program to make sure the software is set to check for a hardkey. For more information, see Upgrade an existing hardkey license for InduSoft Web Studio on page 50.

- For EmbeddedView or CEView, if the software does not recognize the hardkey, then it will automatically check for a softkey. If it does not find a softkey either, then use the Remote Agent utility program on the Windows Embedded device to diagnose the problem. For more information, see Upgrade an existing hardkey license for EmbeddedView or CEView on page 51.

Please note that not all Windows Embedded devices can recognize the USB hardkey technology used by InduSoft; our internal testing has shown hat only Windows Embedded devices that fully support USB flash memory (a.k.a. “thumb drives” or “memory sticks”) will recognize our USB hardkey. Check with the manufacturer of your Windows Embedded device.

**Upgrade an existing hardkey license for InduSoft Web Studio**

Before you begin this task, make sure the hardkey is connected to the Windows computer where you installed the full InduSoft Web Studio software.

Also, you should have already purchased a valid upgrade to your InduSoft Web Studio license. You may purchase the upgrade when you send in the site code (see below), but doing so might increase your downtime. For more information, contact your InduSoft Web Studio software vendor. You can update any license setting
(e.g., product type, number of thin clients) or upgrade the software to a new version. The cost of the upgrade depends on the difference between the current and upgraded license settings.

**Note:** These instructions apply only to upgrading a hardkey license for the full InduSoft Web Studio software. To upgrade a hardkey license for the EmbeddedView or CEView runtime software, see Upgrade an existing hardkey license for EmbeddedView or CEView on page 51.

To upgrade an existing hardkey license for InduSoft Web Studio:

1. In InduSoft Web Studio, stop the running project (if any), and then exit the program.
2. Click **Start > All Programs > InduSoft Web Studio v7.1 > Register**. The Protection Manager program window is displayed.

   ![Protection Manager](image)

   **Protection Manager**

3. Select **Hardkey** if it is not already selected, and then click **Check**.
   
   If you have a valid hardkey license installed, the **Hardkey Settings** dialog box is displayed with the license settings on that hardkey.

   ![Hardkey Settings](image)

   **Checking the hardkey settings**

   Otherwise, if you do not have a valid hardkey license installed, an alert message is displayed.

4. Click **Change License**.
   
   The **Change License** dialog box is displayed with a randomly generated site code.

5. Send the site code to your software vendor.
   
   Typically, you will paste the site code into an email to your software vendor.

6. When you receive the corresponding site key from your software vendor, type or paste it in the **Site Key** box, and then click **Authorize**. (You will be prompted to confirm.)
   
   The new license settings are written to the hardkey, and then a confirmation message is displayed.

   If the new site key is not validated, an error message is displayed. If this happens, double-check that you typed the site key correctly. If you typed it correctly and still get an error message, contact your software vendor for further assistance.

**Upgrade an existing hardkey license for EmbeddedView or CEView**

This task assumes that you have both a Windows computer with the full InduSoft Web Studio software installed and a Windows Embedded device with the EmbeddedView or CEView runtime software installed.
You will actually perform the upgrade on the Windows computer using a utility program that is installed with InduSoft Web Studio.

Also, if InduSoft Web Studio is running on its own hardkey license, you should exit the program and then disconnect that hardkey before you begin this task. You should do this in order to avoid issues that may result from having two hardkeys connected to the same computer.

Lastly, you should have already purchased a valid upgrade to your EmbeddedView or CEView license. You can purchase the upgrade when you send in the site code (see below), but doing so might increase the downtime for your project. For more information, contact your InduSoft Web Studio software vendor. You can update any license setting (e.g., product type, number of thin clients) or upgrade the software to a new version. The cost of the upgrade depends on the difference between the current and upgraded license settings.

To upgrade an existing hardkey license for EmbeddedView or CEView:

1. On the Windows Embedded device, in Remote Agent, stop the running project (if any), and then exit the program.
2. Disconnect the hardkey from the Windows Embedded device, and then connect it to the Windows computer.
3. On the Windows computer, find the utility program Site Key Writer in the InduSoft Web Studio program folder, and then run it.
   - The program should be located at: C:\Program Files\InduSoft Web Studio v7.1\Bin\SitekeyWriter.exe
   - The Site Key Writer dialog box is displayed with a randomly generated site code.
4. Send the site code to your software vendor.
   - Typically, you will paste the site code into an email to your software vendor.
5. When you receive the corresponding site key from your software vendor, type or paste it in the Site Key box, and then click Accept. (You will be prompted to confirm.)
   - The new license settings are written to the hardkey, and then a confirmation message is displayed.
6. Disconnect the hardkey from the Windows computer, and then connect it to the Windows Embedded device.
7. On the Windows Embedded device, run Remote Agent and restart your project.
8. Optional: Confirm the new license settings.
   a) In Remote Agent, click Setup.
      - The Setup dialog box is displayed.
   b) In the Setup dialog box, click License.
      - The License dialog box is displayed with the license settings that are on the connected hardkey.

If the new site key is not validated, an error message is displayed. If this happens, double-check that you typed the site key correctly. If you typed it correctly and still get an error message, contact your software vendor for further assistance.
About softkey licenses

When you install InduSoft Web Studio, EmbeddedView, or CEView, the program generates a unique site code. You can send this site code to your software vendor, who will then generate a site key to match your site code. You can then use the site key to install the license on your Windows computer or Windows Embedded device, as opposed to having the license stored on a hardkey.

**Note:** When you use a softkey, the license is recorded in the computer or device’s permanent memory. If the computer is damaged or lost, you will lose the license.

**Install or upgrade a softkey license for InduSoft Web Studio**

Install or upgrade a softkey license for the full InduSoft Web Studio software running on a Windows computer.

Before you begin this task, make sure the appropriate software has been installed on your computer or device. For more information, see *Installation* on page 34.

Also, you should have already purchased a valid license or upgrade. You may purchase it when you send in the site code (see below), but doing so might increase your downtime. For more information, contact your InduSoft Web Studio software vendor. You can update any license setting (e.g., product type, number of thin clients) or upgrade the software to a new version. The cost of the upgrade depends on the difference between the current and upgraded license settings.

**Note:** These instructions apply only to upgrading a softkey license for the full InduSoft Web Studio software. To upgrade a softkey license for the EmbeddedView or CEView runtime software, see *Install or upgrade a softkey license for EmbeddedView or CEView (Local)* on page 54 or *Install or upgrade a softkey license for EmbeddedView or CEView (Remote)* on page 56.

These instructions apply both to installing a new softkey license and to upgrading an existing softkey license; whatever license you apply will overwrite the previous one, if any.

To install or upgrade a softkey license for InduSoft Web Studio:

1. In InduSoft Web Studio, stop the running project (if any), and then exit the program.

2. Click Start > All Programs > InduSoft Web Studio v7.1 > Register.
   The Protection Manager program window is displayed.

3. Select Softkey if it is not already selected, and then click Check.
The Softkey Settings dialog box is displayed. If you have a valid softkey license installed, then the dialog will show those settings. Otherwise, if you do not have a softkey, then the dialog will simply say "License not found."

Checking the softkey settings

4. Click Change License.
   The Change License dialog box is displayed with a randomly generated site code.

5. Send the site code to your software vendor.
   Typically, you will paste the site code into an email to your software vendor.

6. When you receive the corresponding site key from your software vendor, type or paste it in the Site Key box, and then click Authorize. (You will be prompted to confirm.)
   The new license settings are saved on the computer, and then a confirmation message is displayed.

   If the new site key is not validated, an error message is displayed. If this happens, double-check that you typed the site key correctly. If you typed it correctly and still get an error message, contact your software vendor for further assistance.

Install or upgrade a softkey license for EmbeddedView or CEView (Local)

Use the Remote Agent program running on your Windows Embedded device to locally install or upgrade the softkey license for the EmbeddedView or CEView runtime software.

Before you begin this task, make sure the appropriate software has been installed on your computer or device. For more information, see Installation on page 34.

Also, you should have already purchased a valid license or upgrade. You may purchase it when you send in the site code (see below), but doing so might increase your downtime. For more information, contact your InduSoft Web Studio software vendor. You can update any license setting (e.g., product type, number of thin clients) or upgrade the software to a new version. The cost of the upgrade depends on the difference between the current and upgraded license settings.

Tip: You can purchase some Windows Embedded devices with an EmbeddedView or CEView softkey license already installed. Ask your software vendor about this possibility.

These instructions apply both to installing a new softkey license and to upgrading an existing softkey license; whatever license you apply will overwrite the previous one, if any.

To install or upgrade a softkey license locally:
1. On the Windows Embedded device, run Remote Agent if it is not already running.

![Remote Agent program window](image)

Remote Agent program window

2. Click Setup.
   The Setup dialog box is displayed.

![Setup dialog box](image)

Setup dialog box

3. Click License.
   The License dialog box is displayed. If you have a valid softkey license installed, then the dialog box will show those settings. Otherwise, if you do not have a softkey, then the dialog box will simply say "License not found."

![License dialog box](image)

License dialog box

4. Click Change License.
   The Change License dialog box is displayed with a randomly generated site code.

![Change License dialog box](image)

Change License dialog box

5. Send the site code to your software vendor.
   Typically, you will paste the site code into an email to your software vendor.

6. When you receive the corresponding site key from your software vendor, type or paste it in the Site Key box, and then click Authorize. (You will be prompted to confirm.)
   The new license settings are saved on the device, and then a confirmation message is displayed.
If the new site key is not validated, an error message is displayed. If this happens, double-check that you typed the site key correctly. If you typed it correctly and still get an error message, contact your software vendor for further assistance.

**Install or upgrade a softkey license for EmbeddedView or CEView (Remote)**

Use the Remote Management tool in InduSoft Web Studio to remotely install or upgrade the softkey license for the EmbeddedView or CEView runtime software.

This task assumes that you have both a Windows computer with the full InduSoft Web Studio software installed and a Windows Embedded device with the EmbeddedView or CEView runtime software installed, that both stations are connected to your network, and that Remote Agent is running on the Windows Embedded device.

Also, you should have already purchased a valid license or upgrade. You may purchase it when you send in the site code (see below), but doing so might increase your downtime. For more information, contact your InduSoft Web Studio software vendor. You can update any license setting (e.g., product type, number of thin clients) or upgrade the software to a new version. The cost of the upgrade depends on the difference between the current and upgraded license settings.

**Tip:** You can purchase some Windows Embedded devices with an EmbeddedView or CEView softkey license already installed. Ask your software vendor about this possibility.

These instructions apply both to installing a new softkey license and to upgrading an existing softkey license; whatever license you apply will overwrite the previous one, if any.

To install or upgrade a softkey license remotely:

1. In InduSoft Web Studio, on the Home tab of the ribbon, in the Remote Management group, click Connect. The Remote Management dialog box is displayed:

2. In the Target Station group, do one of the following to configure the connection to the target device running EmbeddedView or CEView:
   - Click Network IP, and then type the IP address of the target device;
   - Click Serial Port, and then select a port from the list provided; or
   - Click Microsoft ActiveSync.

   **Tip:** TCP/IP links provide better communication performance than serial links.

3. When the Connect button becomes active, click it to connect to the target device. The Status and Platform boxes show whether you have successfully connected to the target device.

4. When you have connected, click the Embedded License tab.
The current license settings are displayed, if any, as well as a randomly generated site code for the target device.

5. Send the site code to your software vendor.
   Typically, you will paste the site code into an email to your software vendor.

6. When you receive the corresponding site key from your software vendor, type or paste it in the **Site Key** box, and then click **Send**. (You will be prompted to confirm.)
   The new license settings are saved on the target device, and then a confirmation message is displayed.

If the new site key is not validated, an error message is displayed. If this happens, double-check that you typed the site key correctly. If you typed it correctly and still get an error message, contact your software vendor for further assistance.

Also, after the new license settings are saved on the Windows Embedded device, be sure to save its registry settings. If you do not save these settings, you will lose the license after rebooting the device. For more information about how to save the registry settings, see the manufacturer's documentation.
About License Server

Use the License Server feature to manage multiple IWS licenses and serve them over your network. The licenses are loaded onto a single hardkey, and then that hardkey is connected to a network server running the required license management software.

What do I need to use the License Server feature?

At this time, the License Server feature only works with USB hardkeys supplied by Wibu Systems (www.wibu.com). That is because the License Server feature is built on Wibu Systems' CodeMeter Runtime software, which provides the necessary hardkey drivers, license management, and client-server communication framework.

To make sure that you have the correct hardkey type and license settings, please contact your InduSoft Web Studio software vendor.

The CodeMeter Runtime software is automatically installed with InduSoft Web Studio. Every computer, both client and server, must have it in order to take advantage of the License Server feature. If you want to set up another computer — one on which InduSoft Web Studio has not been installed — as your license server, you must manually download and install the CodeMeter Runtime software.

What designates a computer as a license server is whether the option Run Network Server is selected in the CodeMeter Runtime software settings.

How does the License Server feature work?

First, a computer on the network is set up to be the license server. The computer may or may not also have InduSoft Web Studio installed and licensed. For example, it is often convenient to have your project development workstation and license server be the same computer, so that you can both send projects to target stations and manage the licenses for those stations, but it is not necessary.

Setting up the license server is simply a matter of making sure the CodeMeter Runtime software is installed, connecting the Wibu-type USB hardkey that contains the licenses, confirming that the license settings are correct, and then selecting the option Run Network Server in the CodeMeter Runtime software settings. For more information, see Configure CodeMeter Runtime to serve licenses on page 59.

Then, on every other computer that will get its license from the license server, InduSoft Web Studio's Protection Manager is used to enable the License Server feature and then connect to the license server. Doing so overrides the normal hardkey license settings on that computer. For more information, see Configure Protection Manager to get license on page 61.

Note: Only IWS project development workstations (Engineering) and project runtime servers (Runtime) need to be licensed. Thin clients do not need to be licensed separately, because a Runtime license includes the number of thin clients that may connect to the project runtime server.

What if the server connection is lost?

If, for whatever reason, InduSoft Web Studio loses its connection to the license server, it will continue operating as previously licensed for a grace period of up to 60 minutes.

During this grace period, you can use the function GetLicenseGracePeriod to check the license and take appropriate action. For example, you can configure a Button screen object with a Command animation to execute the following script:

```vbscript
Dim remainingGracePeriod
remainingGracePeriod = $GetLicenseGracePeriod()
If (remainingGracePeriod = -1) Then
    MsgBox "License OK"
Else
    MsgBox "Running on Grace Period! [" + remainingGracePeriod + "] minute(s) remaining"
End If
```
When a user clicks/taps the button during run time, the license is checked and an appropriate message is displayed.

If the grace period expires and InduSoft Web Studio has not either reestablished its connection to the license server or received another license, the program switches to Demo Mode. For more information, see Execution modes on page 48.

Please note that when InduSoft Web Studio does reestablish its connection to the license server, the grace period does not immediately reset to the full 60 minutes. Instead, InduSoft Web Studio must remain connected to the license server for at least twice as long as the connection was lost. So, for example, if the connection was lost for 38 minutes, InduSoft Web Studio must remain connected for 76 minutes. After that time, the grace period is reset.

Configure CodeMeter Runtime to serve licenses

Configure the CodeMeter Runtime software to manage licenses stored on a Wibu-type USB hardkey and serve them over your network.

Before you begin this task, you must have the following:

• A computer on your network that will act as the license server;
• The CodeMeter Runtime software installed on that computer; and
• A Wibu-type USB hardkey that contains your InduSoft Web Studio licenses.

A validated version of CodeMeter Runtime is automatically installed with the full InduSoft Web Studio software, so in most cases, InduSoft Web Studio should be installed on the computer that will be your license server. For more information, see Install the full InduSoft Web Studio software on page 37.

You do not need to license InduSoft Web Studio to run on that computer in order to use CodeMeter Runtime, although you may do so if the computer will also be used as a project development workstation and/or project runtime server. For more information, see Execution modes on page 48.

Alternatively, if your license server will be another computer on which you do not want to install InduSoft Web Studio, you must download and install CodeMeter Runtime on that computer. You can download the software from the Wibu Systems website at: www.wibu.com/downloads-user-software.html

CodeMeter Runtime itself does not require any license to run, because it can only be used to manage Wibu-type USB hardkeys.

Note: InduSoft Web Studio has been validated to work only with CodeMeter Runtime for Windows, and InduSoft does not directly support any version of CodeMeter Runtime. If you need help with the CodeMeter Runtime software, especially with installing CodeMeter Runtime for Mac or Linux, contact Wibu Systems at: www.wibu.com/support-downloads.html

To configure CodeMeter Runtime to serve licenses:

1. On the computer that will be your license server — that is, the computer where CodeMeter Runtime has been installed — connect the Wibu-type USB hardkey to the computer's USB port.
2. On the same computer, open a web browser such as Internet Explorer, and then go to: http://localhost:22350/

Note: Even though you might have turned on Microsoft IIS on this computer, especially if you plan to use it as your IWS project runtime server, it does not need to be turned on for you to go to this localhost address. The CodeMeter Runtime software runs as a separate process on port 22350.
The home page of CodeMeter WebAdmin is displayed.

3. On the navigation bar, click Configuration, and then on the Configuration tab, click Server. The Configuration > Server page is displayed.

4. Select Run Network Server, and then click Apply.

5. On the navigation bar, click Server, and then on the Server tab, click Cluster. The Server > Cluster page is displayed.

6. Look at the page and verify that your Wibu-type USB hardkey is properly connected and providing the correct license information.

7. Close the web browser.

CodeMeter Runtime is now providing the licenses to your network. You should also now be able to access CodeMeter WebAdmin across the network at: http://hostname or IP address:22350/
If you cannot access the license server from another computer, make sure that port 22350 is open on your network firewall.

**Configure Protection Manager to get license**

Configure InduSoft Web Studio's Protection Manager to get its runtime license from a license server on your network.

Before you begin this task, you must have a properly configured license server available on your network. For more information, see Configure CodeMeter Runtime to serve licenses on page 59. Please note the host name or IP address of that license server, just in case.

You must also have Administrator privileges on the computer where you are configuring Protection Manager. This task applies to every installation of InduSoft Web Studio that will get its license from the license server.

To configure Protection Manager:

1. If InduSoft Web Studio is running, stop your project (if it is running) and exit the application.
2. In Windows, choose **Start > All Programs > InduSoft Web Studio v7.1 > Register**. The **Protection Manager** dialog is displayed.

3. Click **License Server**. If the **License Server** button is not available, then you must first select **Hardkey** in the **Protection Type** group. The License Server feature cannot be enabled if a softkey license is already applied. The **License Server Configuration** dialog is displayed.

4. Select **Enable Server**.
5. In the **Mode** list, select one of the following options:

   **Option** | **Description**
   --- | ---
   **Local License** | Only check for a locally installed hardkey license. This is the default mode when **Enable Server** is not selected. For more information, see About hardkey licenses on page 50.
   **Local/Network** | Check first for a locally installed hardkey license, and if one is found, then use it. Otherwise, check the network for a license server.
### Licensing

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network/Local</td>
<td>Check the network first for a license server, and if one is found, then use it. Otherwise, check for a locally installed hardkey license.</td>
</tr>
<tr>
<td>Network</td>
<td>Only check the network for a license server.</td>
</tr>
</tbody>
</table>

This check is performed continuously while InduSoft Web Studio is running, which allows you some flexibility in how and when you use different types of licenses.

For example, a particular installation of InduSoft Web Studio operates as runtime-only, no development, for most of the time, and it get its Runtime-only license from the license server. However, if you want to log on occasionally and make changes to the project, then you may connect your personal hardkey that contains an Engineering + Runtime license.

In this case, you would select **Local/Network**: if your personal hardkey is connected, then InduSoft Web Studio will use it, otherwise it will get its license from the license server.

6. If you want to specify a particular license server, rather than automatically use the first license server found on the network, then do the following:
   a) Clear the **Automatic** check box.
   b) In the **Host** box, type the hostname or IP address of the license server.

   You do not need to include the port number unless you have changed it from the default port 22350.

   **Note:** When **Automatic** is selected, the CodeMeter Runtime software first scans its locally configured Server Search List, and if it cannot find one of those servers, then it does a UDP broadcast to look for any license server on the network. (UDP broadcasts are restricted to the local area network.) For more information about configuring the Server Search List, see the CodeMeter Runtime software documentation.

7. Click **Test**. If Protection Manager can get a valid license using the given settings, then a confirmation message is displayed.

8. Click **OK** to accept the settings and close the **License Server Configuration** dialog, and then click **Close** to close the Protection Manager.

If you cannot access the license server from this computer, then make sure that port 22350 is open on your network firewall.
The Development Environment

InduSoft Web Studio incorporates a modern, Ribbon-based Windows interface to provide an integrated and user-friendly development environment.
Application button

The Application button opens a menu of standard Windows application commands like New, Open, Save, Print, and Close.

![Application button opens menu of commands](image)

Recent Projects

The Recent Projects area of the Application menu lists the most recently opened projects. To open one of the listed projects, simply click it.

New

The New command on the Application menu is used to create a new worksheet file or project. The New dialog (see the following figures) contains two tabs:

- File tab: Select this tab to create new worksheets or screens for an open project.
- Project tab: Select this tab to create a new project.

Instructions for creating new files and projects follow.

Creating a New File

To create a new worksheet or screen:
1. Click the **File** tab.

2. Select **Display** or a **Worksheet** type from the list.

3. Click **OK**.

   The **New** dialog is closed and your selection is opened in the **worksheet editor**.

---

**Note:** When you add an I/O driver to the project, an associated option allows you to open a new **driver worksheet**. You also can create new screens or worksheets by right-clicking on the folder in the **Project Explorer** and selecting the **Insert** option from the shortcut menu.

**Note:** Worksheets for **DDE Client** and **ODBC** are not available for projects running on Windows Embedded target systems, because those technologies are not supported by the Windows Embedded operating system.

---

**Creating a New Project**

To create a new project:
1. Click the **Project** tab.

![New Project tab](image)

2. In the **Project name** box, type a name for your project.

3. By default, IWS stores all projects in the location specified by the Default Project Path preference *(Preferences on the Project tab of the ribbon)*, so that path will be automatically displayed in the **Location** box. To save your project in another location, click **Browse** and then select a folder.

4. Select a **Target platform**.

5. Click **OK** to continue to the **Project Wizard** dialog.

![Project Wizard](image)

6. In the **Resolution** box, select a screen resolution. If you select Custom, then also type the width and height in pixels.

7. To share tags with another PC-based control application, select the application type from the list and click the **Configure** button. (Each type has its own configuration options; please consult the application vendor.) Otherwise, leave it set to `<None>`.

8. Click **OK** when you’re done.

For a more detailed walkthrough, see **Creating a new project**.
**Open Project**

The **Open Project** command on the Application menu is used to open a saved project. Selecting the command opens a standard Windows *Open* dialog, which you can use to locate and open the project file (*.app)*.

![Open dialog](image)

**Open**

The **Open** command on the Application menu is used to open a saved worksheet file. Selecting the command opens a standard Windows *Open* dialog, which you can use to locate and open the worksheet file. The application can open many different file types, so use the **File type** combo-box to filter the files.

![Available worksheet file types in the Open dialog](image)

**Save**

The **Save** command on the Application menu is used to save the active screen or worksheet. The command becomes available only after you modify the worksheet in some way.
Save As
The Save As command on the Application menu is used to open a save the active screen or worksheet at another location.

Save All
The Save All command on the Application menu is used to save all open worksheet files. The command becomes available only after you modify the a worksheet in some way.

Save All as HTML
The Save All as HTML command on the Application menu is used to save all of your project’s screens and screen groups in HTML format.
After saving, the files can be found in the Web folder in the Project Explorer. For more information, see Deploying your project as a Web application.

**Note:** You must close all worksheets before you execute this command.

Save as HTML
The Save as HTML command on the Application menu is used to save the active screen in HTML format.
After saving, the file can be found in the Web folder in the Project Explorer. For more information, see Deploying your project as a Web application.

Save Screen Group as HTML
The Save Screen Group as HTML command on the Application menu is used to save a selected screen group in HTML format.
After saving, the files can be found in the Web folder in the Project Explorer. For more information, see Deploying your project as a Web application.

Print
The Print command on the Application menu is used to print the active screen or worksheet.
Selecting the command opens a standard Windows Print dialog, which you can use to adjust the print range and the number of copies.

Print Preview
The Print Preview command on the Application menu is used to see what the active worksheet would look like when it is printed.
Use the buttons located along the top of the Print Preview window as follows:

- Click Print to open the Print dialog and print the screen or worksheet.
- Click Next Page to view the next page in a series of pages.
- Click Prev Page to view the previous in a series of pages.
- Click Two Page to view two pages at a time.

**Note:** The Next Page, Prev Page, and Two Page buttons become active only when you are printing more than one page.

- Click Zoom In to check details.
- Click Zoom Out to change back to the default size.
Note: The Zoom Out button becomes active after you Zoom In.

- Click Close to close the Print Preview window.

**Print Setup**

The Print Setup command on the Application menu is used to configure the options (e.g., paper size, print orientation) for a selected printer.

Selecting the command opens a standard Windows Print Setup dialog:

![Print Setup dialog](image)

**Note:** To specify a default printer:

1. Go to your Windows Start menu and select Start > Settings > Printers.
2. When the Printers dialog displays, right-click on a printer name.
3. When the shortcut menu displays, select Set As Default.
4. A check displays next to Set As Default indicating the selected printer is the default.

**Close**

The Close command on the Application menu is used to close the active screen or worksheet. When you select this command, you will be prompted to save your changes before closing.

**Close All**

The Close All command on the Application menu is used to close all open screens and worksheets. When you select this command, you will be prompted to save your changes before closing.

**Exit**

The Exit command on the Application menu is used to close all open screens and worksheets, save the project database, and then exit the application. When you select this command, you will be prompted to save your changes before closing.

**Note:** Selecting this command is the same as clicking the Close button on the title bar.
Quick Access Toolbar

The Quick Access Toolbar is a customizable toolbar that contains a set of commands that are independent of the ribbon tab that is currently displayed.

Move the Quick Access Toolbar

The Quick Access Toolbar can be located in one of two places:

- Upper-left corner next to the Application button (default location); or
- Below the ribbon, where it can run the full length of the application window.

If you don’t want the Quick Access Toolbar to be displayed in its current location, you can move it to the other location:

1. Click Customize Quick Access Toolbar.
2. In the list, click Show Below Ribbon or Show Above Ribbon.

Add a command to the Quick Access Toolbar

You can add a command to the Quick Access Toolbar directly from commands that are displayed on the ribbon:

1. On the ribbon, click the appropriate tab or group to display the command that you want to add to the Quick Access Toolbar.
2. Right-click the command, and then click Add to Quick Access Toolbar on the shortcut menu.

You can also add and remove commands — as well as reset the toolbar to its default — using the Customize dialog:

1. Click Customize Quick Access Toolbar.
2. In the list, click More Commands. The Customize dialog is displayed.

3. In the Choose commands from menu, select the appropriate Ribbon tab. The commands from that tab are displayed in the Commands list.
4. In the Commands list, select the command that you want to add to the Quick Access Toolbar.
5. Click Add.
Only commands can be added to the Quick Access Toolbar. The contents of most lists, such as indent and spacing values and individual styles, which also appear on the ribbon, cannot be added to the Quick Access Toolbar.
The new ribbon combines the numerous menus and toolbars from the previous version of IWS into a single, user-friendly interface. Almost all application commands are now on the ribbon, organized into tabs and groups according to general usage.

**Ribbon**

The Ribbon interface

**Home tab**

The Home tab of the ribbon is used to manage your project within the development environment.

The tools are organized into the following groups:

- **Clipboard:** Cut, copy, paste, and find items in project screens and task worksheets.

- **Local Management:** Run and stop the project on the local station (i.e., where the development application is installed), as well as manage the execution tasks. You can also run a project in Debug mode, for debugging VBScript.

- **Remote Management:** Connect to a remote station (e.g., a Windows Embedded device) so that you can download the project to it, and then run, stop, and troubleshoot the project on that station. For more information, see About remote management on page 517.

- **Tools:** Miscellaneous tools to verify the project, import tags from other projects, convert screen resolutions, and register ActiveX and .NET controls.

- **Tags:** Manipulate tags and tag properties in the project database.

**View tab**

The View tab of the ribbon is used to customize the look of the development environment itself.

The tools are organized into the following groups:

- **Show/Hide:** Show and hide the different parts of the development environment, as well as restore the default layout.
• **Zoom**: Zoom in and out of the screen editor.
• **Options**: Change the language and font used in the development environment.
• **Window**: Arrange the windows in the development environment.

**Insert tab**
The **Insert** tab of the ribbon is used to insert new tags, screens, worksheets, and other components into your project.

The tools are organized into the following groups:

- **Global**: Insert tags, classes, translations, and procedures into the Global tab of the Project Explorer.
- **Graphics**: Insert screens and screen groups into the Graphics tab of the Project Explorer.
- **Task Worksheets**: Insert task worksheets into the Tasks tab of the Project Explorer.
- **Communication**: Insert server configurations and communication worksheets into the Comm tab of the Project Explorer.

**Project tab**
The **Project** tab of the ribbon is used to configure your project settings.

The tools are organized into the following groups:

- **Settings**: Configure the general project settings, set the project to run as a Windows service, or enable workgroup collaboration and version control.
- **Security System**: Enable and configure the project security system.
- **Web**: Configure the project to accept connections from a variety of thin clients.

**Graphics tab**
The **Graphics** tab of the ribbon is used to draw project screens.
The Development Environment

Graphics tab of the ribbon

Note: This tab is available only when you have a project screen open for editing.

The tools are organized into the following groups:

- **Screen**: Configure settings for the project screen itself, such as its attributes, script, and background color or image.
- **Editing**: Select and edit objects in the project screen.
- **Shapes**: Draw static lines and shapes.
- **Active Objects**: Draw active objects, like buttons and check boxes.
- **Data Objects**: Draw objects that display historical data, like alarms, events, and trends.
- **Libraries**: Select from libraries of premade objects, such as symbols, .NET and ActiveX controls, and external image files.
- **Animations**: Apply animations to other screen objects.

Format tab

The Format tab of the ribbon is used to format and arrange objects in a project screen.

Note: This tab is available only when you’ve selected one or more objects in a project screen.

The tools are organized into the following groups:

- **Arrange**: Arrange objects in a project screen, including bring to front and send to back, group, align, and rotate.
- **Position**: Precisely adjust the position of a screen object in a project screen.
- **Size**: Precisely adjust the size of a screen object.
- **Style**: Change the fill and line color of a screen object.
- **Fonts**: Change the caption font of a screen object.
Help tab

The Help tab of the ribbon provides additional help with using the software.

Help tab of the ribbon

The tools are organized into the following groups:

- **Documentation**: Access the documentation for the development application, including this help file / technical reference and notes for the individual communication drivers.

- **Information**: Access other information about InduSoft Web Studio, including the license agreement, product website, and release notes, as well as system and support details that make it easier for Customer Support to assist you.
Project Explorer

The Project Explorer organizes all of the screens, worksheets, and other items that comprise your project and presents them in an expandable tree-view.

To open a folder and view its contents, either click the Expand icon to the left of the folder or double-click the folder itself.

To close a folder, click the Collapse icon to the left of the folder.

If you right-click any item in the Project Explorer, then a shortcut menu will appear with contextual commands for that item.

There are four main sections, or tabs, in the Project Explorer: Global, Graphics, Tasks, and Comm.

Global tab

The Global tab of the Project Explorer contains the project tags database, as well as other features that apply to the entire project such as the security system, VBScript procedures, and UI translation.

Global tab of the Project Explorer

The folders on the Global tab are described in the following sections:

Project Tags

The project tags database contains all of the data tags that you create during project development, such as screen tags (e.g., button1_state) or tags that read from / write to connected devices.

Classes

Classes are compound tags that you can create to associate a set of values, rather than a single value, with an object. For example, where you may normally create separate tags for a tank’s pressure, its temperature, and its fill level, you can instead create a “tank” class that includes all three.

Shared Database

The shared database contains tags that were created in another program and then imported into or integrated with your project.

System Tags

System tags are predefined values such as the date, the time, the name of the current user, and so on. You can use these values to develop supervisory functions and housekeeping routines.

All system tags are read-only, which means you cannot add, edit, or remove these tags from the database.
Security
If you choose to enable it, you can use the project security system to control who may log on to your project and what they may do during runtime.

Procedures
Procedures are VBScript functions and sub-routines that can be called by any other script in your project.

Event Logger
The event logger saves important runtime messages and task results to an external database.

Translation
You can use the translation table to develop a multilingual user interface (MUI) for your project.

Graphics tab
The Graphics tab of the Project Explorer contains all of the screens, screen groups, and symbols in your project.

The folders on the Graphics tab are described in the following sections:

Screens
You create screens to provide a graphical interface for your project. Each screen can contain many buttons, sliders, dials, indicators, graphs, and so on.

Screen Groups
You can combine individual screens into screen groups, so that they all open together at the same time.

Thin Clients
You can deploy your project as a web application to be accessed by thin clients such as desktop web browsers, tablets, and smartphones. You can even deploy different versions of your project with different levels of functionality for each type of client.

Project Symbols
This folder contains all of the custom symbols that you create for your project. A symbol is a group of interconnected screen objects that work together to perform a single function — for example, lines, rectangles, and text fragments that have been arranged to make a slider control.

Graphics Script
You can use this worksheet to define VBScript sub-routines that are called only when the graphics module starts (i.e., when a client station connects to the server and displays the graphical interface), while it is running, and when it ends.

Symbols
The symbols library contains not only the custom symbols that you create (see Project Symbols above), but also a large selection of premade symbols that are installed with the development application.

**Layout**

The layout editor displays all of the screens that are currently open for editing. You can use it to visualize how the screens are arranged together and reuse screens in multiple layouts — for example, to create a common navigation bar across your entire project.

**Tasks tab**

The Tasks tab of the Project Explorer organizes the worksheets that are processed as background tasks (i.e., server-based maintenance tasks that are not directly related to screen operations or device I/O) during project runtime.

The folders on the Tasks tab are described in the following sections:

**Alarms**

You can use Alarm worksheets to define when alarms are triggered, how they must be handled, and what messages they generate.

(You can then use the Alarm/Event Control screen object to display your alarms on screen, but that is a separate procedure.)

**Trends**

You can use Trend worksheets to select project tags that should be displayed as data trends and/or saved as historical data.

(You can then use the Trend Control screen object to actually display your trends on screen, but that is a separate procedure.)

**Recipes**

You can use Recipe worksheets to select project tags that will load values from and/or save values to an external file. These worksheets are typically used to execute process recipes, but you can store any type of information such as passwords, operation logs, and so on.

(You can then call the Recipe function to actually run a configured Recipe worksheet, but that is a separate procedure.)

**Reports**

You can use Report worksheets to design runtime reports that are either sent to a printer or saved to disk.

(You can then call the Report function to actually run a configured Report worksheet, but that is a separate procedure.)
ODBC
You can use ODBC worksheets to set up connections and exchange data with other ODBC-compliant databases.

Math
You can use Math worksheets to develop complex runtime logic using the built-in scripting language.

Script
You can use Script worksheets to develop complex runtime logic using VBScript.

Scheduler
You can use Scheduler worksheets to run commands at specified times, dates, or trigger events.

Database
You can use Database worksheets to set up connections and exchange data with external databases using the standard ADO.NET interface (as an alternative to ODBC).

Comm tab
The Comm tab of the Project Explorer organizes the worksheets that control communication with remote devices, using either direct communication drivers or other common protocols.

The folders on the Comm tab are described in the following sections:

Drivers
You can use Driver worksheets to communicate with PLCs and other hardware, using any of the hundreds of direct communication drivers that are installed with the development application.

OPC DA 2.05
You can use OPC worksheets to communicate with OPC servers via the OPC Classic protocol.

OPC UA
You can use OPC UA worksheets to communicate with OPC servers via the new OPC Unified Architecture protocol.

OPC .Net
You can use OPC .Net worksheets to communicate with OPC servers via the new OPC .NET 3.0 protocol (formerly OPC Xi).

OPC XML/DA
You can use OPC XML/DA worksheets to communicate with OPC servers via the new OPC XML-DA protocol.

TCP/IP
You can use TCP/IP worksheets to configure communication between your own project and other IWS projects. The TCP/IP Client and TCP/IP Server modules enable two or more projects to keep their databases synchronized using the TCP/IP protocol.

**DDE**

You can use DDE worksheets to communicate with other Microsoft Windows applications, such as Microsoft Excel, that support the Dynamic Data Exchange protocol.
Screen/Worksheet Editor

Use the powerful, object-oriented screen editor to create and edit a variety of screens and worksheets for your projects. You can input information using your mouse and keyboard, output control data to your processes, and automatically update screens based on data input from your processes.

Other screen editor features include:

- Simple point-and-click, drag-and-drop interface
- Grouping objects to preserve the construction steps of individual objects
- Editing objects without having to ungroup internal object components or groups
- Handling bitmap objects and background bitmaps
- Status line support in project windows and dialogs
Database Spy

The Database Spy window is a debugging tool that allows you to: monitor and force values to project tags; execute and test functions; and execute and test math expressions.

![Sample Database Spy window]

The window contains the following elements:

- For each item that you want to monitor during runtime:
  - **Tag/Expression**: Specify a project tag, system tag, or expression that you want to monitor.
  - **Value**: Displays the value returned by the tag/expression.
  - **Quality**: Displays the quality (GOOD or BAD) of the value returned by the tag/expression.
  - **Continuous**: Select this option to have the project continuously evaluate the tag/expression.

- **DB tabs**: The windows is divided into multiple sheets, so that you can keep your items organized.

- **Scroll bars**: Use to view areas of the Database Spy that are obscured from view because of the window size or the size of the current sheet.

**Tip**: The Database Spy is dockable, which means you can move it to another location in the development environment. Click on the titlebar and drag it to a new location. Release the mouse button to attach or dock the window to its new location.
Output (LogWin)

You can use the *Output* window to view debugging messages generated during project runtime. The window displays OPC, DDE, and TCP/IP transactions, module activation, trace tags, and so on.

![Sample Output Window](image)

The window contains the following elements:

- **XRef** tab: Use the *Cross Reference* tool to get a tag, and to find every place in the project where the tag is being used. Results appear on this tab, providing path and filename, column, row in the spreadsheet. So, if something changes in the tag, and produces unexpected or unsuccessful results, you can locate all instances of the tag for debugging purposes.

  **Note:** The XRef tab does not work for functions, only tags, but it does allow you to look for array indices.

- **Hide Docked Window** button (x): Click to open or close the window.

  Alternatively, to hide the window, you can deselect (uncheck) the *Output Window* option on the View tab of the ribbon.

- **Contract/Expand** button (x): Click to contract and expand the *Output* window.

- **Scroll Bars**: Click and drag to view areas of the *Output* window that are obscured from view because of the window size or the length of your data.

The *Output* window is dockable, which means you can move it to another location in the development environment. Click on the title bar and drag the window to a new location. Release the mouse button to attach or dock the window to its new location.

**Configuring the Log to Show Additional Information**

By default, the log shows only debugging and error messages — that is, messages indicating that your project is not running properly. If the log showed all messages generated by IWS, it would quickly overflow with information, making it unusable.

To configure the log to show specific additional information:

1. Right-click anywhere in the *Output* window, and then click *Settings* on the shortcut menu. The *Log Settings* dialog is displayed.

2. In the *Log Options* tab of the dialog, select the specific types of messages that you want the log to show.
### The Development Environment

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Field Read Commands and Field Write Commands</strong></td>
<td>Show any read and/or write commands that are sent to connected devices.</td>
</tr>
<tr>
<td>Protocol Analyzer</td>
<td>Show messages generated by configured device drivers.</td>
</tr>
<tr>
<td>OPC DA 2.05</td>
<td>Show messages generated by the OPC DA 2.05 Client Runtime task.</td>
</tr>
<tr>
<td>OPC XML/DA</td>
<td>Show messages generated by the OPC XML/DA Client Runtime task.</td>
</tr>
<tr>
<td>OPC UA</td>
<td>Show messages generated by the OPC UA Client Runtime task.</td>
</tr>
<tr>
<td>OPC .NET</td>
<td>Show messages generated by the OPC .Net Client Runtime task.</td>
</tr>
<tr>
<td>Logon/Logout</td>
<td>Display a message whenever a user logs in or logs out. (For more information, see Security.)</td>
</tr>
<tr>
<td>Trace Messages</td>
<td>Show messages generated by the <code>Trace()</code> function. This function is used to generate customized messages from within your project.</td>
</tr>
<tr>
<td>Database Messages</td>
<td>Show messages generated by the ODBC and ADO.NET database interfaces.</td>
</tr>
<tr>
<td>Recipe/Report</td>
<td>Show messages generated by the Recipe and Report tasks.</td>
</tr>
<tr>
<td><strong>Display Open/Close</strong></td>
<td>Display detailed information whenever a screen is opened or closed:</td>
</tr>
<tr>
<td></td>
<td>• Disk Load Time: Time to load the screen file from the disk into memory.</td>
</tr>
<tr>
<td></td>
<td>• Open Time: Time to open the screen, including initializing tags used in</td>
</tr>
<tr>
<td></td>
<td>the screen and running any &quot;OnOpen&quot; scripts or functions.</td>
</tr>
<tr>
<td></td>
<td>• Total Load Time: Total time to load the screen (includes Disk Load Time</td>
</tr>
<tr>
<td></td>
<td>and Open Time above).</td>
</tr>
<tr>
<td></td>
<td>• First Draw Time: Time to first drawing of screen objects.</td>
</tr>
<tr>
<td></td>
<td>• First OnWhile Time: Time to first running of any &quot;OnWhile&quot; scripts or</td>
</tr>
<tr>
<td></td>
<td>functions.</td>
</tr>
<tr>
<td></td>
<td>• Total Open Time: Total time to open the screen (includes First Draw Time</td>
</tr>
<tr>
<td></td>
<td>and First OnWhile Time above).</td>
</tr>
<tr>
<td></td>
<td>• Close Time: Time to close the screen, including finalizing tags used in</td>
</tr>
<tr>
<td></td>
<td>the screen and running any &quot;OnClose&quot; scripts or functions.</td>
</tr>
<tr>
<td></td>
<td>• Total Close Time: Total time to close the screen, including the time to</td>
</tr>
<tr>
<td></td>
<td>close the screen file on the disk.</td>
</tr>
<tr>
<td></td>
<td>This information can be used to analyze runtime performance on low-end</td>
</tr>
<tr>
<td></td>
<td>target systems. If a particular step of opening or closing takes an unusually long time, then it can be identified and redesigned.</td>
</tr>
<tr>
<td>TCP/IP Messages</td>
<td>Show messages generated by TCP/IP communications.</td>
</tr>
<tr>
<td>DDE Messages</td>
<td>Show messages generated by DDE communications.</td>
</tr>
<tr>
<td>Insert date/time</td>
<td>Timestamp each message.</td>
</tr>
</tbody>
</table>

3. If you selected **OPC XML/DA, OPC UA, OPC .NET**, you must specify exactly which messages are displayed in the **Output** window. To do this:

a. Click the browse button to the right of the option. An additional `Log Settings` dialog is displayed.

![OPC UA Log Settings dialog](image)

Each task has its own settings, but the descriptions of the settings are the same for all three tasks.
b. Select the type(s) of messages to display.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Enable trace messages on changes in connection status between the OPC Client (i.e., your project) and the OPC Server. Examples of messages:</td>
</tr>
<tr>
<td></td>
<td>OPC UA: Connection established with server “Connection1”</td>
</tr>
<tr>
<td></td>
<td>OPC UA: Error to connect to server “Connection2” – Time out waiting for server response</td>
</tr>
<tr>
<td>Subscription</td>
<td>Enable trace messages on subscriptions to server items, such as the creation of new subscriptions and changes in data type. Examples of messages:</td>
</tr>
<tr>
<td></td>
<td>OPC UA: Group 1 =&gt; Subscription created</td>
</tr>
<tr>
<td>Read</td>
<td>Enable trace messages on read operations. Examples of messages:</td>
</tr>
<tr>
<td></td>
<td>OPC UA: Read Group 1 Started – OK</td>
</tr>
<tr>
<td></td>
<td>OPC UA: Read Group 1 Completed – OK</td>
</tr>
<tr>
<td></td>
<td>OPC UA: Read Group 25 Started – Error, asynchronous reading pending for the current group</td>
</tr>
<tr>
<td>Write</td>
<td>Enable trace messages on write operations. Examples of messages:</td>
</tr>
<tr>
<td></td>
<td>OPC UA: Write Group 1 Started [Line 1 = 10.25, Line 42 = 20.45] – Status OK</td>
</tr>
<tr>
<td></td>
<td>OPC UA: Write Group 1 Completed [All Items] – Status OK</td>
</tr>
</tbody>
</table>

c. In the Verbosity list, select the level of verbosity.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Critical issues that have caused the communication to fail. These issues must be resolved before you can resume communication.</td>
</tr>
<tr>
<td>Warning</td>
<td>Non-critical issues that may cause the communication to fail under other conditions. These issues should be resolved as soon as possible. Includes Error above.</td>
</tr>
<tr>
<td>Information</td>
<td>All trace messages on the communication. This is the default. Please note that this can be extremely verbose, depending on which type(s) of messages you have selected to display and how many item subscriptions you have created. Includes Error and Warning above.</td>
</tr>
</tbody>
</table>
d. Click **OK** to close the dialog.

4. In the *Log Tags* tab of the dialog, click **Add** to browse for *project tags*.

![Log Settings — Tags Tab](image)

The *Output* window will display a message whenever the value of a selected tag changes.

5. Click **OK** to save your settings and close the *Log Settings* dialog.
Title Bar

The Title Bar located along the top of the development environment displays the application name (e.g., InduSoft Web Studio) followed by the name of the active screen or worksheet (if any).

![Example of Title Bar]

The Title Bar also provides the following buttons (from left to right):

- **Minimize button**: Click to minimize the development environment window to the Taskbar.
- **Restore Down / Maximize**: Click to toggle the development environment window between two sizes:
  - **Restore Down button**: reduces the window to its original (default) size.
  - **Maximize button**: enlarges the window to fill your computer screen.
- **Close button**: Click to save the database and then close the development environment. If you modified any screens or worksheets, the application prompts you to save your work. This button’s function is similar to clicking **Exit Application** on the Application menu.

**Note:** Closing the development environment does not close either the project viewer or the runtime system, if they are running.
Status Bar

The Status Bar located along the bottom of the development environment provides information about the active screen (if any) and the state of the application.

```
```

**Example of Status Bar**

The Status Bar fields (from left to right) are described in the following table:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Mode</td>
<td>The current execution mode of the application.</td>
</tr>
<tr>
<td>CAP</td>
<td>Indicates whether the keyboard <strong>Caps Lock</strong> is on (black) or off (grey).</td>
</tr>
<tr>
<td>NUM</td>
<td>Indicates whether the keyboard <strong>Num Lock</strong> is on (black) or off (grey).</td>
</tr>
<tr>
<td>SCRL</td>
<td>Indicates whether the keyboard <strong>Scroll Lock</strong> is on (black) or off (grey).</td>
</tr>
<tr>
<td>Object ID</td>
<td>The ID number of a selected screen object.</td>
</tr>
<tr>
<td>Cursor Position</td>
<td>The location of the cursor on the active screen or worksheet. If it's a screen, then the position of the mouse cursor is given as X,Y coordinates, where X is the number of pixels from the left edge of the screen and Y is the number of pixels from the top edge of the screen. If it's a worksheet, then the position of the text cursor is given as Line and Column.</td>
</tr>
<tr>
<td>Object Size</td>
<td>The size (in pixels) of a selected screen object, where W is the width and H is the height.</td>
</tr>
<tr>
<td>No DRAG</td>
<td>Indicates whether dragging is disabled (No DRAG) or enabled (empty) in the active screen.</td>
</tr>
<tr>
<td>Tag Count</td>
<td>The total number of tags used so far in the project.</td>
</tr>
</tbody>
</table>
Creating a new project

This task describes how to create a new IWS project, including how to select the product type and default screen resolution.

1. Click the Application button (in the top-left corner of the development application window), and then click **New** on the Application menu.
   The **New** dialog is displayed.

2. Click the **Project** tab.

3. In the **Project name** box, type the name of your project.
   Keep the following guidelines in mind:
   • You must follow the usual Windows naming conventions, particularly regarding the use of special characters; and
   • Do not use spaces in the name if you want to access your project from a Thin Client, because URLs cannot include spaces.

4. Click **Browse** to the right of the **Location** box, and then navigate to the folder where you want to save your project.
   **Tip:** By default, projects are saved in your Documents folder at C:\Users\username\Documents \InduSoft Web Studio v7.1 Projects\. To change this default location for future projects, edit the **Default project path** setting in the project settings. For more information, see **Preferences tab**.

5. If you have collaboration enabled and you want to add this project to your source control, select **Add new project to collaboration**.
   For more information, see **About collaboration and source control** on page 452.

6. In the **Target platform** list, select the platform of the computer or embedded device that will host your project runtime.
   Selecting a platform will filter the list of available product types to show only those types that work on the selected platform. It is not absolutely necessary to select a target platform — if you leave the selection as (All), then all product types are shown — but it helps you to make the decision.

7. In the **Product type** list, select the product type for your project.
   The product type determines how many tags your project will support, among other things. For more information, see **About target platforms and product types** on page 91.
8. Click **OK**.
The *New* dialog is closed, and the *Project Wizard* dialog is displayed.

Tip: You may change the product type later, after you have created the project, by using the **Target System** command on the *Project* tab of the ribbon.

9. In the *Resolution* list, select the default resolution for your project screens. If you select **Custom**, then also enter the width and height.
In most cases, the default resolution should match the display size of the workstation or device that will host your project runtime, or if you plan to access the project from other clients, then the default resolution should match the display size of those clients.

Tip: You may change the resolution of individual project screens later, after you have created the project, by editing the screen attributes. Also, you may change the resolution of all project screens — effectively selecting a new default resolution — by using the **Convert Resolution** command on the *Home* tab of the ribbon.

10. Click **OK**.
The *Project Wizard* dialog is closed, and the project is created.

**About target platforms and product types**

A project's target platform and product type determine important things about the project, such as how many tags the project will support and which features can be used during run time.

**Target platform**

The target platform is generally either Windows or Windows Embedded.

**Windows**

A computer that runs one of the following operating systems:
- Microsoft Windows XP Service Pack 3
- Microsoft Windows Vista Service Pack 2
- Microsoft Windows 7 Service Pack 1
- Microsoft Windows 8 or 8.1 (excluding Windows RT)
- Microsoft Windows Server 2003 Service Pack 2
- Microsoft Windows Server 2008 Service Pack 2
- Microsoft Windows Server 2008 R2 Service Pack 1
- Microsoft Windows Server 2012
- Microsoft Windows XP Embedded Service Pack 3
• Microsoft Windows Embedded Standard 7

The computer must have the full InduSoft Web Studio software installed, even if it is not used for project development, because the full software includes the modules that are needed to host the project runtime on Windows. The computer must also have an appropriate runtime license key. For more information, see Install the full InduSoft Web Studio software on page 37.

All project features are available when the target platform is Windows.

**Windows Embedded**

An embedded device that runs one of the following operating systems:

• Microsoft Windows XP Embedded Service Pack 3
• Microsoft Windows Embedded Standard 7
• Microsoft Windows Mobile or Windows CE 5.x/6.x
• Microsoft Windows Embedded Compact 7

The device must have either EmbeddedView (for Windows XP Embedded and Windows Embedded Standard) or CEView (for Windows Mobile, Windows CE, and Windows Embedded Compact) installed. The device must also have an appropriate runtime license key. For more information, see Install EmbeddedView or CEView on a target device on page 42.

Please note that when the target platform is Windows Embedded, certain project features are not available:

• All DDE features, including the DDE Server task and the DDE Client task and worksheets.
• ODBC Runtime task, worksheets, and functions. (ADO database interface is still available.)
• OPC HDA Server task.
• OPC .Net Client task and worksheets.
• OPC XML/DA Client task and worksheets.
• Minor graphic features such as support for image formats other than BMP or JPG and fill effects for shapes other than rectangle.
• Miscellaneous other built-in functions, as described in the documentation for the Built-in Scripting Language.

In most cases, if you want to host your project runtime on a Windows XP Embedded or Windows Embedded Standard device and you do not plan to do project development on that device, then you should only install EmbeddedView on the device and select Windows Embedded as the target platform. EmbeddedView has a smaller footprint than the full InduSoft Web Studio software, and it can be installed and managed remotely.

However, if you want to use any of the features that are not available when the target platform is Windows Embedded, then you must install the full InduSoft Web Studio software on the device and select Windows as the target platform.

**Note:** The full InduSoft Web Studio software cannot be installed on a Windows Embedded Compact device under any circumstances.

**Product type**

The product type primarily determines how many tags you can use in the project, including tags shared or imported from other systems. Given the available system resources, Windows can typically support far more tags than Windows Embedded.

Please note that the computer or embedded device that will host your project runtime must have a license key that matches or exceeds the selected product type. To verify the license key, run Protection Manager (Start > All Programs > InduSoft Web Studio v7.1 > Register) on that computer or embedded device. Although you can change both the product type and license key later, we recommend that you verify the license key and select the correct product type now so that you do not waste time developing a project that uses more tags than you are licensed for.

Also, if you plan to run your project on multiple stations with different license keys, then we recommend that you develop for the lowest common product type.
Target system

The target platform and product type together determine the target system, which you can change later by using the Target System command (on the Project tab of the ribbon).

Changing the product type of an existing project

This task describes how to use the Target System command to change the product type of an existing project.

1. On the Project tab of the ribbon, in the Settings group, click Target System.
   A list of available product types is displayed.

2. In the list, select the new product type of the project.
   The product type determines how many tags your project will support, among other things. For more information, see About target platforms and product types on page 91.

The project is converted to the selected product type.

Note: If you changed from a Windows target platform to a Windows Embedded target platform, then some project features will no longer be available.
Configuring additional project settings

Select **Project Settings** to open the **Project Settings** dialog, which controls settings that affect the overall project.

**Information tab**

Use the following parameters to identify the project (for documentation purposes only).

- Description
- Revision
- Company
- Author
• Field Equipment
• Notes

**Options tab**

Use this tab to specify parameters relating to your project in general.

![Project Settings: Options tab](image)

A description of these parameters follows:

- **Target system**: Use the combo box to specify the target system for the current project. The target system sets the project restrictions (such as number of tags supported) and must match your license. The description of the main license restrictions for each target system is displayed below the combo-box where you chose it.

  **Note**: If you specify a **Target System** level that does not match the actual license level on the target system, then your project might not run properly.

- **Resolution**: Displays your project’s screen resolution.

- **Alarm History and Events**: Type a value into the History Life Time (days) field to specify how long to keep alarm and event history files. After the specified number of days, the project automatically deletes existing alarm/event history files that are older than the period specified. If you type zero in this field, the project does not delete any history files automatically. In such a case, you should create an external procedure to clean the old history files; otherwise, the free memory in the computer will eventually be depleted.

- **History Format**: Select the format of the Alarm/History event, as follows:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proprietary</td>
<td>Saves the history data in the Alarm sub-folder of your project folder (by default) in text files using the proprietary format.</td>
</tr>
<tr>
<td>Database</td>
<td>Saves the history data in the SQL Relational Database specified by the user, using the built-in ADO interface.</td>
</tr>
<tr>
<td>Binary</td>
<td>Saves the history data in the Alarm sub-folder of your project folder (by default) in binary files using the proprietary format.</td>
</tr>
</tbody>
</table>

For more information, see [Saving your alarm history / event log to an external database](#).
• **Custom Fields**: Specify up to 10 custom fields for alarms and events, respectively, that will be saved in the history. For alarms, the custom fields are automatically added to the Alarm worksheet body. For events, the custom fields are available as parameters of the function `SendEvent`.

If your project is running when you change these settings, you must stop the project and then run it again for the changes to take effect.

- **Note**: This feature is not supported in projects running on Windows Embedded target systems.

• **Default Database**: Allows you to configure a Default Database, which can be shared by different tasks and objects. See Configuring a Default Database for All Task History for more information.

• **Performance Control**: Allows you to configure how memory is allocated for screen graphics during runtime. For more information, see Configure the performance control settings on page 101.

### SAVING YOUR ALARM HISTORY / EVENT LOG TO AN EXTERNAL DATABASE

By default, your project’s alarm history and event log are saved to proprietary-format text files in your project’s Alarms folder. However, you can change your project settings to save them to an external SQL database instead.

- **Note**: If your project was created with InduSoft Web Studio v7.1+SP1 or earlier and then upgraded to the current version, you should consider starting over with new database tables.

  Beginning with InduSoft Web Studio v7.1+SP2, new database tables are automatically indexed by event time in order to improve runtime performance. Existing database tables cannot be indexed in this way, so if you can afford to discard that data, you should change your database configuration to create new tables.

  If you do this, you must also manually edit your project file (`projectname.APP`) to add the following setting:

  ```
  [Alarm]
  AddEventTimeColumn=1
  ```

  The setting exists for backward compatibility, and it defaults to 0 for projects that were upgraded from earlier versions.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**.
2. In the **Alarm History and Events** area, in the **History Life Time** box, type the number of days of history that you want to save.

   As the history exceeds the specified number of days, it will be automatically deleted in a first-in, first-out manner. If no number is specified — that is, if it is left blank or set to 0 — then history will never be deleted. There is no limit to how much history you can save, but the more you save, the more disk space it will take.

3. From the **History Format** list, select **Database**.

4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), in the **Default Database** area, click **Configure**. The **Default Database Configuration** dialog is displayed. Use the dialog to configure the database connection. For more information, see [Configuring a default database for all task history](#).

5. To configure a separate database for either your event log or your alarm history, click **Event Database** or **Alarm Database**, respectively.

   In either case, a **Database Configuration** dialog is displayed. Use the dialog to configure the database connection. For more information, see [Database Configuration](#).

6. Click **OK**.

### CONFIGURING A DEFAULT DATABASE FOR ALL TASK HISTORY

You can configure a Default Database that will save the historical data from all Tasks in a project. After you do, when you create a new Task worksheet, you can choose either to use the Default Database or to configure a new database for that specific worksheet.

To configure the connection settings for the Default Database:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**. The **Project Settings** dialog is displayed.
2. Click **Configure**. The *Default Database Configuration* dialog is displayed.

![Default Database Configuration dialog](image1)

Please refer to *Database Configuration dialog* for help completing the fields in this window.

**DATABASE CONFIGURATION**

The *Database Configuration* dialog allows you to configure the necessary settings to link IWS to an external database file.

![Database Configuration dialog](image2)

- **Database** combo-box: Allows you to select either Primary or Secondary. With Primary, all settings displayed in the Database Configuration window apply to the Primary Database interface. Otherwise, they apply to the Secondary Database interface. You can configure the Secondary database in the following modes:
  - **Disabled**: In this mode, IWS saves data in the Primary Database only. If the Primary Database is unavailable for any reason, the data is not saved anywhere else. This option may cause loss of data if the Primary Database is not available.
  - **Redundant**: In this mode, IWS saves data in both Primary and Secondary Databases. If one of these databases is unavailable, IWS keeps saving data only in the database that is available. When the database that was unavailable becomes available again, IWS synchronizes both databases automatically.
  - **Store and Forward**: In this mode, IWS saves data in the Primary Database only. If the Primary Database becomes unavailable, IWS saves the data in the Secondary Database. When the Primary Database becomes available again, IWS moves the data from the Secondary Database into the Primary Database.

**Note**: The Primary and Secondary can be different types of databases. However, they must have the same fields.
Using the Secondary Database, you can increase the reliability of the system and use the Secondary Database as a backup when the Primary Database is not available. This architecture is particularly useful when the Primary Database is located in the remote station. In this case, you can configure a Secondary Database in the local station to save data temporarily if the Primary Database is not available (during a network failure, for instance).

- **Use project default** checkbox: When this option is checked, IWS uses the settings configured in the Default Database for the task that is being configured (Connection string, User name, Password, Retry Interval and Advanced Settings). When this option is not checked, you can configure these settings individually to the current task.

- **Connection string** field: This field defines the database where IWS will write and read values as well as the main parameters used when connecting to the database. Instead of writing the Connection string manually, you can press the browse button (…) and select the database type from the Data Link Properties window.

  ![Data Link Properties dialog](Image)

  **Data Link Properties dialog**

  **Note:** The list of Database Providers shown in the Data Link Properties window depends on the providers actually installed and available in the computer where you are running IWS. Consult the operating system documentation (or the database documentation) for further information regarding the settings of the Provider for the database that you are using.

- **User name** field: User name used to connect to the database. The user name configured in this field must match the user name configured in the database.

- **Password** field: Password used to connect to the database. The password configured in this field must match the password configured in the database.

- **Retry Interval** field: If IWS is unable to connect to the database for any reason, it retries automatically to connect to the database after the number of seconds configured in this field have passed.
• **Advanced** button: After pressing this button, you have access to customize some settings. For most projects, the default value of these settings do not need to be modified and should be kept.

![Database Configuration: Advanced dialog](image)

• **Time Zone** combo box:
  • **Local Time + Time Difference**: Save the local time on the computer, plus the difference (bias) between the local time zone and Coordinated Universal Time (UTC).
  • **Local Time**: Save the local time only with no bias. This is not recommended.
  • **UTC**: Save the UTC time only. This is the default, and it is strongly recommended for most situations.

• **Milliseconds** combo box: You can configure how the milliseconds will be saved when saving the date in the database. Each database saves the date in different formats; for example, some databases do not support milliseconds in a Date field. The following options are available:
  • **Default**: Uses the format pre-defined for the current database. The databases previously tested by InduSoft are previously configured with the most suitable option. When selecting Default, IWS uses the setting pre-configured for the current database type. If you are using a database that has not been previously configured, the Default option attempts to save the milliseconds in a separate field.
  • **Disable**: Does not save the milliseconds at all when saving the date in the database.
  • **Enable**: Saves the milliseconds in the same field where the date is saved.
  • **Separate Column**: Saves the milliseconds in a separated column. In this case, the date is saved in one field (without the milliseconds precision) and the number of milliseconds is saved in a different column. This option is indicated where you want to save timestamps with the precision of milliseconds but the database that you are using does not support milliseconds for the Date field.

  **Tip:** The default option for each database is configured in the StADOSvr.ini file, stored in the \BIN sub-folder of IWS. See Studio Database Gateway for information about how to configure the StADOSvr.ini file.

• **Database Gateway**: Enter the Host Name/IP Address where the IWS Database Gateway will be running. The TCP Port number can also be specified, but if you are not using the default, you will have to configure the IWS Database Gateway with the same TCP Port. See the Studio Database Gateway section for information about how to configure the advanced settings for the IWS ADO Gateway.

• **Disable Primary Keys**: For some modules, IWS will try to define a primary key to the table in order to speed up the queries. If you are using a database that does not support primary keys (e.g., Microsoft Excel), then you should select (check) this option.

• **Disable Delimiters**: Select this troubleshooting option to disable the delimiters that are used to format communications with the database. Delimiters can cause problems when a Trend Control or Grid builds a query that includes aggregates such as Min and Max.

• **Disable SQL variables**: Select this troubleshooting option to disable SQL variables, such as @Value and ?, that are often used in SQL statements and queries. Some specific database providers do not support these variables.
Table Pane

This area allows you to configure the settings of the Table where the data will be saved. All tasks can share the same database. However, each task (Alarm, Events, Trend worksheets) must be linked to its own Table. IWS does not check for invalid configurations on this field, therefore you should make sure that the configuration is suitable for the database that you are using.

- **Use default name** checkbox: When this option is checked (default), IWS saves and/or retrieves the data in the Table with the default name written in the **Name** field.

- **Automatically create** checkbox: When this option is checked (default), IWS creates a table with the name written in the **Name** field automatically. If this option is not checked, IWS does not create the table automatically. Therefore, it will not be able to save data in the database, unless you have configured a table with the name configured in the **Name** field manually in the database.

- **Name**: Specifies the name of the Table from the database where the history data will be saved.

  **Tip:** To specify a sheet in a Microsoft Excel spreadsheet file, use the following syntax:

  \[sheetname$\]

- **Refresh** button: If the database configured is currently available, you can press the **Refresh** button to populate the **Name** combo-box with the name of the tables currently available in the database. In this way, you can select the table where the history data should be saved instead of writing the Table name manually in the **Name** field.

Run-Time Pane

This area allows you to set runtime values. The following fields are available:

- **Status** (output) checkbox: The tag in this field will receive one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disconnected from the database. The database is not available; your configuration is incorrect or it is an illegal operation.</td>
</tr>
<tr>
<td>1</td>
<td>The database is connected successfully.</td>
</tr>
<tr>
<td>2</td>
<td>The database is being synchronized.</td>
</tr>
</tbody>
</table>

- **Reload** (output): Specify a reload tag if you are using curly brackets in any of the configuration fields. When you want to reconnect to the database using the updated values on your tags, set the tag on this field to 1. IWS will update the configuration when trying to perform an action in the database, setting the tag back to 0 when it is finished.

**See also:**

Configuring a Default Database for All Task History.

**CONFIGURE THE PERFORMANCE CONTROL SETTINGS**

Configure the performance control settings to determine how memory is allocated for screen graphics during run time.

The project runtime client software has been improved to keep screen graphics (e.g., objects, images, fonts) in memory rather than load them from the hard drive each time a screen is opened. This makes opening and switching screens much faster, which in turn improves the overall run-time performance.
Devices that run the client software often have limited memory, however, so it is necessary to change the method of memory allocation as the memory becomes full.

When memory allocation is enabled (which it is by default in new projects) and your project is run, all unused objects are kept in memory so that project screens can be reopened or redrawn quickly. This method or state of memory allocation is called Always.

As the memory fills with objects, however, the amount of free memory decreases and may eventually reach the Dynamic Limit (i.e., the value configured in the Before starting dynamic allocation setting). When this happens, unused objects may be kept in memory but are removed if the space is needed for other objects that are actually being used. This method or state of memory allocation is called Dynamic.

As the memory continues to fill with objects — typically because the client has many project screens open and therefore many objects being used — the amount of free memory decreases until it finally reaches the Critical Limit (i.e., the value configured in the Before disabling allocation setting). When this happens, all unused objects are removed from memory and memory allocation is disabled until you restart the project. This method or state of memory allocation is called Critical.

**Tip:** Icon files (*.ICO) cannot be kept in memory. If you use many icon files in your project — particularly in Button objects — consider replacing them with new files in another format such as GIF, JPG, or PNG.

Performance control has default settings that should work for most projects running on most clients, but if you have problems, you can adjust the settings for your project or even disable memory allocation altogether. To configure the performance control settings:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**. The **Project Settings** dialog box is displayed, with the **Options** tab selected.
2. In the **Options** tab, in the **Performance Control** area, click **Configure**. The **Performance Control** dialog box is displayed.
3. To completely disable memory allocation, clear **Enable memory allocation**.
4. To adjust the limit at which memory allocation will change from Always to Dynamic, type a new value in the Before starting dynamic allocation box.

5. To adjust the limit at which memory allocation will change from Dynamic to Critical, type a new value in the Before disabling allocation box.

6. To restore the default settings, click Set Default.

You can monitor memory allocation during run time by calling the function GetPerformanceMetric.

**Viewer tab**

Use this tab to configure the project viewer and change certain runtime behaviors.

*Project Settings: Viewer tab*

- **Titlebar** checkbox and text field: Click (enable) this box and type a new name into the field provided to specify or change the default titlebar text for the Viewer window.

- **Minimize Box**, **Maximize Box**, and **Close Box** checkboxes: Click these boxes to display (enable) or hide (disable) these buttons on the Viewer window.

- **Start Maximized** checkbox: Click (enable) this box to maximize the Viewer window automatically when you run your project.

- **Menu** checkbox and **Options...** button: Click (enable) the checkbox and the Options... button to specify which menu options are available at runtime. When the Runtime menu options dialog displays (as follows), click the checkboxes to display (enable) or hide (disable) these menu options.
Creating a New Project

- **Resize Border** checkbox: Click (enable) this box to allow the user to resize the Viewer window during runtime.
- **Status Line** checkbox: Click (enable) this box to display the Status Line in the runtime project.
- **Startup screen** field: Click the combo box and select the screen (.SCR) or screen group (.SG) you want to display automatically when you open the runtime project.

**Note:** Another way to specify a screen or screen group as the startup screen is to right-click on it in the **Project Explorer** and then choose **Set as startup** from the shortcut menu.

- **Show ??? when quality is not GOOD** checkbox: Click (enable) this box to display question marks (???) instead of the tag's value when their quality is not good.
- **Hide Taskbar** checkbox: Click (enable) this box to hide the Windows taskbar by default.
- **Disable Palm Rejection**: Select this option to disable Palm Rejection during project runtime. Palm Rejection is a feature on Windows 7 touchscreen devices that detects and rejects accidental touches from the operator’s palm. However, it can somewhat slow the touchscreen’s responsiveness, so disabling it can improve the performance of the project runtime.

**Note:** Palm Rejection is not available on anything other than Windows 7 touchscreen devices, so selecting **Disable Palm Rejection** will have no effect in projects running on other target systems.

**Tip:** Windows 7 touchscreen devices also have an option to allow the user to press and hold on the screen to right-click. This is useful on consumer devices like tablets and smartphones, but it can interfere with the operation of an IWS project where the user needs to be able to press and hold buttons and other screen objects. The option is selected by default, so to disable the option on the device:

1. Click the **Start** button, and then on the Start menu, click **Control panel**. The **Control panel** window is displayed.
2. Double-click **Pen and Touch**. The **Pen and Touch** dialog is displayed.
3. Click the **Touch** tab.
4. From the **Touch actions** list, select **Press and hold**, and then click **Settings**. The **Press and Hold Settings** dialog is displayed.
5. Clear **Enable press and hold for right-click**.
6. Click **OK** to close the **Press and Hold Settings** dialog.
7. Click **OK** to close the **Pen and Touch** dialog.

- **Enable ToolTips** checkbox: Click (enable) this box to see Windows ToolTips when running your project. You can configure tooltips in the **Hint** field of the **Object Properties** dialog of each object.

- **Save pictures in separate files** checkbox: When this option is selected, images that you paste into a screen are automatically converted to Linked Picture objects and then saved as separate BMP image files in your project folder at **projectname\Web\Resources**. You can then reuse the images without increasing the size of your project, because every instance of the image will link to the same file. That should improve runtime performance.

When this option is cleared, images that you paste into a screen are kept as Bitmap objects and embedded in the screen file. Each instance of an image will be embedded, so if you use the same image more than once, then it may greatly increase the size of the screen file.

**Note:** The effect of this option is permanent: once a pasted image is converted to a Linked Picture object, clearing the option will not convert it back to a Bitmap object.

The Resources sub-folder is automatically maintained by the development application, and it is used only for the bitmap images that are pasted and converted. You should not put other image files in it, because they may be moved or deleted. Instead, put other images that you want to link to in the Web sub-folder.

- **Auto Screen Scaling** checkbox: Click (enable) this box to automatically scale project screens when you resize the Viewer window. This feature is available for local projects running on Windows PC or in the Thin Client. This parameter is not available for projects running on Windows Embedded target systems.
• **Enable Enhanced Graphics** checkbox: When this option is selected, your project screens will have enhanced graphics such as anti-aliasing and opacity.

Anti-aliasing smooths the lines, edges, captions, and gradient fills on Shapes (i.e., Line objects et al.), Text objects, and Button objects (Standard and OS styles). It is not supported for conical gradient fills.

Opacity (0% to 100%) allows you to make a screen object translucent and show other objects behind it. You can then arrange the objects to produce certain visual effects. To set opacity on an object, apply the Visibility/Position animation.

Enabling enhanced graphics might decrease run-time performance on some clients, so you should test it thoroughly before you deploy your project. Also, enhanced graphics are not supported on Windows Embedded devices; your project's target system must be a Windows computer.

• **Active area indication** group: Click (enable) the Show Object Edge and Change Mouse Cursor checkboxes in this area to modify the object edge and the mouse cursor when moving the cursor over any object where the Command animation has been applied.

• **Virtual Keyboard**: When this option is checked, the Virtual Keyboard is enabled for your project. (This option does not apply when your project is running on a Thin Client; for more information, see Project Settings: Web.) The keyboard allows the user to enter data during runtime on touchscreen groups — that is, without typing on a physical keyboard. For example, a Text object with the Text Data Link animation applied and the Input Enabled option checked.

You can establish a default configuration for the virtual keyboard:

• **Default**: Select the default keyboard type to be used in your project, when no keyboard type is specified by the calling object or function.

• **Scale**: Using this option, you can shrink or enlarge the keyboard to fit the size of the target display. A value of 100% represents the default size of each keyboard type.

• **Show Hint** checkbox and field: When this option is checked, a hint is displayed in the title bar of the keyboard. For individual objects, the hint is configured in the Object Properties dialog. Otherwise, type a string value in the Show Hint field to serve as a default hint.

• **Enable Min/Max Fields** option: When this option is checked, the minimum and maximum allowed values are displayed at the bottom of the keyboard. For objects, these values are configured in the Object Properties dialog. Otherwise, the Min and Max properties of the associated Tag are used by default.

  **Note:** The Min and Max fields are displayed only on the Keypad keyboard type, and only when the associated Tag is defined as Integer or Real. If Min is greater than Max, then input will be disabled. If Min/Max configured on the object is different from Min/Max configured in the Tag properties, then your project will attempt to scale the input accordingly.

• **Built-in Dialogs** combo box: Select or type the scale at which the built-in dialogs (i.e., Logon, E-Sign, and Virtual Keyboard) should be displayed during run time. This will make it easier to see and use the dialogs on project runtime clients with small, high-resolution displays. You can specify a scale between 100% (native resolution) and 400%, or you can select Auto which means the client will automatically select the best scale for the display.

• **Mouse Cursor** checkbox: Click (enable) this box to show the mouse cursor in the runtime project.

  **Note:** The Mouse Cursor option is not supported in Windows CE running on Armv4I processors.

• **Execute only topmost object commands** checkbox: This option controls how your project behaves when the user clicks in an area where two or more screen objects overlap. If this option is checked, then only the commands on the topmost object will be executed. If this option is not checked, then the commands on all of the overlapping objects will be executed.

  **Note:** The topmost object is the one with the highest ID number. (The ID number of an object is displayed in the status bar at the bottom of the development environment.) You can use the Move to Back / Move to Front and Move Backward / Move Forward tools to change the order in which objects are stacked.
• **Enable focus on buttons, commands and text objects**: This option is selected by default. When it is selected, clicking on a command button or text input during runtime will put focus on that object, as shown in the illustration below:

![Focus on text input (left), no focus on text input (right)](image)

After that, the end-user can press **Tab** to tab through all such objects in the screen or they can press **Enter** to activate the currently focused object (i.e., click on a command button, enter text in a text input), similar to a normal Windows application. This is useful if the client station has a physical keyboard and the end-user needs to quickly work through many such objects, because it saves time that would otherwise be spent repeatedly switching between the keyboard and the mouse or touchscreen.

To force the end-user to always use the mouse or touchscreen to activate screen objects, clear this option.

• **Use popup input for text objects**: Select this option to display a small popup for text inputs, as shown in the illustration below:

![Popup for text input](image)

This is an alternative to typing directly into the text object (which can seem like editing the screen itself and therefore be confusing to some end-users) and to displaying a Virtual Keyboard for input (which requires using the mouse or touchscreen).

**Note:** If the **Virtual Keyboard** option (above) is selected, then it will override this option.

Also, if the **Enable focus on buttons, commands and text objects** option (above) is cleared, then this option is automatically selected. This is to ensure there is some on-screen indication of which text input is currently active.

• **Multi-Touch Settings**: Configure the default Multi-Touch settings for all screens in your project. For more information, see **About the Multi-Touch settings for project screens** on page 258.

**Communication tab**

Use this tab to specify communication parameters relating to your project in general.

• **TCP area**: Configure the communication settings for the **TCP/IP Client and Server modules**:
• **Port** field: TCP Port used by the TCP/IP Client and TCP/IP Server tasks. When changing this value in the local project, be sure to change the same value in the remote project that is communicating with the local one.

This setting is required for all projects and cannot be left blank. Make sure that your network configuration will allow connections to this port on whichever station will be the project server.

**Note:** If you change the port to anything other than 1234, you will disable Mobile Access and Mobile Access Tabular.

• **Send Period (ms)** field: Period (in milliseconds) used between two consecutive messages sent by the TCP/IP Client or TCP/IP Server modules to update tag values in the remote station. Typically, a lower number equals better performance and higher traffic in the network.

• **Enable binary control** checkbox: Check this option to enable binary control when the TCP/IP Client or the TCP/IP Server module sends messages to the remote station. Binary control increases the security of the system; however, it decreases the efficiency (performance) of the communication. When connecting two stations via the TCP/IP Client and TCP/IP Server module, make sure this setting is either enabled in both projects or disabled in both.

• **Driver and OPC** menu: Select the method used by all communication drivers and OPC Client worksheets configured in the current project when writing values to the remote PLC/device:

  • **Send every state**: When the communication task is configured to write values upon a change of tag value, all changes in the tag value are buffered in a queue and sent to the device when the communication task (Driver or OPC) is executed.

  **Note:** There is a limit on the size of the buffer for tag value changes, to prevent accumulated changes from decreasing run-time performance. If the buffer size is exceeded, then only the most recent changes are kept until the next time the task is executed and the changes are sent. Also, a warning message is logged in the dump file.

  To adjust the buffer size, manually edit the following setting in your project file `{projectname}.APP`:

  ```
  [Options]
  DriverAndOpcBufferSize=5
  ```

  • **Send last state**: When the communication task is configured to write values upon a change of tag value, only the current (last) value of the tag is sent to the device when the communication task (Driver or OPC) is executed. When this method is selected, if the tag changed value more than once while the communication task was not being executed, the transient values of the tag are not sent to the device. This is the desired behavior for most projects.

• **Tag Integration** area: Use these settings to integrate tags from remote devices into your project's tags database. For more information, see About tag integration on page 170.

• **Preloading tags from server** area: To improve performance, the runtime viewer loads all of the Server tags on a screen into memory before it displays that screen. Configure the timeout settings for both remote and local viewers:

  • **Timeout when executing on remote** field: Specifies the time (in milliseconds) that a Secure Viewer or Thin Client running on a remote station will wait to load the tags.

  • **Timeout when executing on local** field: Specifies the time (in milliseconds) that the Viewer running on the local station (i.e., the Server) will wait to load the tags.

• **Execution Environment** area: Configure the communication settings for the Remote Management tool, which sends your project files to a target system:

  • **Timeout (ms)** field: Specifies the time (in milliseconds) that the project will wait to communicate with the target system.

  • **Enable File Compression** checkbox: Select this option to compress the system and project files before sending them to the target system. This may reduce the download time if you have a slow connection between your server and the target system. If you have a fast connection, however, then selecting this option may actually decrease performance because each compressed file must be decompressed.
on the target system before the next file is sent. Select this option only if you have an extremely slow connection, such as dial-up.) File compression is disabled by default.

**Preferences tab**

Use this tab to configure your preferences when building projects with the IWS development environment.

![Project Settings: Preferences tab](image)

**Warning Messages**

- **Display warning message before downloading screen to the target system** option: When this option is selected and IWS is connected to a target system (via Remote Management), you are prompted to download the updated screen to the target system immediately after saving a screen on the screen editor. If this option is not selected, the screen is downloaded automatically, regardless of any confirmation.

- **Display confirmation message when renaming project tags** option: When this option is selected and you modify the name of any tag in the Project Tags worksheet, you are prompted to replace the old tag name with the new tag name in the whole project. If this option is selected, IWS will execute the global replace command to replace the old tag name with the new tag name in all documents of your project (screens and worksheets).

- **Display confirmation message when changing the screen position and size** option: When this option is selected, you are prompted to update the screen attributes (Width, Height, Top and Left) after modifying them on the Layout interface.

- **Display warning message after saving symbols** option: When this option is selected, a warning message is shown after saving symbols.

- **Display warning message when replacing cells using driver browser** option: When this option is selected and you have used tag integration to connect to a remote device, a warning message is displayed when you replace a local tag in a worksheet with a tag selected from the remote device.

**Quality Feedback Service**

This section allows you to configure your project to generate log files and/or dump files that can be used to diagnose hardware and software problems, such as memory leaks and unexpected errors. These files are saved in the Dump sub-folder of the running project (\Documents\InduSoft Web Studio v7.1 Projects \projectname\Web\Dump).

- **Generate a log file when an unexpected error happens** option: When this option is selected, the runtime modules append the Log File (Web\Dump\Dump.txt) whenever an internal exception (error) occurs. These exceptions may not necessarily crash the runtime modules, but they can affect the stability of the system and should be investigated.
**Note:** The Log File is continually appended until it reaches its maximum size of 2MB. After it reaches its maximum size, the existing file is deleted and a new file is created.

- **Generate a dump file when an unexpected error happens** option: When this option is selected, the runtime modules generate a new Dump File (*.dmp) with useful information about the conditions of the error. This is a binary file that can only be read by your support representative.

**Note:** Dump Files are named WinXXX.dmp — where XXX is an identifying number (in hexadecimal format) automatically generated by the system — in order to prevent an existing file from being overwritten when a new error occurs. Therefore, if more than one error occurs, you will find multiple Dump Files in the directory. The Log File indicates the name of the Dump File associated with each error.

- **Enable log of memory** option: When this option is selected, the runtime modules append the Log File every 15 minutes with information about the current memory allocation. (The first log entry is written out 15 minutes after the runtime module is started.) This information can be used to identify memory leaks.

- **Enable detailed dump files** option: When this option is selected, the generated Dump Files will contain more detailed information than they normally do. Please note that these files take much more hard drive space, so you should select this option only when you are working with your support representative to troubleshoot a specific issue.

Even if none of these Quality Feedback options are selected, a post-mortem Dump File (WinDump.dmp) will always be generated when the runtime module is terminated by a fatal error. However, for debugging purposes, it is strongly recommend that you enable all options in this section and then send the Log File and all Dump Files to your support representative.

### Other Preferences

- **Reset Tags Database when starting project** option: When this option is selected, the project tags are automatically reset to their startup values whenever you run your project. For more information, see Reset Tags Database.

- **Enforce Web functionality equivalence in local project screens** option: When this option is selected, the development environment will warn you when you try to use functions or features that are not supported on both the project runtime server and the thin clients (i.e., Secure Viewer and Web Thin Client). This ensures that your project always behaves the same regardless of where it is viewed. For example, when this option is selected, the function Open cannot be called from Global Procedures or Script worksheets, because it is meaningful only when it is executed on stations that display project screens.

  **Note:** This option is cleared by default in order to maintain compatibility with previous versions of InduSoft Web Studio. If you do not need to maintain compatibility, you should select this option.

- **Auto reload project on Viewer/Web Clients when it is changed** option: When this option is selected, target systems (i.e., Thin Clients and Secure Viewers) will check the server to see if they have the most recent version of the project. If they do not, they will automatically download the new version from the server.

- **Disable high quality when resizing bitmaps to improve performance** option: When this option is selected, bitmaps in project screens are resized at lower quality. Normally, when bitmaps are displayed at anything other than actual size, they must be resampled to maintain high quality, and if this resampling must be done frequently and/or for many bitmaps in the same screen, it might decrease run-time performance.

- **Default project path** box: This is the location where new projects are saved by default when you create them. When you install the InduSoft Web Studio software, this is automatically set to C:\Users\username\Documents\InduSoft Web Studio v7.1 Projects\, but you can reset it to any location on your computer or network.

  **Note:** If you selected the Collaboration feature when you installed the InduSoft Web Studio software, you cannot change the Default project path setting, regardless of whether you are actually connected to a collaboration server. Instead, if you want to change where your projects are saved, you must remap your workspace in the Collaboration Server settings. For more information, see Collaboration.
Configuring your project's default email settings

Some features, such as alarms and certain functions, are able to send email to designated recipients. To use these features, you must configure your project’s email settings.

The email settings can be configured at any point during runtime by calling the CnfEmail function. However, you can also configure default settings that are automatically used when the project is first run and then restored as needed during runtime, overwriting any changes made by calling the CnfEmail function.

1. On the **Project** tab of the ribbon, in the **Web** group, click **Email/FTP**.
   The *E-mail and FTP configuration* dialog is displayed.
2. Click the **E-Mail Settings** tab.

3. In the **E-mail (From)** box, type your email address.
4. In the **Server** and **Port** boxes, type the server address and port number for your outgoing mail server.
   The default port for SMTP is 25, but it depends on your server and network configuration. Please consult your email provider.
5. If your outgoing mail server requires authentication, select **My server requires authentication**. If authentication must also be encrypted, select **Enable SSL**. Then type your credentials in the **User Name** and **Password** boxes.
   Most outgoing mail servers do require authentication, to prevent spamming and other abuse from unknown users.

   - **Note:** Encryption via SSL is not supported in projects running on Windows Embedded target systems.
6. In the **Status** box, type the name of a tag (Integer type) that will receive status codes when the project sends email.
7. In the **Reload** box, type a tag/expression. When the value of this tag/expression changes, the project will reload these default email settings.
8. Click **OK** to save your configuration and close the dialog.
Configuring your project’s default FTP settings

Some features in InduSoft Web Studio, such as certain functions, are able to transfer files between computers using FTP. To use these features, you must configure your project’s FTP settings.

The FTP settings can be configured at any point during runtime by calling the CnfFTP function. However, you can also configure default settings that are automatically used when the project is first run and then restored as needed during runtime, overwriting any changes made by calling the CnfFTP function.

1. On the Project tab of the ribbon, in the Web group, click Email/FTP. The Email/FTP Configuration dialog is displayed.
2. Click the FTP tab.

![Email/FTP Configuration dialog]

3. In the User Name and Password boxes, type your credentials for the FTP server.
4. In the Server and Port boxes, type the server address and port number.
   The default port for FTP is 21, but it depends on your server and network configuration. Please consult your server administrator.
5. Select Active or Passive mode, depending on the server’s configuration.
   Passive FTP mode can be used to bypass some network firewalls. Again, please consult your server administrator.
6. In the Status box, type the name of a tag (Integer type) that will receive status codes when the project transfers a file.
7. In the Reload box, type a tag/expression. When the value of this tag/expression changes, the project will reload these default FTP settings.
8. Click OK to save your configuration and close the dialog.
Execution Tasks dialog

The *Execution Tasks* dialog is used to configure which tasks must be automatically started when the project is run, as well as to manually start/stop tasks during runtime.

The *Execution Tasks* tab displays the list of available tasks for the current project. Their status and startup modes (*Automatic* or *Manual*) are also displayed.

![Execution Tasks dialog – Execution Tasks](image)

The following table lists the tasks that are not available for projects running on Windows Embedded target systems:

<table>
<thead>
<tr>
<th>Task</th>
<th>Available for Windows Embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background Task</td>
<td>Yes</td>
</tr>
<tr>
<td>Database Client Runtime</td>
<td>Yes</td>
</tr>
<tr>
<td>Database Spy</td>
<td>No</td>
</tr>
<tr>
<td>DDE Client Runtime</td>
<td>No</td>
</tr>
<tr>
<td>DDE Server</td>
<td>No</td>
</tr>
<tr>
<td>HDA OPC Server</td>
<td>No</td>
</tr>
<tr>
<td>Driver Runtime</td>
<td>Yes</td>
</tr>
<tr>
<td>LogWin</td>
<td>No</td>
</tr>
<tr>
<td>ODBC Runtime</td>
<td>No</td>
</tr>
<tr>
<td>OPC Client Runtime</td>
<td>Yes</td>
</tr>
<tr>
<td>Studio Scada OPC Server</td>
<td>Yes (see note)</td>
</tr>
<tr>
<td>TCP/IP Client Runtime</td>
<td>Yes</td>
</tr>
<tr>
<td>TCP/IP Server</td>
<td>Yes</td>
</tr>
<tr>
<td>Viewer</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Note:* In order to run the Studio Scada OPC Server task in EmbeddedView on a Windows Embedded Standard computer, you must first download and install the OPC Core Components.
on that computer. You can download the necessary software from the OPC Foundation website at: www.opcfoundation.org/DownloadFile.aspx?CM=3&RI=820&CN=KEY&CI=274&CU=7
You do not need to do anything for InduSoft Web Studio on Windows or CEView on Windows Embedded Compact, because the necessary software is installed automatically.

You can configure tasks for automatic execution when the project is run by selecting the task in the list, clicking **Startup**, and then selecting **Automatic**.

The tasks configured with Startup = Automatic are automatically executed when the project is run; the tasks configured with Startup = Manual are not.

You can also Start/Stop each runtime task by clicking the name and then pressing the **Start** or **Stop** button.

**Tip:** You can also start/stop each task during runtime by using the **StartTask** and **EndTask** built-in functions. You can also use the **IsTaskRunning** function to check if each task is running during runtime.
Run a project as a Windows service

Your IWS project can be configured to run under Windows services.

Microsoft Windows services, formerly known as NT services, allow you to create long-running programs that run in their own Windows sessions. These sessions can be automatically started when the computer starts up, can be paused and restarted, and do not show any user interface. These features make services ideal for use on a server or whenever you need long-running functionality that does not interfere with other users who are working on the same computer. You can also run services in the security context of a specific user account that is different from the logged-on user or the default computer account. For more information about services, please refer to the Microsoft Developer Network (MSDN) Library.

Why would you want to run your project under Windows services?

- To ensure that your project always runs with whatever system privileges it needs, regardless of the privileges of the user that is currently logged on to Windows;
- To prevent the user from interfering with your project while it is running; or
- To let your project keep running when there is no user logged on at all.

Create and configure the Windows service

Note: All of the procedures described below were tested using Microsoft Windows 7.

Also, to perform these actions, you must be logged on as a user with Administrator privileges and you should know how to use the Computer Management console. (To access the console, right-click the Computer icon, and then click Manage on the shortcut menu.)

There are two ways to create and configure the Windows service for your project: you can use the Service Configuration tool in the IWS development environment, or you can use the command-line utility that is installed with the InduSoft Web Studio software.

Service Configuration Tool

You can configure and run a new service from within the development environment by clicking Service on the Project tab of the ribbon. This opens the Service Configuration dialog box:

Service Configuration dialog box

Application

The location of the project file (projectname.app) that the service will load and run when it is started. This must be a complete file path. Use the browse button (...) to find and select the project file on your computer.

User

The Windows user account under which the service will run. This is an optional setting; if it is not used, then the service will run under Local System.
Note: Try to avoid running the service under Local System. That account has too much privilege to the file system and too little privilege to run the OPC Client and Server modules properly. The best alternative is to create a user solely to run IWS and configure its privileges to fit the needs of your project. For more about this, see "Configuring User Privileges" below.

Password

The password for the specified user account. This is an optional setting; it is not needed if no user is specified or if the specified user does not have a password.

Startup Type

How the Windows service will start. The following options are available:

- **Automatic**: The service will start automatically when the computer starts up.
- **Manual**: The service can be started manually in the Computer Management console or by clicking Start, as described below.
- **Disabled**: The service will be created and then disabled. It cannot run until a user with Administrator privileges enables it in the Computer Management console.

Action pane

Start or stop the service. Please note that these buttons are not enabled until the service is actually created.

Creating a New Service

To create a new service:

1. Next to the Project box, click ... to open a standard Windows file browser. Use the browser to find and select your project file.
2. In the User and Password boxes, type the username and password (if any) for the Windows user account under which the service will run.
3. Select a Startup Type.
4. Click Apply. The service is created with the specified settings.

After the service has been created, it will appear in the Services console (Start > Control Panel > System and Security > Administrative Tools > Services) under the name "InduSoft Web Studio". You can use that console to quickly stop and restart the service, if you do not want to run the IWS development application.

Command-line Utility

You can also configure the service by using the command-line utility, StdSvcInst.exe. It offers a few more options than the Service Configuration tool described above — such as specifying a name and description for the service — and it can be used without running the IWS development application. The utility is located in the Bin folder of your IWS program directory. To run the utility, open a command prompt, navigate to the Bin folder (C:\Program Files\InduSoft Web Studio v7.1\Bin), and then enter the command with the desired parameters.

The utility has the following command syntax:

```
StdSvcInst { -create -app filepath -startup { auto | manual | disabled } -user username -password password -name displayname -descr description | -start | -stop | -delete }
```

- **-create**
  Creates the Windows service.

- **-app filepath**
  Specifies which project file (projectname.app) the service will load and run when it is started. (This is the same as the Project box in the Service Configuration dialog.) You must include the complete file path, and it must be enclosed in quotes.
This parameter is required when you create a new service.

- **startup \{ auto | manual | disabled \}**
  Specifies how the service will start. (This is the same as the Startup Type in the Service Configuration box.) This parameter is optional; if it is not used, then the default behavior for a new service is manual.

- **user username**
  Specifies the Window user account under which the service will run. (This is the same as the User box in the Service Configuration dialog.) This parameter is optional; if it is not used, then the service will run under Local System.

- **password password**
  Specifies the password for the given user account. (This is the same as the Password box in the Service Configuration dialog described above.) This parameter is optional; it is not needed if no user is specified or if the specified user does not have a password.

- **name displayname**
  Defines the service name that is displayed in the Computer Management console. The name must be enclosed in quotes. This parameter is optional; the default name is "Studio".

- **descr description**
  Defines the service description that is displayed in the Computer Management console. The description must be enclosed in quotes. This parameter is optional.

- **start**
  Starts the service. This is the same as starting the service using the Computer Management console or by clicking Start in the Service Configuration dialog.

- **stop**
  Stops the service. This is the same as stopping the service using the Computer Management console or by clicking Stop in the Service Configuration dialog.

- **delete**
  Deletes the service.

**Example: Creating the Service**

In this example, we want to create a new Windows service with the following options:

| IWS Project File | 
|------------------|---|
| C:\\Users\\username\\Documents\\InduSoft Web Studio v7.1 Projects \projectname\\projectname.app |
| Startup Mode | 
| Automatic |
| User | 
| IWS |
| Password | 
| IWS |
| Service Name | 
| "InduSoft Web Studio" |
| Service Description | 
| "Starts IWS project" |

Note that the system must already have a user account named "IWS" with password "IWS".

So, to create the service with the desired options:

1. Make sure you are logged on as a user with Administrator privileges.
2. Open a command prompt (Start > All Programs > Accessories > Command Prompt).
3. Navigate to the Bin folder:

   `cd "C:\Program Files\InduSoft Web Studio v7.1\Bin"`
4. Enter the command:

```
StdSvcInst -create -app "C:\Users\username\Documents\InduSoft Web Studio v7.1 Projects\projectname\projectname.app" -startup auto -user IWS -password IWS -name "InduSoft Web Studio" -descr "Starts IWS project"
```

If the procedure is successful, then the system will display the message **Service created**. Otherwise, it will display an error message.

**Example: Changing the Project File**

After you create the service, you may want to change the IWS project file that it runs. You can do this by using the `-app` parameter:

1. Make sure you are logged on as a user with Administrator privileges.
2. Stop the service if it is running.
3. Open a command prompt.
4. Navigate to the `Bin` folder.
5. Enter the command — for example, to set `MyProject` as the project file:

```
StdSvcInst -app "C:\Users\username\Documents\InduSoft Web Studio v7.1 Projects\MyProject\MyProject.app"
```

**Example: Deleting the Service**

To delete the service:

1. Make sure you are logged on as a user with Administrator privileges.
2. Stop the service if it is running.
3. Open a command prompt.
4. Navigate to the `Bin` folder.
5. Enter the command:

```
StdSvcInst -delete
```

**Configure user privileges**

The service will run under the privileges of the user account specified in the **User** field of the **Service Configuration** tool (or by the `-user` parameter of the command-line utility). If IWS needs some system resource to which that account does not have privileges, it will fail. Therefore, you must configure the account to have the necessary privileges.

**Note:** The following actions can be performed only by a user with Administrator privileges.

**Enabling the User Account to Log On as a Service**

Before anything else, the specified user account must be enabled to log on to the computer as a service. To enable the account:

2. In the console window, select the **User Rights Assignment** folder (Security Settings > Local Policies > User Rights Assignment).
3. In the list of policies, double-click **Log on as a service**.
   The **Log on as a service** dialog is displayed.
4. Click **Add User or Group**.
   The **Select Users or Groups** dialog is displayed.
5. Type the name of the user account under which you want the service to run.
6. Click OK.

**Giving the User Account Full Control Over the Project Folder**

For your IWS project to run properly, the specified user account must have full control over the project folder and all of the files in it. To give the account those privileges:

1. In *Windows Explorer*, locate your IWS project folder (i.e., the folder that contains the file `projectname.app`).
2. Right-click the folder, and then click Properties on the shortcut menu.
3. In the properties sheet, click the Security tab, and then click Edit.
4. In the Permissions dialog box, click Add, and then add the user account that you specified when you created the service.
5. Select the user that you added, and then in the list of permissions, set Full Control to Allow.
6. Click OK to apply your changes and close the dialog, and then click OK again to close the properties sheet.

**Allowing the User Account to Run the OPC Client/Server Module**

As mentioned previously, normal users have too few privileges to properly run the OPC Client/Server module. Therefore, you must configure the user account to have those privileges:

1. Open the Component Services console (Start > Control Panel > System and Security > Administrative Tools > Component Services).
2. In the console window, select the DCOM Config folder (Console Root > Component Services > Computers > My Computer > DCOM Config).
3. In the DCOM Config pane, right-click Studio Scada OPC Server, and then click Properties on the shortcut menu.
4. In the properties sheet, click the Identity tab.
5. Select This user and then complete the fields with the same username and password that you specified when you created the service.
6. Click OK to apply your changes and close the properties sheet.
7. Close the Component Services console.

**Troubleshooting**

When you run your IWS project as a Windows service, it has no user interface. Therefore, if an error occurs, it will only be logged as a Windows application event. You can check the messages by using the Event Viewer console (Start > Control Panel > System and Security > Administrative Tools > Event Viewer).
Tags and the Project Database
About Tags and the Project Database

Tags are a core component of any IWS project. Simply put, tags are variables used by IWS to receive and store data obtained from communication with plant floor devices, from the results of calculations and functions, and from user input. In turn, tags can be used to display information on screens (and Web pages), to manipulate screen objects, and to control runtime tasks.

But tags are more than simple variables. IWS includes a real-time database manager that provides a number of sophisticated functions such as time-stamping of any value change, checking tag values against runtime minimum and maximum values, comparing tag values to alarming limits, and so on. A IWS tag has both a value and various properties that can be accessed, some at development and others only at runtime.

All tags are organized into one of the following categories, which are represented by folders on the Global tab of the Project Explorer:

- **Project Tags** are tags that you create during project development. Places where project tags are used include:
  - Screen tags
  - Tags that read from/write to field equipment
  - Control tags
  - Auxiliary tags used to perform mathematical calculations

- **Shared Database** tags are created in a PC-based control program and then imported into IWS’s tags database.
  
  For example you might create tags in SteepleChase and import them into IWS so IWS can read/write data from a SteepleChase PC-based control product.
  
  You cannot modify shared tags within IWS — you must modify the tags in the original PC-based control program, and then re-import them into the Tags database.

- **System Tags** are predefined tags with predetermined functions that are used for IWS supervisory tasks. For example,
  
  - Date tags hold the current date in string format
  - Time tags hold the current time in string format
  Most system tags are read-only, which means you cannot add, edit, or remove these tags from the database.

  To see a list of the system tags, select the Global tab in the Project Explorer, open the System Tags folder, and open the Tag List subfolder. The above figure shows a partial list of system tags.
After creating a tag, you can use it anywhere within the project, and you can use the same tag for more than one object or attribute.

**Project Tags Folder**

The *Project Tags* folder contains all tags created and customized by the user. You can create project tags for displays, to read from and write to field equipment, for control, to perform mathematical calculations, and so forth.

To update a list of project tags, right-click on the *Project Tags* folder (or *Datasheet View* icon) and select the *Refresh* option.

⚠️ **Important:** Before deleting a tag, we strongly recommend using the *Object Finder* tool (on the Home tab of the ribbon) to verify that you are not using the tag in another part of the project (screens, math sheets, so forth). If you delete a tag from the project database that is being used in another part of the project, you will cause a compiling error and the project will function poorly.

To create a new tag, right-click on the *Project Tags* folder, the *Tag List* sub-folder, or *Datasheet View* icon and select *Insert Tag* from the shortcut menu. You also can click *Tag* on the Insert tab of the ribbon.

The *New Tag* dialog displays, as shown in the following figure:

![New Tag dialog](image)

Use this dialog to specify the following parameters:

- **Name** field: Type a name for the new tag. The first character must be a letter and you can use up to 255 characters in the name.
- **Array Size** field: Type a value to specify the size of the tag. Any size greater than 0 implies that the tag is an array.
- **Type** combo-box: Select a standard tag type from the list (*Boolean*, *Integer*, *Real*, or *String*). You also can define new types as structures formed by the classes.
- **Description** text box: Type a tag description for documentation purposes.
- **Scope** combo-box: Click to select one of the following options:
  - **Server** (default): The tag is maintained on the project server and it's shared by all connected clients (e.g., Thin Client, Secure Viewer). A change to the tag value affects the entire project.
  - **Local**: A virtual copy of the tag is maintained separately on each local station (server + clients), and a change to the tag value affects only the station on which the change was made.

These options have no affect on projects that do not have Web capabilities. If you select a **Scope** option for a project with Web capabilities, then any object property using the Local tag will not work properly over the Web.

⚠️ **Note:** You must create unique tag names. You cannot create a tag that uses the name of an existing tag.

You can view or edit the properties of a tag from either of the following dialoges:
• *Tag Property* dialog: Click *Properties* on the Home tab of the ribbon when the tag name displays in the *Tag name* field or double-click on the tag name in the *Tag List* subfolder located in the *Project Tags* folder.

• *Project Tags* dialog: Click the *Datasheet View* icon in the *Project Tags* folder.

The *Project Tags* datasheet includes columns for many of the tag properties.

![Project Tags datasheet](image)

Use this dialog to create, modify, or delete tags or tag properties. You can right-click on a tag property and use standard Windows commands to cut (Ctrl+X), copy (Ctrl+C), or paste (Ctrl+V), any tag and its properties. You can also undo (Ctrl+Z) the last modification to a field.

**Tip:** You can sort the data in the *Project Tags* sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the shortcut menu.

**EXTENDING THE PROJECT TAGS DATASHEET**

The Project Tags worksheet can be extended up to 65,488 rows, if necessary.

The datasheet is normally limited to a maximum of 32,721 rows. (This is separate from the maximum size of the project database as a whole, as well as the runtime limit that is set when you select a target platform for a new project.)

To extend the worksheet, edit your project file (*project_name*.app) to include the following entry:

```plaintext
[Options]
EnableExtendedTagCount=1
```

Doing so, however, brings the following restrictions:

• Project tags in rows 32,722–65,488 of the worksheet cannot be used as array indices in expressions. That is, in an expression like `Abs(numArray[indexTag])`, `indexTag` cannot be in that range of rows. (This restriction does not apply to the VBScript interface.)

• In a Class worksheet, only the first 32 class members can have alarms. For all class members after the first 32, alarms will not work.

Generally speaking, extending the Project Tags datasheet stretches the capabilities of IWS and should be done only when it’s absolutely necessary. It is better to design your project to conserve tags.

**About classes**

*Class* tags are compound tags that permit a high-degree of encapsulation within the Tags database. Where basic tags receive a single value, *classes* are designed to receive multiple values.

You can create a class-type tag by grouping basic or array tags, which then become the class *members*. The maximum number of members for any class depends on the product specification.
You specify class-type tags in one of two formats:

- For a simple class tag the syntax is `TagName.ClassMemberName`. (Where the period is used as a separator.) For example, if you wanted to monitor several different conditions (such as `temperature`, `level` and `pressure`) in a tank, you might create a class tag as follows:
  - `Tank.Temperature`
  - `Tank.Level`
  - `Tank.Pressure`

- For creating a complex class tag (using an array tag) the syntax is `ArrayTagName[ArrayIndex].ClassMemberName`. (Where again, the period is used as a separator.) If you wanted to monitor the `temperature`, `level`, and `pressure` conditions in multiple tanks, you might create a class tag as follows:
  - `Tank[tk].Temperature`
  - `Tank[tk].Level`
  - `Tank[tk].Pressure`

Where `tk` is an array index, representing the tank number.

### Classes Folder

The `Classes` folder contains all of the project classes and their respective members. Classes are compound tags consisting of user-defined data-type structures or tag types (**Integer**, **Real**, **Boolean**, and **String**). Classes allow for high-level encapsulation in the project database. A class-type tag provides a set of values for its members.

To define a class you must define the **members** and their types. Class members are variables that hold values for an object with particular characteristics. Thus, the defining a class can be very useful for projects with a repeating group of variables.

#### Note:

When you create a class folder, a **Class** icon displays in the **Tag List** subfolder located in the **Project Tags** folder.

To access the members of a class, use the following syntax with a period (.) as the separator: `TagName.MemberName`. For example: `tk.LEV` or `tk.TMP`.

If the `Tank` tag is an array, you use the following syntax:

```
ArrayTagName[ArrayIndex].MemberName
```

For example: `Tank[1].Level` or `Tank[n].Temperature`

A class-type tag contains a set of values (rather than a single value) associated with the class. You create class-type tags by grouping simple tags, which become the members. The maximum number of members for any class depends on the product specification. Class members can hold standard **Integer**, **Real**, **Boolean**, and **String** values, as mentioned previously.

To create a new class, use one of the following methods to open the **Insert Class** dialog:

- On the Insert tab of the ribbon, in the Global group, click **Class**;
- Right-click on the **Classes** folder, the **Members List** sub-folder, or the **Datasheet View** icon in the **Classes** folder; or
- Create a new class tag in the **Project Tags** folder.
When the *Insert Class* dialog displays, enter a class name in the **Name** field, and then click **OK** to close the dialog.

![Insert Class dialog](image)

**Note:** You must type a unique class name. You cannot create two classes with the same name. In addition, you cannot configure shared tags and system tags as classes.

IWS saves class folders in the *Tag List* subfolder (located in the *Project Tags* folder). You can edit the classes in this folder.

When the *Class datasheet* displays, you can use it to create, modify, or delete any class members and their viewable properties. (You cannot edit classes from the *Tag Property* dialog.)

![Class datasheet](image)

**Note:** The *Classes* folder can contain up to 16,384 classes and up to 4,096 members per class. This is a technical limitation of the tags database, not a licensing restriction; unused classes do not count against the total number of tags used.

When a new tag is created with a class type, however, each class member counts as a tag used because each member holds a value. (For example, if you create a class with 5 members and then create 5 tags with that class type, then you have a total of 25 tags used.) The total number of tags used cannot exceed the number of tags supported by the project’s target system / runtime license.

To edit a class member or property, you can right-click on the item and use standard Windows commands to cut (Ctrl+X), copy (Ctrl+C), or paste (Ctrl+V). You can also undo (Ctrl+Z) the last modification to a field.

You also edit member properties as follows:

- **Name** field: Type a name for the member or member property. The first character must be a letter and you can use up to 255 characters in the name.
- **Type** combo-box: Select a member type (*Boolean*, *Integer*, *Real*, or *String*).
- **Description** field: Type a description of the member property for documentation purposes.

**Note:** Members of a class cannot be of another class type.

Also, you must create a unique class name. You cannot reuse the name of an existing class. However, you can create members with the same name in different classes.

To delete a class and all its members, right-click on a class folder and select delete. IWS disables the delete option if you are running any runtime tasks. In addition, you cannot delete a class if it is associated with any tag.
Shared Database Folder

The *Shared database* folder contains all tags shared between IWS and the selected PC-based Control device. You must create and modify these tags in the PC-based Control software, and then they will be imported automatically into IWS under the following conditions:

- When you start IWS;
- When you right-click on the *Shared Tags* folder, which refreshes (updates) the database.

**Note:** Each PC-based Control software package has its own individual interface characteristics and conditions that allow IWS to import its tags. For example, the PC-based Control software application might have to be running for you to import database tags into IWS.

If there are no PC-based Control software products associated with the application, the *Datasheet View* and *Tag List* subfolders (located in the *Shared Database* folder) will be empty.

You cannot edit shared tags in the IWS environment, but you can modify them in the PC-based Control software. You can, however, configure shared tags in any IWS task like any other tag. Shared tags are read-only, and viewable on the *Tag Property* dialog and the *Shared Tag* datasheet.

**Note:** Right-click on the *Shared Database* folder (or click *Datasheet View*) and select the *Refresh* option to update your last "version" of the PC-based Control software's tags database. To change the Shared tags database (create a new tag, delete tags, or change tag properties), you must activate this command update the IWS shared database.

To view the *Tag Property* dialog:

- Click on the *Tag Properties* tool on the *Tag Properties* toolbar (the tag name must be in the *Tag name* field); or
- Double-click on the tag name located in the *Tag List* subfolder (*Project Tags* folder)

When the *Shared Tag* datasheet displays, it contains four columns (*Name*, *Size*, *Type*, and *Description*). This datasheet is read-only, you can use it to view shared tags only.

System Tags Folder

The *System Tags* folder contains predefined tags that have specific functions (time, date, acknowledge alarms, storage of the logged user, and so forth). You cannot edit or delete these tags; but you can access their values from any IWS task, copy them, and use them elsewhere.

**Note:** To update IWS's shared database with the system tags files, right-click on the *System Tags* folder or *Datasheet View* icon, and then click the *Refresh* option.

For a list of system tags, including their properties and descriptions, see *List of System Tags*.

You can view the properties of a system tag using the *System Tags* datasheet, which contains four columns (*Name*, *Size*, *Type*, and *Description*).

**Important:** Most system tags are read-only. To change the time, for example, you must use the proper math function and set the system time rather than writing to the system time tag.
Designing a Tag

Understanding the Tag Name Syntax

Observe the following guidelines when naming a tag:

- Your tag names must be unique — you cannot specify the same name for two different tags (or functions). If you type an existing tag name, IWS recognizes that the name exists and will not create the new tag.
- You must begin each tag name with a letter. Otherwise, you can use letters, numbers, and the underscore character (_) in your tag name.
- You cannot use the following symbols in a tag name:
  ` ~ ! @ # $ % ^ & * ( ) - = \ + \ [ \ ] { } < > ?
- You can use a maximum of 255 characters for a tag name or a class member name. You can use uppercase and lowercase characters. Tag names are not case sensitive. Because IWS does not differentiate between uppercase and lowercase characters, you can use both to make tag names more readable. (For example: TankLevel instead of tanklevel.)
- Tag names must be different from system tag names and math functions.

Some valid tag examples include:

- **Temperature**
- **pressure1**
- **count**
- **x**

Choosing the Tag Type

IWS allows you to create the following types of tags:

- **Basic tags** hold a single value.
- **Array tags** are a set of tags that use the same name with unique indexes.
- **Class tags** are a set of compound tags that consist of user-defined data types (Boolean, Integer, Real or String) or data-type structures.
- **Indirect tags** are pointers that provide indirect access to another tag type, including class tags.

A discussion of these tag types follows.

Basic Tags

A basic tag receives a single value. Typically, most tags defined for a project are basic tags. Some examples of a basic tag include:

- **TankID** (to identify different tanks in your project)
- **Temperature** (to identify the current temperature of an object)
- **Status** (to identify whether an object is open or closed)

Array Tags

An array tag consists of a set of tags that all have the same name, but use unique array indexes (a matrix of n lines and one column) to differentiate between each tag. An array index can be a fixed value, another tag or an expression. Maximum array sizes are determined by product specifications.

You can use array tags to:
• Simplify configurations
• Enable multiplexing in screens, recipes, and communication interfaces
• Save development time during tag declaration

You specify array tags in one of two formats:
• For a simple array tag, type:

  `ArrayTagName[ArrayIndex]`

• For a complex array tag (where the array index is an expression consisting of a tag and an arithmetic operation), type:

  `ArrayTagName[ArrayIndex+c]`

Where:
• `ArrayTagName` is the tag name;
• `[ArrayIndex]` is the unique index (fixed value or another tag);
• `+` is an arithmetic operation; and
• `c` is a numerical constant.

**Note:**
• You must specify a maximum index for each array tag by typing a value (n) in the Array Size column of an Project Tags datasheet or in the Array Size field on a New Tag dialog. (See "Creating project database Tags").

  When you create an n-position array tag, IWS actually creates n+1 positions (from 0 to n). For example, if you specify `ArrayTag[15]`, the array will have 16 elements, where 0 is the start position and 15 is the end position.

• You must not use spaces in an array tag.

  When IWS reads a tag it begins with the first character and continues until it finds the first space or null character. Consequently, the system does not recognize any characters following the space as part of the array tag.

  For example, if you type `a[second +1]` IWS regards `a[second` as the tag and considers it invalid because IWS does not find (recognize) the closing bracket. However, if you type `a[second +1]`, this is a valid array tag.

You can specify an array tag wherever you would use a variable name. Also, because array tags greatly simplify configuration tasks and can save development time, we suggest using them whenever possible.

For example, suppose you want to monitor the temperature of four tanks. The conventional configuration method is:
• `temperature1` — high temperature on tank 1
• `temperature2` — high temperature on tank 2
• `temperature3` — high temperature on tank 3
• `temperature4` — high temperature on tank 4

You can use array tags to simplify this task as follows (where [n] represents the tank number):
• `temperature[n]` — high temperature on tank n

The following table contains some additional examples of an array tag:
### Array Tag Examples

<table>
<thead>
<tr>
<th>Array Tag Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank[1], Tank[2], Tank[500]</td>
<td>Simple arrays, where the array indexes (1, 2, and 500) are numerical constants. For example, tank numbers.</td>
</tr>
<tr>
<td>Tank[tk]</td>
<td>A simple array, where the array index (tk) is a tag. For example, a tag representing the tank number.</td>
</tr>
<tr>
<td>Tank[tk+1]</td>
<td>A complex array, where the array index (tk+1) is an expression. For example, the value of tk (tank number) plus 1.</td>
</tr>
</tbody>
</table>

Note: When using another tag to reference the index of an array, if the value of the tag is outside the size of the array, then the following results are given:
- If IndexTag is greater than the size of the array, then MyArray[IndexTag] will point to the end position of the array; and
- If IndexTag is less than 0, then MyArray[IndexTag] will point to the start position of the array (i.e., MyArray[0]).

### Indirect Tags

Indirect tags "point" to other database tags (including class-type tags). Using indirect tags can save development time because they keep you from having to create duplicate tags (and the logic built into them).

You create an indirect tag from any string-type tag simply by typing the @ symbol in front of the tag name @TagName.

- To reference a simple tag, assume the strX tag (a string tag) holds the value "Tank", which is the name of another tag, then reading from or writing to @strX provides access to the value of the Tank tag.
- To reference a class-type tag and member, you simply create a string tag that points to the class tag and the member. For example, if a tag strX (a string tag) holds the value "Tank.Level", which is the name of the class tag, then reading from or writing to @strX provides access to the value of the Tank.Level member.
- You can also point directly to a class-type tag member; by identifying a class-type that points to a class member. For example: to access the Tank.Level member of the class, you must store the "Tank" value within the strX tag and use the syntax, @strX.Level.

### Choosing the Tag Data Type

Another consideration when designing a tag is what type of data the tag will receive. IWS recognizes the following, standard tag data types:
- **Boolean** (one bit): Simple boolean with the possible values of 0 (false) and 1 (true). Equivalent to the "bool" data type in C++. Typically used for turning objects off and on or for closing and opening objects.
- **Integer** (four bytes): Integer number (positive, negative, or zero) internally stored as a signed 32-bit. Equivalent to the "signed long int" data type in C++. Typically used for counting whole numbers or setting whole number values. Examples: 0, 5, −200.
- **Real** (floating point, eight bytes): Real number that is stored internally as a signed 64-bit. Equivalent to the "double" data type in C++. Typically used for measurements or for decimal or fractional values.
- **String** (alphanumeric data, up to 1024 characters): Character string up to 1024 characters that holds letters, numbers, or special characters. Supports both ASCII and UNICODE characters. Examples: Recipe product X123, 01/01/90, *** On ***.

You can also assign a new tag to a class that you have previously created.

You can find these tag types (and their respective icons) in the Global tab of the Project Explorer.

See also: Understanding Tag Properties and Parameters
• **Server** (default): The tag is maintained on the project server and it's shared by all connected clients (e.g., Thin Client, Secure Viewer). A change to the tag value affects the entire project.

• **Local**: A virtual copy of the tag is maintained separately on each local station (server + clients), and a change to the tag value affects only the station on which the change was made.
Creating Database Tags

Adding Tags to the Datasheet

Use the following steps to create tags from the Project Tags datasheet:

1. Select the Global tab and open the Project Tags folder.
2. Double-click the Datasheet View icon to open the Project Tags datasheet:

3. Locate an empty line in the datasheet and configure the following fields.

   **Tip:** You can use the keyboard Tab key to move to the next column.

   - **Name** field: Type a name using the proper syntax. (For more information, see "Tag Syntax").
   - **Array Size** field:
     - For an array tag, type a value to specify the maximum index of the array.
     - For any other tag type, type zero (0).
   - **Type** combo-box: Click the arrow to select a tag data type (Boolean, Integer, Real, or String) from the list. (If necessary, review "Choosing a Tag Data Type").
   - **Description** field (optional): Type a description for documentation purposes only.
   - **Scope** combo-box: Click the arrow to specify whether the tag value will be shared with (displayed on) Thin Client stations. (For more information, see "Choosing the Tag Scope").

4. Click in a new line to create another tag, or if you have no other tags to create, then save and close the Project Tags datasheet.

The following example shows a variety of tags configured in an Project Tags datasheet.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Array</th>
<th>Type</th>
<th>Description</th>
<th>Scope</th>
<th>Startup</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4:1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Example of Project Tags datasheet
Creating Tags "On-the-Fly"

Instead of opening the Project Tags datasheet every time you want to create a new tag, you can create individual tags "on-the-fly" by performing any of the following actions:

- On the Insert tab of the ribbon, in the Global group, click Tag;
- In the Project Explorer, right-click on the Project Tags folder, the Datasheet View icon, or the Tag List subfolder and then select Insert Tag from the shortcut menu; or

Inserting a Tag

- Type a new tag name into any Tag/Expression text field (available from Object Properties dialogs, worksheets, and so forth). When the Question dialog asks if you want to create a new tag, click Yes.

Creating a New Tag

Any of these actions causes a New Tag dialog to display, which you can then complete as needed. For more information, see "Configuring a New Tag".

Editing Tags

You can change the properties of a tag at any time during development or runtime. This section describes two methods you can use to edit tags.

Note: You can right-click on a tag property and use standard Windows commands to cut (Ctrl +X), copy (Ctrl+C), or paste (Ctrl+V) any tag and its properties. You can also Undo (Ctrl+Z) the last modification to a field.

From the Project TagsDatasheet

Use the following steps to edit one or more tags in the Project Tags datasheet:

1. Select the Global tab, open the Project Tags folder, and double-click on the Datasheet View button.
2. When the Project Tags datasheet opens, locate your tag.
3. Double-click in the column containing the information to be changed, and then type the new information into the datasheet.

If you changed a tag name, the Confirm Global Replace dialog box is displayed. Click Yes to replace the tag throughout your project, so that all objects, animations, tasks, and scripts will keep using the same renamed tag. Click No to change the tag name only, but be aware that if the old tag name is still used anywhere in your project, it will cause compiler errors when you try to run your project. To find where the old tag name is used, verify your project.

When the Confirm Global Replace dialog box is displayed, you can also select the Do not display this dialog box again check box. If you select the check box, you will no longer be able to globally replace tags by editing the datasheet. It is equivalent to always clicking No in the Confirm Global Replace dialog box, as described above.
4. When you are finished editing, save your changes to the tags database.

Tip: You can sort the data in the Project Tags sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the shortcut menu.

From the Tag List Folder

Use the following steps to edit one or more tags from the Tag List folder:

1. Select the Global tab, open the Project Tags folder, and double-click on the Tag List folder to view a list of all your tags.

2. Locate your tag and double-click on the tag name to open a Properties dialog.

Note: You also can right-click on the tag’s icon and choose Properties from the shortcut menu.

The Properties dialog contains fields and combo-boxes that correspond in name and function to the columns on the Project Tags datasheet.

3. Make your changes in the Properties dialog as follows:
   - To change the current Type or Scope properties, click the arrow button and select the new information from the list.
   - To change the Size or Description, highlight the existing text and type the new information into the text box.

4. Click OK to save your changes to the tags database and close the Properties dialog.

Tip: You can sort the data in the Project Tags sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the shortcut menu.
About classes

*Class* tags are compound tags that permit a high-degree of encapsulation within the Tags database. Where basic tags receive a single value, *classes* are designed to receive multiple values.

You can create a class-type tag by grouping basic or array tags, which then become the class *members*. The maximum number of members for any class depends on the product specification.

You specify class-type tags in one of two formats:

- For a simple class tag the syntax is *TagName.ClassMemberName*. (Where the period is used as a separator.)
  
  For example, if you wanted to monitor several different conditions (such as *temperature*, *level* and *pressure*) in a tank, you might create a class tag as follows:
  
  - `Tank.Temperature`
  - `Tank.Level`
  - `Tank.Pressure`

- For creating a complex class tag (using an array tag) the syntax is `ArrayTagName[ArrayIndex].ClassMemberName`. (Where again, the period is used as a separator.)

  If you wanted to monitor the *temperature*, *level*, and *pressure* conditions in multiple tanks, you might create a class tag as follows:
  
  - `Tank[tk].Temperature`
  - `Tank[tk].Level`
  - `Tank[tk].Pressure`

  Where `tk` is an array index, representing the tank number.

Classes Folder

The *Classes* folder contains all of the project classes and their respective members. Classes are compound tags consisting of user-defined data-type structures or *tag types* (*Integer*, *Real*, *Boolean*, and *String*). Classes allow for high-level encapsulation in the project database. A class-type tag provides a set of values for its members.

To define a class you must define the *members* and their types. Class members are variables that hold values for an object with particular characteristics. Thus, the defining a class can be very useful for projects with a repeating group of variables.

**Note:** When you create a class folder, a *Class* icon displays in the *Tag List* subfolder located in the *Project Tags* folder.

To access the members of a class, use the following syntax with a period ( . ) as the separator: *TagName.MemberName*. For example: `tk.LEV` or `tk.TMP`.

If the *Tank* tag is an *array*, you use the following syntax:

```
ArrayTagName[ArrayIndex].MemberName
```

For example: `Tank[1].Level` or `Tank[n].Temperature`

A class-type tag contains a set of values (rather than a single value) associated with the class. You create class-type tags by grouping simple tags, which become the members. The maximum number of members for any class depends on the product specification. Class members can hold standard *Integer*, *Real*, *Boolean*, and *String* values, as mentioned previously.

To create a new class, use one of the following methods to open the *Insert Class* dialog:

- On the Insert tab of the ribbon, in the Global group, click *Class*;
- Right-click on the *Classes* folder, the *Members List* sub-folder, or the *Datasheet View* icon in the *Classes* folder; or
• Create a new class tag in the *Project Tags* folder.

When the *Insert Class* dialog displays, enter a class name in the **Name** field, and then click **OK** to close the dialog.

![Insert Class dialog](image)

**Note:** You must type a unique class name. You cannot create two classes with the same name. In addition, you cannot configure shared tags and system tags as classes.

IWS saves class folders in the *Tag List* subfolder (located in the *Project Tags* folder). You can edit the classes in this folder.

When the *Class* datasheet displays, you can use it to create, modify, or delete any class members and their viewable properties. (You cannot edit classes from the *Tag Property* dialog.)

![Class datasheet](image)

**Note:** The Classes folder can contain up to 16,384 classes and up to 4,096 members per class. This is a technical limitation of the tags database, not a licensing restriction; unused classes do not count against the total number of tags used.

When a new tag is created with a class type, however, each class member counts as a tag used because each member holds a value. (For example, if you create a class with 5 members and then create 5 tags with that class type, then you have a total of 25 tags used.) The total number of tags used cannot exceed the number of tags supported by the project’s target system / runtime license.

To edit a class member or property, you can right-click on the item and use standard Windows commands to cut (Ctrl+X), copy (Ctrl+C), or paste (Ctrl+V). You can also undo (Ctrl+Z) the last modification to a field.

You also edit member properties as follows:

- **Name** field: Type a name for the member or member property. The first character must be a letter and you can use up to 255 characters in the name.
- **Type** combo-box: Select a member type (*Boolean*, *Integer*, *Real*, or *String*).
- **Description** field: Type a description of the member property for documentation purposes.

**Note:** Members of a class cannot be of another class type.

Also, you must create a unique class name. You cannot reuse the name of an existing class. However, you can create members with the same name in different classes.

To delete a class and all its members, right-click on a class folder and select delete. IWS disables the delete option if you are running any runtime tasks. In addition, you cannot delete a class if it is associated with any tag.
About tag properties

Tag properties are metadata associated with each tag in the project database.

Most of the time, you may think of tags as simple program variables that store values, because that is how you generally use them in your project. Each tag, however, is in fact a complex data structure that can be handled in different ways during run time, depending on how the tag properties are configured.

In addition to standard metadata like array size, data type, description, and scope, the tag properties are used to configure alarm conditions, get tag quality, convert between different units of measurement, access individual bits, retain values through project restarts, save historical data, and so on.

Many tag properties can be viewed and edited directly in the Project Tags datasheet. In fact, every column after Name in the datasheet is another tag property, and while the datasheet shows the most common properties by default, you can configure it to show even more. To do this, right-click anywhere in the datasheet, and then click the additional tag properties on the shortcut menu. The selected properties are added as columns in the datasheet.

Other tag properties can be configured using the Tag Properties dialog. To access this dialog, select the tag in the datasheet, and then on the Home tab of the ribbon, in the Tags group, click Properties.

Finally, you can get and set many tag properties during run time just as you would get and set the values of the tags themselves, using either of the following methods:
• Call the functions `TagsDBGetTagProperty` and `TagsDBSetTagProperty` in any scripting interface; or
• Use the tag field syntax (e.g., `tagname->property`) wherever you would normally specify a project tag.

**Tag Properties: Parameters**

Each tag type has the following properties:

- **Parameters**
- **Alarms Properties**
- **History Properties**

To configure these properties for a tag, click **Properties** on the Home tab of the ribbon. A *Tag Properties* dialog displays, similiar to the following:

![Tag Properties dialog for Boolean tag](image)

Use the parameters on this dialog to configure the different tag properties. Instructions for configuring each type of tag follows.

**String-Type Tag Parameters**

From the *Tag Properties* dialog, specify string-type parameter properties as follows:

- **Retentive Value** checkbox: Click (check) to save the tag value continuously in case the system unexpectedly shuts down, so that it restarts from the last saved value.
- **Retentive Parameters** checkbox: Click (check) to save runtime changes in the tag’s properties. Only certain properties can be saved; for a complete list of which can and cannot, see List of tag properties on page 140.

⚠️ **Caution:** Enabling either of the Retentive options for tags that frequently change values can cause heavy disk access, which slows performance.

- **Startup Value** field: Type a tag value for the system load. The tag assumes this value if you disable the **Retentive Value** option.
- **Unit** field: Type any string (up to 9 characters) as a brief description or reference. This tag is accessible during runtime.

⚠️ **Note:** IWS will not accept writing values that fall outside the ranges defined in the **Min** and **Max** fields. In addition, IWS generates a message in the *Output* window to indicate that the system tried to write a value outside the defined range.
Integer and Real-Type Tag Parameters

Note: For descriptions of the Retentive Value, Retentive Parameters, Startup Value, and Unit parameters, see String-Type Tag Parameters above.

From the Tag Properties dialog, specify integer and real-type parameter properties as follows:

- **Engineering Units area**
  - **Min** field: Specify a minimum value for the tag in engineering units. This tag is accessible during runtime.
  - **Max** field: Specify maximum value for the tag in engineering units. This tag is accessible during runtime.
  - **Unit** field: Type any string (up to 9 characters) as a brief description or reference of the tag. This tag is accessible during runtime.

- **Signal Conditioning area**
  - **Dead Band** checkbox: Click (check) to insert the dead band value of a tag. Dead band value is a variation around a central value of the tag, which is not recognized for alarms.
  - **Smoothing** checkbox: Click (check) to reduce the rate of change for the tag's values. Use only for integers and real tags. For example, if you select the **Smoothing** option for the **LEVEL1** tag containing the value = 50. Then in the next search, if the **LEVEL1** changes to 60, the system will store the average of 50 + 60 in the database, so the new value = 55.

Boolean-Type Tag Parameters

Note: For descriptions of these parameters, see String-Type Tag Parameters above.

See also: Using Tag Properties: Alarms, Using Tag Properties: History

Tag Properties: Alarms

Use the Tag Properties dialog to view the configured alarms for a selected tag. IWS disables this command if there are open alarm worksheets. Before using these dialogs, you should have already created the alarm groups.

ALARM TYPES:
- **HiHi**: A Very High alarm is present.
- **Hi**: A High alarm is present.
- **Lo**: A Low alarm is present.
- **LoLo**: A Very Low alarm is present.
- **Rate**: An alarm based on rate of change is present.
- **Deviation**: An alarm based on deviation from a given set point is present. Example:
  - If SetPoint = 50, Deviation + = 5, Deviation − = 5, and Deviation Dead Band = 0.5;
  - IWS generates an alarm when the temp ≥ 55 or temp ≤ 45; and
  - A return to the normal occurs when temp ≤ 54.5 or temp ≥ 45.5.

ALARM LIMITS:
- **HiHiLimit**: When creating Very High alarms in the Tag Properties dialog, use this field to specify the limits. You can access this field during runtime and use it during modifications on the fly.
- **HiLimit**: When creating High alarms in the Tag Properties dialog, use this field to specify the limits. You can access this field during runtime and use it during modifications on the fly.
- **LoLimit**: When creating Low alarms in the Tag Properties dialog, use this field to specify the limits. You can access this field during runtime and use it during modifications on the fly.
• **LoLoLimit**: When creating Very Low alarms in the *Tag Properties* dialog, use this field to specify the limits. You can access this field during runtime and use it during modifications on the fly.

• **DevSetpoint**: Reference point for a tag value deviation that triggers an alarm. Define the alarm message in the *Tag Properties* dialog or on an *Alarm* worksheet. You can access this field during runtime.

• **Dev+Limit**: Limit deviation to a value higher than the **DevSetpoint** in a tag value that triggers an alarm. Define the alarm message in the *Tag Properties* dialog or on an *Alarm* worksheet. You can access this field during runtime.

• **Dev+Limit**: Limit deviation to a value lower than the **DevSetpoint** in a tag value that triggers an alarm. Define the alarm message in the *Tag Properties* dialog or on an *Alarm* worksheet. You can access this field during runtime.

• **RateLimit**: Limit of rate variation in a tag value that triggers an alarm. Define the alarm message in the *Tag Properties* dialog or on an *Alarm* worksheet. You can access this field during runtime.

**Alarms for Integer and Real Type Tags**

From the *Tag Properties* dialog, specify an alarm for integer and real-type tags as follows:

• **Alarms Enabled** checkbox: Click (check) to enable checking according to configuration.

• **Remote Ack tag** field: Type a tag to acknowledge this alarm.

• **Dead Band Value** field: Type a value of the filter for alarms normalization. For example: if the TEMP1 tag is configured with an alarm with Hi Limit = 90 and Dead Band = 5, then IWS generates an alarm when TEMP1 ≥ 90. The return to normal occurs when TEMP1 ≤ 85.

• **Translation Enabled** checkbox: Click (check) to enable the translation of messages defined on the *Options* tab in the *Project Settings* dialog. For additional information about translations, see *Translation Tools*.

> **Note:** IWS saves Alarm messages with the **Translation Enabled** attribute enabled in a file called *Alarm.TXT* located in your project’s \**DATABASE\* directory.

• **HiHi (HiHiLimit)** checkbox: Click (check) to indicate a Very High alarm is present. Accessible during runtime.

• **Hi (HiLimit)** checkbox: Click (check) to indicate a High alarm is present. Accessible during runtime.

• **Lo (LoLimit)** checkbox: Click (check) to indicate a Low alarm is present. Accessible during runtime.

• **LoLo (LoLoLimit)** checkbox: Click (check) to indicate a Very Low alarm is present. Accessible during runtime.

• **Rate (RateLimit)** checkbox: Click (check) to indicate a Rate alarm is present. Accessible during runtime.

• **Deviation+** checkbox: Click (check) to indicate a Deviation alarm is present. Accessible during runtime.

• **Deviation-** checkbox: Click (check) to indicate a Deviation alarm is present. Accessible during runtime.

• **Deviation SetPoint** field: Specify a reference point for the deviation. Accessible during runtime.

• **Deviation Dead Band** field: Specify a reference value for the deviation.
Alarms for Boolean-Type Tags

Use the Alarms-Bool Type tab on the Tag Properties dialog to specify alarm properties for Boolean-type tags.

Configure the parameters as follows:

- **Alarms Enabled** checkbox: Click (check) to enable checking according to configuration.

- **Remote Ack tag** field: Type a tag to enable remote alarm acknowledgment, which occurs when the tag values change.

- **Translation Enabled** checkbox: Click (check) to enable the translation of messages defined on the Options tab in the Project Settings dialog. For additional information about translations, see Translation Tools.

  **Note:** IWS saves Alarm messages with the Translation Enabled attribute enabled in a file called Alarm.TXT located in your project's DATABASE directory.

- **Off** checkbox: Click (enable) to generate an alarm message always when the tag value is zero.

- **On** checkbox: Click (enable) to generate an alarm message always when the tag value is one.

- **Changed** checkbox: Click (enable) to generate an alarm message always when the tag value changes.

- **Text value** pane: Use the Off, On, and Ack fields to configure mnemonics (for example, Closed or Open) for the Off, On, and Ack alarm states (Boolean tags only). During runtime, the Alarm Control object will display this text in the Value column of the alarm message. You can also access this text using the tag fields.

  **Note:** If you do not configure a mnemonic, the Alarm Control object displays the tag value (0 or 1) in the Value column.

**Tag Properties: History**

Use the Tag Properties dialog to view the history for a selected tag. IWS disables this command if there are open trend worksheets. Before using this dialog, you should have already created the trend groups.

**Note:** History does not support string-type tags. See Recipes folder for information about storing string values.

**Integer and Real Type History**

From the Tag Properties dialog, enable history for integer and real-type tags as follows:

- **History Enabled** checkbox: Click (check) to enable storage of the selected tag value samples.

- **Group Number** field: Specify the group number to which this tag is associated.
• **Log Dead Band** field: Specify a value sample taken when the variation value is equal to, or greater than, the Log Dead Band.

**Boolean Type History**
From the *Tag Properties* dialog, enable history for Boolean-type tags as follows:

• **History Enabled** checkbox: Click (check) to enable storage of the selected tag value samples.

• **Group Number** field: Specify the group number to which this tag is associated.

**List of tag properties**
This is a list of tag properties that you can access using the tag field syntax.
To get the value of a tag property during run time, use the following syntax (without spaces) anywhere that you would normally specify a tag:

\[\text{tagname->property}\]

You can access the following tag properties during run time:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>R or R/W</th>
<th>Data Type</th>
<th>Available on Data Type...</th>
<th>Retain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the tag, as configured in the Project Tags database.</td>
<td>R</td>
<td>String, up to 32 chars</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>MemberName</td>
<td>The name of the class member, in a properly configured <em>Class</em>. NOTE: The syntax must be: [Class.Member-&gt;MemberName]</td>
<td>R</td>
<td>String, up to 32 chars</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: [Tank.Lvl-&gt;MemberName] = Lvl</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>Array Size. If the tag is not an array tag, it returns the value 0</td>
<td>R</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Index</td>
<td>The index number of an element in an Array. (An Array is any Tag of size greater than 0.) NOTE: The syntax must be: [Tag[Index]-&gt;Index]</td>
<td>R</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Example: [Tag[1]-&gt;Index] = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>The description of the tag, configured in the Tags datasheet.</td>
<td>R</td>
<td>String</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Quality</td>
<td>Tag quality (192=GOOD; 0=BAD). The project updates this field every time the tag receives the result of an expression or a value from a communication task (such as driver or OPC). If the expression is invalid (such as, division by zero) or if there is a reading communication error associated with the tag, then the project sets the quality to BAD.</td>
<td>R</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>Time and date when the value of the tag last changed.</td>
<td>R</td>
<td>String</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Blocked</td>
<td>This property can have two values:</td>
<td>R/W</td>
<td>Boolean</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>R or R/W</td>
<td>Data Type</td>
<td>Available on Data Type...</td>
<td>Retain</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>---------------------------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bool</td>
<td>Int</td>
<td>Real</td>
</tr>
<tr>
<td>Tags and the Project Database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R or R/W</td>
<td>Data Type</td>
<td>Available on Data Type...</td>
<td>Retain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bool</td>
<td>Int</td>
<td>Real</td>
<td>Str</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 0: The tag is blocked and all runtime tasks will ignore it. It is effectively removed from the project database.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 1: The tag is unblocked and all runtime tasks can access it normally.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This is useful when you want to dynamically disable all actions associated with a specific tag. Even when a tag is blocked, however, it still counts towards the total number of tags used for licensing purposes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit</td>
<td>A brief description (up to 9 characters) of the Engineering Unit (i.e., the unit of measurement) for the Tag value. For example, Kg, BTU, psi.</td>
<td>R/W</td>
<td>String, up to 9 chars</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Max</td>
<td>The maximum value that can be written to the tag during run time.</td>
<td>R/W</td>
<td>Real</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Min</td>
<td>The minimum value that can be written to the tag during run time</td>
<td>R/W</td>
<td>Real</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>B0 … B31</td>
<td>Value (0 or 1) of any of the 32 bits (b0, b1, b2, ... b31) of an Integer tag. (B0: LSB B31: MSB)</td>
<td>R/W</td>
<td>Boolean</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>DisplayValue</td>
<td>A converted Tag value that is only displayed on-screen: <strong>DisplayValue</strong> = (Value / UnitDiv) + UnitAdd</td>
<td>R/W</td>
<td>Real</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>DisplayUnit</td>
<td>A brief description (up to 9 characters) of the Engineering Unit for DisplayValue. <strong>NOTE:</strong> This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.</td>
<td>R</td>
<td>String, up to 9 chars</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>UnitDiv</td>
<td>Number by which the Tag value is divided to get DisplayValue. To perform no division, UnitDiv should be 1. <strong>NOTE:</strong> This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.</td>
<td>R</td>
<td>Real</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>UnitAdd</td>
<td>Number added to the Tag value to get DisplayValue. To perform no addition, UnitAdd should be 0. <strong>NOTE:</strong> This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.</td>
<td>R</td>
<td>Real</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>DisplayMax</td>
<td>The maximum value that can be input to DisplayValue during run time: <strong>DisplayMax</strong> = (Max / UnitDiv) + UnitAdd</td>
<td>R/W</td>
<td>Real</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>R or R/W</td>
<td>Data Type</td>
<td>Available on Data Type...</td>
<td>Retain</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------</td>
<td>-----------</td>
<td>---------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>DisplayMax</td>
<td>If DisplayMax is changed during run time, then Max is also changed as follows:</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max = (DisplayMax – UnitAdd) * UnitDiv</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DisplayMin</td>
<td>The minimum value that can be input to DisplayValue during run time:</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DisplayMin = (Min / UnitDiv) + UnitAdd</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If DisplayMin is changed during run time, then Min is also changed as follows:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Min = (DisplayMin – UnitAdd) * UnitDiv</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiHiLimit</td>
<td>Limit value for the HiHi alarm.</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N N</td>
<td></td>
</tr>
<tr>
<td>HiLimit</td>
<td>Limit value for the Hi alarm.</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N N</td>
<td></td>
</tr>
<tr>
<td>LoLimit</td>
<td>Limit value for the Lo alarm.</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N Y</td>
<td></td>
</tr>
<tr>
<td>LoLoLimit</td>
<td>Limit value for the LoLo alarm.</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N Y</td>
<td></td>
</tr>
<tr>
<td>RateLimit</td>
<td>Limit value for the Rate alarm.</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N Y</td>
<td></td>
</tr>
<tr>
<td>DevSetpoint</td>
<td>Setpoint value for Deviation alarms.</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N n/a</td>
<td></td>
</tr>
<tr>
<td>DevPLimit</td>
<td>Limit value for the Deviation+ alarm.</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N Y</td>
<td></td>
</tr>
<tr>
<td>DevMLimit</td>
<td>Limit value for the Deviation- alarm.</td>
<td>R/W</td>
<td>Real</td>
<td>N Y Y N Y</td>
<td></td>
</tr>
<tr>
<td>HiHi</td>
<td>If 0, the HiHi alarm is not active. If 1, the HiHi alarm is active.</td>
<td>R</td>
<td>Boolean</td>
<td>Y Y Y N n/a</td>
<td></td>
</tr>
<tr>
<td>Hi</td>
<td>If 0, the Hi alarm is not active. If 1, the Hi alarm is active.</td>
<td>R</td>
<td>Boolean</td>
<td>Y Y Y N n/a</td>
<td></td>
</tr>
<tr>
<td>Lo</td>
<td>If 0, the Lo alarm is not active. If 1, the Lo alarm is active.</td>
<td>R</td>
<td>Boolean</td>
<td>Y Y Y N n/a</td>
<td></td>
</tr>
<tr>
<td>LoLo</td>
<td>If 0, the LoLo alarm is not active. If 1, the LoLo alarm is active.</td>
<td>R</td>
<td>Boolean</td>
<td>Y Y Y N n/a</td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>If 0, the Rate alarm is not active. If 1, the Rate alarm is active.</td>
<td>R</td>
<td>Boolean</td>
<td>Y Y Y N n/a</td>
<td></td>
</tr>
<tr>
<td>DevP</td>
<td>If 0, the Deviation+ alarm is not active. If 1, the DevP alarm is active.</td>
<td>R</td>
<td>Boolean</td>
<td>N Y Y N n/a</td>
<td></td>
</tr>
<tr>
<td>DevM</td>
<td>If 0, the Deviation- alarm is not active. If 1, the DevM alarm is active.</td>
<td>R</td>
<td>Boolean</td>
<td>N Y Y N n/a</td>
<td></td>
</tr>
<tr>
<td>AlrStatus</td>
<td>Integer value with the status of the current active alarms associated to the</td>
<td>R</td>
<td>Integer</td>
<td>Y Y Y N N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tag. Each bit of this integer value indicates a specific status:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Bit 0 (LSB): HiHi Alarm active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Bit 1: Hi Alarm active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Bit 2: Lo Alarm active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Bit 3: LoLo Alarm active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Bit 4: Rate Alarm active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Bit 5: Deviation+ Alarm active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Bit 6: Deviation- Alarm active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Examples: If Tag!&gt;AlrStatus returns the value 2, it means that &quot;Hi&quot; alarm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>is active. If it returns the value 3, it means that the &quot;HiHi&quot; and the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;Hi&quot; alarm are active simultaneously.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Tags and the Project Database

#### Available on Data Type:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>R or R/W</th>
<th>Data Type</th>
<th>Available on Data Type...</th>
<th>Retain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bool</td>
<td>Int</td>
<td>Real</td>
</tr>
</tbody>
</table>

| Ack      | R | Y | Y | Y | N | N |
| UnAck    | R/W | Y | Y | Y | N | N |
| AlrAckValue | R/W | String, up to 32 chars | Y | N | N | Y |
| AlrOffValue | R/W | String, up to 32 chars | Y | N | N | Y |
| AlrOnValue | R/W | String, up to 32 chars | Y | N | N | Y |
| AlrDisable | R/W | Boolean | Y | Y | Y | N | N |

**Note:**
- If a property is marked "n/a" with regards to being retentive, it's because the property is inherent in the tag definition (e.g., Name, Size) or the value of the property is continuously derived during...
run time (e.g., alarm activation, DisplayValue). To enable retention for a tag, select the **Retentive Parameters** option in the **Tag Properties** dialog.

- If the project attempts to write a value outside of the range specified in the **Min** and **Max** properties, the Tags Database will not accept the new value and a warning message is written in the **Output** window. If both **Min** and **Max** properties are configured with the value 0 (zero), it means that any value applied to the tag type will be written to the tag.
- You cannot use tag properties (such as Bit fields) to configure **Alarm** or **Trend** worksheets.
- Although you can apply tag properties to **System Tags**, those properties will not persist when you download your project to a CE device.

**Change how out-of-range tag values are handled**

A project tag will occasionally receive a value that is outside of its normal range. You can change how these out-of-range values are handled during run time.

**Note:** This topic applies only to Integer and Real-type tags, because they are the only tags for which the **Min** and **Max** properties have meaning. Boolean-type tags can only have values of 0 and 1, and String-type tags do not have numeric values at all.

A project tag has a range of possible values, and that range is determined by the tag's **Min** and **Max** properties. By default, if the tag receives a new value that is outside of its range — that is, if the value is less than the minimum or greater than the maximum — then the received value is ignored, the existing value is retained, the tag quality is set to **UNCERTAIN**, and a warning message is sent to the **Output** window.

In some situations, however, it is useful to change how these out-of-range values are handled — in particular, you may need to emulate how out-of-range values are handled by another vendor's hardware or software, such as Wonderware InTouch.

To change this behavior, you must manually edit your project file (e.g., **projectname.app**) to add or modify the following settings:

```plaintext
[Options]
WriteOutOfRange={FALSE|TRUE}
CapOutOfRange={FALSE|TRUE}
```

**WriteOutOfRange**

By default, this setting is FALSE. When this setting is TRUE, out-of-range tag values are accepted as good (i.e., the tag quality is not set to **UNCERTAIN**).

**CapOutOfRange**

By default, this setting is FALSE. When this setting is TRUE, out-of-range tag values are capped at the tag's minimum and maximum.

**Note:** If a setting is not present in the project file, then its default value is used.

To change how out-of-range tag values are handled:

1. Locate your project file, which is typically at: `[...]\Documents\InduSoft Web Studio v7.1 Projects \projectname\projectname.app`
2. Open the project file with a standard text editor, such as Notepad, and then add or modify the settings **WriteOutOfRange** and **CapOutOfRange**.

The following table shows how the two settings work together:

<table>
<thead>
<tr>
<th>Settings</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Options] WriteOutOfRange=FALSE</td>
<td>The received value is ignored, the existing value is retained, the tag quality is set to <strong>UNCERTAIN</strong>, and a warning message is sent to the <strong>Output</strong> window. This is the default behavior.</td>
</tr>
</tbody>
</table>
### Settings

<table>
<thead>
<tr>
<th>CapOutOfRange=FALSE</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the received value is lower than Min, then the tag value is set to Min, and if the received value is greater than Max, then the tag value is set to Max. The tag quality is set to UNCERTAIN.</td>
</tr>
</tbody>
</table>

### [Options]

<table>
<thead>
<tr>
<th>WriteOutOfRange=FALSE</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapOutOfRange=TRUE</td>
<td>Min and Max are ignored when setting tag values; the tag value is set to whatever value it receives, and the tag quality is set to GOOD. Min and Max are still applied to linear conversions in device communication.</td>
</tr>
</tbody>
</table>

### [Options]

<table>
<thead>
<tr>
<th>WriteOutOfRange=TRUE</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapOutOfRange=FALSE</td>
<td>If the received value is lower than Min, then the tag value is set to Min, and if the received value is greater than Max, then the tag value is set to Max. The tag quality is set to GOOD.</td>
</tr>
</tbody>
</table>

### [Options]

<table>
<thead>
<tr>
<th>WriteOutOfRange=TRUE</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapOutOfRange=TRUE</td>
<td>If the received value is lower than Min, then the tag value is set to Min, and if the received value is greater than Max, then the tag value is set to Max. The tag quality is set to GOOD.</td>
</tr>
</tbody>
</table>

3. Save and close the project file.
Using Tags in Your Project

Once you have added a tag to the project database, you can use that tag in your project by associating it to objects on a screen.

The basic process for associating tag to screen objects consists of the following steps:

1. In the project screen, select the object to which you want to apply the tag.
2. Click one of the buttons in the Animations group to apply that animation to the object.
3. Double-click on the object to open its Object Properties dialog.
4. Locate the Tag text box for that property and type the tag name into the field.
   
   Tag text box names and locations will vary, depending on the type of property you are using. For example:

   ![Image of Tag Properties Dialog]

   **Applying Tags to an Object**

   Comprehensive instructions for applying tags to screen objects are provided throughout the documentation where appropriate.
Deleting a tag from the project database

Delete a tag that is no longer in use by deleting its line in the Project Tags or Shared Database datasheet.

Before you delete a tag, we strongly recommend that you use the Cross-Reference tool to make sure the tag is not being used anywhere in your project. (If you delete a tag that is still being used, then you will not be able to verify and run your project.) Fix any screens or worksheets where the tag is being used before you proceed.

Note: This task applies to both the Project Tags and Shared Database datasheets.

To delete a tag:

1. Stop the project if it is running.
2. Open the datasheet for editing.
3. In the datasheet, find the line for the tag you want to delete.
4. Right-click the line, and then select Delete Line from the shortcut menu.
   If the option is disabled, then you may need to clear any sorting or filtering that you previously applied to the datasheet.
   An alert dialog is displayed asking you to confirm the action.
5. Click Yes.
   The line is deleted from the datasheet.
6. Save and close the datasheet.
Sort or filter the rows in a worksheet

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item. Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications.

Please note that you can sort or filter rows only in the following types of worksheets:

- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.

Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.

![Alarm worksheet rows in their original order](image1)

![Alarm worksheet rows sorted by Type](image2)
Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.

**Alarm worksheet rows filtered where Tag Name is "Tag3"**

**Alarm worksheet rows filtered where Type is "Lo"**

**Tip:** You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order.
   
   The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   **Note:** You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.

3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either Tab or Return.
   
   You may include * and ? as wildcard characters in your string:

   - * matches any number of characters, including none. For example, Tag* would match Tag, Tag3, Tag34567, TagA, and Tag_TEMP.

   - ? matches exactly one character. For example, Tag? matches Tag3 and TagA, while Tag?????? matches Tag34567 and Tag_TEMP.

   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.

4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either Tab or Return.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.
Using the Tags Toolbar

The Tags toolbar provides a text box and several tools (shortcuts) that enable you to create, locate, and access different tags, functions, and tag properties.

Global Replace Tool

When clicking on the Global Replace tool from the Tag Properties Toolbar, the following window displays:

From the Global Replace dialog, you can replace any tag(s) from all documents (screens and worksheets) of the whole project. You can edit both the From and the To column.

When replacing composed tags (array size > 0 and/or Type = Class), you can configure a specific array position (for example, TagA[1]) or class member (for example, TagB.MemberX) or both (for example, TagC[3].MemberY). If you configure only the Main Tag Name (for example, TagC) in the From column, all tags from this main tag will be modified for the tag configured in the To column.

If an invalid replacement is configured (for example, replace the Main Tag tag from a class type tag for a simple tag (not a class tag), the OK button will be disabled. When the OK button is pressed, the tags configured on the Global Replace dialog will be replaced in the order that they were configured on the dialog interface.

Note: You must close all documents (screens and worksheets) before executing this command.

When changing the tag name on the Project Tags database worksheet, IWS will ask you if you intend to replace this tag through the whole project.

The Replace option will be created in the Edit menu. By using this option, the Global Replace dialog is prompted, however, the changes are applied only the current screen or worksheet in focus.

Replacing project tags in a document or screen object

To replace all occurrences of a tag in the current document, do one of the following:

• On the Home tab of the ribbon, in the Tags group, click Replace; or
• On the Graphics tab of the ribbon, in the Editing group, click Replace.

To replace all occurrences of a tag in a screen object, double-click the object to open its Object Properties dialog and then click Replace.

All of these methods will open the Replace dialog, which is described below.
You can replace one or more tags by clicking the **Whole Tag Name** tab. Current tags used are displayed. The original tag names are shown in the **From** column on the left, and you can enter your new tag names in the **To** column on the right.

![Whole Tag Name tab](image)

Note that this does not rename or delete any tag — it only replaces the tags used in the object with other tags from the database.

You can also replace one or more strings (e.g., button captions, descriptive text) by clicking the **String Value** tab.

![String Value tab](image)

When you are done, click **OK**.

**Removing unused tags from the project database**

The **Remove unused tags** tool is used to scan the project database for unused tags, which you can then select and remove.

"Unused tags" are tags that you have defined in the project database but have not used in any screen or task worksheet. Since your project has a limited number of available tags (as determined by your product/license type), you may want to remove some or all of these unused tags to decrease your project’s tag count.

1. Save and close all open project screens and worksheets.
2. On the **Home** tab of the ribbon, in the **Tags** group, click **Remove unused tags**.
   
The development application automatically verifies your project, and if it finds any unused tags, then it lists them in the **Remove Unused Tags** dialog.
Note: The listed tags may include some that are accessed during runtime using indirect syntax (e.g., \texttt{GetTagValue(TagName)} or \texttt{@TagName}, where the value of \texttt{TagName} is the name of an unused tag).

3. Determine which tags you want to remove, if any.
   - If you want to remove all of the listed tags, simply click Remove.
   - If you want to keep some of the listed tags, clear the Remove check boxes on the left for those tags, and then click Remove.
   - Click Check all or Uncheck all to select or clear, respectively, all of the Remove check boxes on the left.
   - If you do not want to remove any of the listed tags, click Close.

   The development application removes the selected tags and then asks if you want to verify the project again.

4. Click Yes to verify the project again.

Reset Tags Database

Select Reset Tags Database to "reload" the tags database on the local station. This command affects all tags stored in the Project Tags folder. This option is useful for resetting the project tags and restoring the values they had when the project was loaded for the first time. When you stop the project but leave the development environment open, the tags are not reset by default when the project is run again. Therefore, you can execute this command to reset them before the project runs again.

When this command is executed, the Startup Value configured for each tag (Tags Properties dialog) is written to the respective tag. If you did not configure any Startup Value for a numeric tag (Boolean, Integer or Real), the value 0 (zero) is written to the tag. If you did not configure any Startup Value for a string tag, the empty value ("") is written to the tag.

This command is disabled (in gray) if there is at least one runtime task running on the local station. You must close all runtime tasks (Stop on the Home tab of the ribbon) before this command can be executed.

Note: The tags stored in the System Tags folder and in the Shared Tags folder (if any) are not affected by this command.
Tip: If you want to reset the project tags automatically whenever you run the project (Run on the Home tab of the ribbon), you can check the option Reset Tags Database when starting project on the Preferences tab of the Project Settings dialog.

Tagname Text Box

Type a name into the Tagname text box to create a new tag for your project. The Cross Reference and Tag Properties tools will reference this tag name for their actions.

Object Finder Tool

Click the Object Finder tool to open the Object Finder dialog, which lists all Tags and Functions currently configured for the project.

To select an existing tag/function, double-click on the tag/function name, and then click OK to close the box. The selected name displays in the Tagname text box.

- To select a specific array index, click the Index button after specifying the array tag name.
- To select a specific member name, click the Member button after specifying the class tag name.
- To create a new tag, click the New button.
- When the New Tag dialog displays, enter the following information, then click OK to close the box:
  - Name
  - Array Size
  - Type (Boolean, Integer, Real, String, Class:Control, Class:msgonline, or Class:Alr)
  - Description
  - Scope (local or server)
**Cross Reference Tool**

Click the Cross Reference tool to search all project screens and worksheets for the tag noted in the Tagname text box. This function writes a log, detailing all the occurrences of the tag, to the XRef tab in the Output window. For example, the results of searching for a BlinkFast tag are as follows:

![XRef Results]

**Tag Properties Tool**

Click the Tag Properties tool to configure parameters for each tag. The Tag Properties dialog displays so you can specify these parameters. (For more information about specifying tag properties, see Understanding Tag Properties and Parameters.)

![Sample Tag Properties dialog]

**Note:** The Tag Properties dialog that displays for Boolean, Integer, Real, and String tags will differ in content.
Import Wizard

The Import Wizard is a powerful tool that reduces engineering time during project development. Using the Import Wizard, you can import tags from different data sources directly to the project tags database. Depending on the data source, you can import not only the tag names, but also the communication interface (the link between the tags and the PLC addresses).

When you click **Import Wizard** on the Home tab of the ribbon, an **Import Database Wizard** dialog displays to step you through the process of importing tags. There are three steps for importing tags from these data source types:

- InduSoft Web Studio Project Database
- OPC Server Database
- CSV Database
- ODBC Database
- RSLogix™ 5000 CSV Database
- PanelBuilder32™ Database
- PanelMate Plus™ Database
- OMRON™ CX Programmer Database
- Studio XML Screen
- TwinCAT™ PLC Database

**Step1: Select the Source Type**

Click the data Source Type, which is where the tags are being imported from. Click **Next**.
Continue to the appropriate section for the instructions you need to complete the import database procedure:

- **Importing from Other Project Databases** on page 158
- **Importing from OPC Server Databases** on page 160
- **Importing from CSV Databases** on page 161
- **Importing from ODBC Databases** on page 162
- **Importing from RSLogix 5000 CSV Databases** on page 162
- **Importing from PanelBuilder32 Databases** on page 163
• Import PanelMate programs on page 164
• Importing from TwinCAT PLC Databases on page 165
• Importing from OMRON CX Programmer Databases on page 166

Step 2: Configure the Source Type Settings

![Import OPC Server dialog]

Most of the settings in the second window depend on the data Source Type selected in the first step. The screenshot above is an example of one data Source Type (OPC Server Database). The settings that are common for any data Source Type are described below:

- **Options** box: Select *Do not import duplicated tags* if you do not want imported tags to overwrite tags with the same name that already exist in the Tags Database of the current project. Select *Replace duplicates with tags imported* to overwrite tags in the Tags Database with imported tags of the same name.

- **Use Prefix**: Check to specify a prefix (up to 4 characters) to be concatenated to the name of the imported tags. It is useful to use a prefix to differentiate the imported tags from the tags created manually.

**Note**: The other settings vary according to the data source selected in the first step, and they are described in the specific sections for each data Source Type.

After configuring the settings in this dialog, click Next.
Step 3: Filter the tags

The screenshot above is an example of one data Source Type (OPC Server Database). The fields and settings that are common for all data Source Types include the following:

- **Grid**: Displays the list of tags found on the data source.
  - **Checkbox**: Check to import the tag from the data source to the Tags Database of the current project.
  - **TagName**: Name of the tag
  - **Size**: Array size of the tag
  - **Type**: Data type of the tag (Boolean, Integer, Real, String or Class:<ClassName>)
  - **Description**: Description of the tag
- **Check** button: Click to select/import all tags in the grid
- **Uncheck** button: Click to uncheck all tags in the grid
- **Filter** button: Click to filter the tags. The Filter dialog will display, allowing you to specify a mask for each column in the grid. Wild cards (*) and (?) can be used to filter data.
- **Clear Filter** button: Click to reset the filter.
- **Import Filtered Tags Only** checkbox: Check this option to import only the tags that are visible in the grid (filtered).
- **Status** box: Displays a message describing the status of the tag currently selected in the grid. This information is especially useful to indicate why a tag cannot be imported.
- **Legend** box: Describes the meaning of the colors that represent tag status:
  - (Red) **Error**: Tag cannot be imported because it is not supported by IWS. See the **Status** box for a detailed description of the error.
  - (Blue) **Tag will be imported**: Tag will be imported after you click the Finish button.
  - (Gray) **Tag can be imported**: Tag can be imported but it has not been checked.
- **Database size** box: Displays summary information regarding the current Import Wizard:
  - **Current**: Indicates the number of tags configured in the Project Tags database of the current project
  - **Importing**: Indicates the number of tags selected to be imported
  - **Replacing**: Indicates the number of tags configured in the Project Tags database of the current project that will be replaced by an imported tag with the same name.

After selecting the tags to import, click the Finish button, or click Cancel to abort the operation.
Note: The other settings vary according to the data source selected in the first step, and they are described in the specific sections for each data Source Type (see below).

**Importing from Other Project Databases**

This wizard allows you to import the interfaces (tags and worksheets) of other IWS projects. When you import only the tags (rather than the whole project) from a remote computer, the TCP/IP Client worksheet can be automatically created to link the tags between both stations (the local and the remote), and to share the value of these tags between both stations during runtime.

**Import Project Database Wizard dialog**

- **Import tags only**: When this option is selected, the tags from the other project will be imported to the current project. The other interfaces of the project (worksheets) will not be imported.

- **Import the whole project**: When this option is selected, the following interfaces from the other project will be imported to the current project:
  - Tags Database
  - Global Procedures
  - Screens
  - Screen Groups
  - Web Pages
  - Alarms
  - Trends
  - Recipes
  - Reports
  - ODBC
  - Math
  - Scripts
  - Scheduler
  - Drivers
  - OPC
  - TCP/IP
• **DDE**

This option is useful for merging projects and importing template projects.

---

**Note:** When you select the option to **Import the whole project**, the following worksheets will always be imported, regardless of existing worksheets with the same number in the current project: ODBC, Math, Script, Scheduler, Drivers, OPC, TCP/IP and DDE. If there are worksheets with the same number in the current project, worksheets imported from the other project will be inserted as additional worksheets in the current project, and the number of each worksheet will be automatically increased to avoid replacing files on the current project.

---

• **Do not import duplicated:** When this option is selected, the following interfaces are not imported in case there is already an equivalent interface in the current project:
  - Tags Database (tags with the same name will not be imported)
  - Global Procedures (the global procedures will not be imported at all)
  - Screens (screens with the same name will not be imported)
  - Screen Groups (screen groups with the same name will not be imported)
  - Web Pages (Web pages with the same name will not be imported)
  - Alarms (alarms assigned to tags with the same name will not be imported)
  - Trend (trend logs assigned to tags with the same name will not be imported)
  - Recipes (recipes with the same name will not be imported)
  - Reports (reports with the same name will not be imported)
  - Script (the startup script will not be imported at all)

• **Application:** Click **Browse** and select the `project_name.APP` file that has the tags you want to import.

• **Generate TCP/IP Client worksheet:** If you want your project to share tags with another IWS project running on a remote server, select this option and enter the IP address of that server. IWS will automatically configure the TCP/IP Client worksheet to exchange data with the remote project.

• **Use prefix:** Select this option to specify a prefix (up to 4 characters) that will be prepended to the names of the imported tags. It is useful to differentiate the imported tags from the tags created manually.
**Importing from OPC Server Databases**

This wizard allows you to import tags from either a local OPC Server or a remote one. When you import tags from the OPC Server, the OPC Client worksheet is automatically created to link the tags, eliminating the need to configure the communication interface between the OPC Client from IWS and the external OPC Server.

![Import OPC Server Database Wizard](image)

**Import OPC Server Database Wizard**

- **Local/Remote**: Provide the following options:
  - **Local**: Select this option to import tags from an OPC Server installed in the local computer.
  - **Remote**: Select this option to import tags from an OPC Server installed in a remote computer. Type the IP Address (or the host name) of the remote computer where IWS is running in the **Remote** field.

- **Merge Local and Remote OPC Servers** checkbox: If you selected a Remote server, check this option to display the list of OPC Servers installed in the local computer and also in the remote computer. Uncheck this checkbox to display only the list of OPC Servers installed in the remote computer.

- **Identifier** combo-box: Displays the list of available OPC Servers.

- **Branch**: Click on the Browse button (…) to select the branch of the OPC Server from which the tags (items) will be imported. Leave this field blank if you want to import tags from all branches configured in the OPC Server.

- **Use the item path for the tagname** checkbox: Check this option to concatenate the path name to the item name when importing tags from the OPC Server. Uncheck this option to use only the item names configured in the OPC Server.

In the grid displayed in Step 3 (**Import Wizard** on the Home tab of the ribbon) for this Data Source Type, there is an additional field with the label **OPC**, which displays the name of the items from the OPC Server.

**Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.
Importing from CSV Databases

This wizard allows you to import tags from a text file in the CSV (Comma Separated Values) format, or any similar format.

**Import CSV Database Wizard**

- **File Name**: Press the Browse button to select the text file from which the tags will be imported.
- **Data Column** box: Select a number for each tag property that corresponds to its column number in the import file. For example, if the Tag, Array Size and Type are listed in the second, third and first columns in the import file, respectively, select 2 in Tag, 3 in Array Size and 1 in Type. The Tag property (tag name) is mandatory, but the other properties are optional.

  For properties that are not included in the text file, select the option **Not used**. IWS will insert defaults or leave the field blank, according to the following list:

  - **Array Size**: 0
  - **Type**: Integer
  - **Description**: <Blank>
  - **Web Data**: Local

- **Delimiters** checkbox: Select the delimiter(s) used in the text file to divide one column from another. For a CSV file, the delimiter is Comma (the default). You can select more than one delimiter at a time, and you can use the Other option to enter a custom delimiter.

**Note**: See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.
**Importing from ODBC Databases**

This wizard allows you to import tags from an external SQL Relational Database such as Microsoft Access, SQL Server, Oracle, My SQL, Sybase and others, through the ODBC interface.

![Import ODBC Database Wizard](image)

- **Select Data Source** button: Click to select the ODBC Data Source Name (DSN) linked to the database from which the tags will be imported. The DSN must have previously been created with the Data Sources (ODBC) window ([Control Panel] > [Administrative Tools] > [Data Sources (ODBC)]). After you select a DSN, the other fields in this window will be populated automatically with information from the selected database.
- **Table** combo-box: Select the table that holds the tags in the import database.
- **Tag** combo-box: Select the name of the column that holds the tags in the import database.
- **Array Size** combo-box: Select the name of the column that holds the array size for the tags in the import database.
- **Type** combo-box: Select the name of the column that holds the tag type in the import database.
- **Description** combo-box: Select the name of the column that holds the tag description in the import database.
- **Web Data** combo-box: Select the name of the column that holds the Web Data for the tags in the import database.

**Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

**Importing from RSLogix 5000 CSV Databases**

This wizard allows you to import tags from a program for ControlLogix/FlexiLogix PLC developed with RSLogix™ 5000 and exported to a CSV file. When you import tags from the RSLogix™ 5000 CSV file, the
ABCIP driver worksheet is automatically created to link the tags imported with the PLC, eliminating the need to configure the communication interface between IWS and the PLC manually.

- **PLC Options** box: Provides the following options:
  - **Use Full Scope**: Select to import the tags using the full scope configured in the PLC program.
  - **Do Not Use Scope**: Select to ignore the scope of the tags configured in the PLC program.
  - **Use Limited Scope**: Select to set the number of characters from the Scope that must be used when importing the tags from the PLC program.
  - **PLC IP Address**: Type the IP Address of the PLC. This information will be used to configure the communication driver worksheets automatically.
  - **CSV File**: Click the Browse button to select the CSV file exported by the RSLogix™ 5000 with the list of tags configured in the PLC program.
  - **L5K File**: Click the Browse button to select the L5K file saved by the RSLogix™ 5000 with the list of UDT (User Defined Type) tags configured in the PLC program. This file is optional for the wizard. However, if this file is not selected, the UDT tags will not be imported.
  - **Create class tags when suitable**: Check this checkbox to create tags and classes for UDT tags imported from the PLC program. Uncheck this checkbox to import tags as single tags (rather than class type) from the PLC program.

In the grid displayed in Step 3 (Importing a Database) for this Data Source Type, there is an additional field with the label **Address**, which displays the name of the items from the RSLogix™ program.

**Note**: See Steps 1, 2 and 3 of Import Wizard for the settings and fields that are common for all Source Types.

### Importing from PanelBuilder32 Databases

**Note**: This wizard is sold as an add-on and requires a license to be enabled. Consult your software for further information.

This wizard allows you to import not only the tags, but also the screens, alarm configuration and communication interface from a text file (report) exported by the PanelBuilder32™ software. Using this wizard,
you can convert PanelView™ program (developed with PanelBuilder32™) into the IWS format and run them under any platform supported by IWS.

- **Import Screens**: Check this option to import the graphical screens (including their objects and animations) to IWS.
- **Import Comm. Configuration**: Check this option to import the communication interface (tags linked to PLC addresses) to IWS.
- **Report File**: Press the Browse button to select the name of the text file exported from PanelBuilder32™ (report printed to a text file).
- **ControlLogix Only**: When importing a program that was configured to exchange data with ControlLogix PLCs, IWS can convert the communication interface to Ethernet/IP (ABCIP driver). To do so, type the IP Address of the PLC and its slot number. This information will be used to create the communication interface for the imported program. If the original program was already configured to use the Ethernet/IP interface, these fields can be left blank, because the IP Address and CPU Slot Number are retrieved from the program file itself.

In the grid displayed in Step 3 for this Data Source Type, there is an additional field with the label **Address**, which displays the tag addresses from the PanelBuilder project.

**Tip:** Please consult the documentation for this import wizard for detailed information about how to export an program from the *.PBA format to the text (*.TXT) format, using PanelBuilder32™, and import it into IWS.

**Note:** See Steps 1, 2 and 3 of Import Wizard for the settings and fields that are common for all Source Types.

**Note:** IWS does not support some special characters (e.g., [ ] . –) in tag names. When you import your PaneBuilder database into IWS, these special characters will be converted into underscores (_).

**Importing PanelMate programs**

This wizard allows you to import not only the tags, but also the screens, alarm configuration, and communication settings from an operator interface program that was created with PanelMate™ software.

Using this wizard, you can convert a PanelMate program into a IWS project and then run it on any platform supported by InduSoft Web Studio. The wizard can import programs from PanelMate Plus 4.01 (or earlier) and PanelMate Power Pro 2.21 (or earlier), and it supports the following communication drivers:
• Allen-Bradley Serial (ABKE)
• Modbus Serial, ASCII and RTU (MODBU)
• Modbus Plus (MODPL)
• Allen-Bradley Remote I/O (STRIO)

**Note:** This import wizard is sold as an add-on for InduSoft Web Studio, and it requires a license to be enabled. For more information, consult your software vendor.

Also, if you are running InduSoft Web Studio on a Windows operating system that has User Account Control (UAC) enabled, then you may have problems using this import wizard. Close the application, and then run it again as an administrator (i.e., right-click the InduSoft Web Studio program icon, and then click **Run as administrator** on the shortcut menu).

**Import Screens**

Check this option to import the graphical screens (including their objects and animations) to InduSoft Web Studio.

**PanelMate Model**

Click **Browse** to select the directory where the database files of the PanelMate program that you intend to import are stored.

**Application**

After specifying the correct file path in the **PanelMate Model** box, the programs that are available in that directory will be available in this combo-box. Select the program that you want to import, and then click **Next**.

See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common to all Source Types.

**Importing from TwinCAT PLC Databases**

This wizard allows you to import database variables from a TwinCAT PLC program that has been developed with Beckhoff’s TwinCAT software. Also, when you run the import, InduSoft Web Studio automatically creates and configures a TWCAT driver worksheet, eliminating the need to manually configure communication between the two systems.

This feature works with all versions of TwinCAT through version 3.x.
Note: Please refer to Step 2 of the Import Wizard instructions for settings that are common to all sources.

The Import TwinCAT PLC Database Wizard dialog allows you to configure the following settings:

- **AMS Net ID**: Enter the AMS Net ID of the TwinCAT PLC that you want to communicate with. For example: 5.0.112.206.1.1
- **TCP Port**: Select the port on which the PLC’s runtime system has been configured to run. You can either select one of the standard ports (e.g., 801, 811, 821, 831) or specify a custom port number.
- **Symbol File**: Click the Browse button to select the TwinCAT symbol file (.SYM or .TPY) that contains the variables to be imported. For more information, see Export a symbol file from your TwinCAT project on page 379.

You are now ready to import the variables into your IWS project database. Return to Step 3 of the Import Wizard.

**Importing from OMRON CX Programmer Databases**

Note: This import wizard creates the communication driver for the OMRON communication driver, which is enabled only for customers that purchase the product directly from OMRON. Consult your software vendor for further details. Moreover, the OMRON communication driver communicates with the OMRON PLCs by the FINS Gateway, which is supported for the Windows 2000/XP operating systems. Therefore, the FINS Gateway must be installed on the computer to enable communication between IWS and the PLCs through the OMRON driver.

This wizard allows you to import tags from a program for OMRON PLCs developed with CX Programmer and exported to a CXT file. When importing tags from the CX Programmer CXT file, the OMRON driver
worksheet is automatically created to link the tags imported with the PLC, eliminating the need to configure the communication interface between IWS and the PLC manually.

• **Prefix**: This box allows you to concatenate one of the following types of prefixes to the tags imported from the CX Programmer program:
  - **Custom**: Check this option to concatenate a custom prefix with up to 8 characters to the name of the imported tags.
  - **PLC**: Check this option to concatenate either the PLC name or the PLC Number to the name of the imported tags.
  - **Program**: Check this option to concatenate either the Program name or the Program Number to the name of the imported tags.

• **Serial Auto Address**: This area allows you to configure the Network Address and the Initial Node Address for the PLCs configured in the product with Serial communication (if any):
  - **Network Address**: This setting will be applied to all PLCs configured in the project with Serial communication.
  - **Node Address**: This setting will be applied to the first PLC configured in the project with Serial communication. This setting will be incremented and applied to subsequent PLCs configured in the product with Serial communication.

• **CXT File**: Click the Browse button to select the CXT file, exported by CX Programmer, from which the tags will be imported.

In the grid displayed in Step 3 for this Data Source Type, there is an additional field with the label **Address**, which displays the name of the tags from the CX Programmer program.

**Note**: See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

**Import a Studio XML Screen**

Use the Import Wizard to import a Studio XML Screen, which is an external text file created with InduSoft Web Studio’s custom XML schema.

Before you begin this task, you must have a properly formatted Studio XML Screen file that you can import.

A Studio XML Screen file contains the same information as a regular screen file (*.SCR). It is simply formatted as human-readable XML instead of binary data, which makes it more flexible and portable.

XML files can be manually created in any text editor, of course, but our intent is to enable you to quickly and programmatically create large numbers of screens in Microsoft Visual Studio using our custom XML schema and its associated C# API. This often requires additional training and support, so the schema and API are
not included in the standard installation of InduSoft Web Studio. Instead, the schema, API, documentation, sample files, and support contract can be purchased as a separate toolkit. For more information, please contact your software vendor.

Once you have created your screens, you can use the Import Wizard to batch import them into your IWS project.

**Tip:** You can also use the function ImportXML to import Studio XML Screen files during run time.

To import one or more Studio XML Screens:

1. On the **Home** tab of the ribbon, in the **Tools** group, click **Import Wizard**. The **Import Wizard** dialog box is displayed.
2. In the **Source Type** list, click **Studio XML Screen**, and then click **Next**. The next step of the import wizard is displayed.

3. Under **Options**, choose whether imported screens should automatically replace existing screens in your project. Screens are considered to be duplicates if they have the same file name. For example, Objects.xml and Objects.scr would be duplicates.
   - If you do not want the imported screens to replace existing screens in your project, select **Do not import duplicate screens**. A warning will be displayed for each duplicate that you try to import.
   - If you want the imported screens to automatically replace existing screens in your project, select **Replace duplicates with imported screens**.
4. Click **Browse**. A standard **Open** dialog box is displayed.
5. Use the file browser to locate and select the Studio XML Screen files that you want to import, and then click **Open**. You can Ctrl-click and Shift-click to select more than one file. The selected file(s) are displayed in the **Screen(s)** box.
6. Click **Next**.
The selected screens are processed, and then the next step of the import wizard is displayed showing the project tags that are included in the selected screens.

7. In the list of project tags, select the tags that you want to import with the screens:
   - For each tag in the list, select or clear the check box to the left.
   - To select all of the check boxes, click Check. To clear all of the check boxes, click Uncheck.
   - To filter the list of tags, click Filter and then configure filter strings for one or more columns. You can use wildcard characters (* and ?) in the filter strings.

8. Click Finish.

The screens and included tags are imported into your project. Also, the screens are automatically published for Thin Clients and Mobile Access — i.e., the corresponding *.HTML, *.SCC, and *.SSMA files are automatically generated and saved in your project’s Web folder, so the imported screens should be immediately available for you to select.
About tag integration

Tag integration allows you to directly integrate tags from a third-party control application or device into your IWS project.

Tag integration is an enhanced framework for device communication. It is built on the same communication drivers that are described in the Drivers section, but instead of manually configuring driver worksheets to associate project tags with device registers, you can use the Object Finder to browse the device registers and add them directly to your project.

Registers added in this way appear as tags in the Shared Database folder. They are "live" — that is, they are continuously and bilaterally updated during run time as long as the source is also running and properly connected — and they can be used anywhere in your project that you would normally use project tags.

These tags count against your target system’s tag limit only when you use them in your project. This is different from the driver worksheets, where you might create a group of tags in order to associate them with device registers but then never actually use them in your project.

Tag integration is configured in the Communication tab of the project settings.

This feature is supported only for certain third-party control applications and devices, because additional work is required to upgrade an existing communication driver for tag integration. However, many of the communication drivers included with InduSoft Web Studio can be upgraded, so if you do not see the one you want, then please contact your software vendor and ask about custom development.

Tip: By default, the project runtime will update integrated tags every 600 milliseconds, which is the rate at which the system tag BlinkSlow toggles. To adjust the rate, manually edit the project file (projectname.APP) to add the following entry:

```
[Options]
MainDrvAlwaysTrigger=tagname
```

tagname can be either another system tag (e.g., BlinkFast, Second, Minute) or a tag that you have created. Whenever the value of the tag changes, the integrated tags will be updated.

This works because the project runtime automatically creates a virtual Main Driver Sheet to manage the integrated tags. Please keep in mind that the same trigger is used by all Main Driver Sheets in your project.
Edit the tags database during run time

You can use the Tags Database (TagsDB) functions to add and remove project tags, classes, and class members during run time, as well as to configure properties and alarm conditions on project tags.

There are several important things to keep in mind when using the TagsDB functions, because they do much more than set and get tag values. They actually change the structure of the tags database, which can cause serious problems for a running project and all connected clients if it is not done properly. As such, most of the TagsDB functions can be executed only under the following conditions.

First, the InduSoft Web Studio development application must be installed and running on the project server, it must be fully licensed for Engineering+Runtime, and the project must be open in the application. The TagsDB functions use the development application’s database editor in essentially the same way that you do when you manually edit your project during run time. Because of this limitation, the TagsDB functions cannot be used at all in projects designed for and downloaded to Windows Embedded devices.

Second, the TagsDB functions can be called only from scripts executed on the project server. None of the connected clients — not even the project client running on the same computer as the project server, because it runs as a separate process on that computer — can make structural changes to the tags database without interfering with other clients, decreasing run-time performance, and potentially corrupting the database. Therefore, generally speaking:

- The functions can be called from the Startup Script (which is executed when the project itself is run), Script Groups (which are continuously executed by the Background Task), and any Global Procedures called by them; and
- The functions cannot be called from the Graphics Script (which is executed separately by each client), Screen Scripts (which are attached to individual screens), and Command animations.

**Tip:** To work around these restrictions, do one of the following:

- Create a Global Procedure to call the TagsDB functions, and then call the function `RunGlobalProcedureAsync` to run that procedure; or
- Create a Script Group to call the TagsDB functions, configure a tag/expression to control the execution of that Script Group, and then change the value of that tag/expression.

Third, in any script that calls TagsDB functions to make structural changes to the tags database, you must first call the function `TagsDBBeginEdit` in order to lock the database for editing and prevent any other run-time changes. Then, at the end of the script, you must call the function `TagsDBEndEdit` in order to finish the changes that were made and allow the database to resume normal run-time behavior. Both functions must be called in the same script, because that script (more specifically, the program thread running that script) effectively owns the tags database while it is locked. You cannot call `TagsDBBeginEdit` in one script and then `TagsDBEndEdit` in another.

When a project is edited during run time, the project server and every connected client must be updated with the changes as they are made. Normally, this is not a problem when you manually edit your project, because you make your changes slowly and one at a time. In contrast, the TagsDB functions allow you to make a large number of changes quickly, so updating the project server and the connected clients with all of those changes while the project is running can severely decrease run-time performance. Therefore, to maintain performance and protect the tags database, the project runtime — including all background tasks such as alarms, trends, and other scripts — is effectively paused and the changes are applied as a batch.

**Note:** The function `TagsDBBeginEdit` has a persistent effect. This means that if you call the function to lock the tags database during project run time and then stop the project, the database will remain locked and you will not be able to manually edit it.

Restarting the project may or may not unlock the database, depending on how you designed your project and which script locked the database in the first place. Instead, you should use the Database Spy to manually call the function `TagsDBEndEdit`. When that is executed successfully, you can safely restart your project.
**Examples**

The following example shows how to use the TagsDB functions in VBScript to add a new class, then add a new class member to that class, then add a new tag of that class, then set an alarm and a trend on that tag:

```vb
If($TagsDBBeginEdit()=0) Then
    If($TagsDBAddClass("TempClass")=0) Then
        If($TagsDBAddClassMember("TempClass","TempMember","Real")=0) Then
            If($TagsDBAddTag("TempTag","TempClass",2,0)=0) Then
                If($TagsDBSetAlarm("TempTag[0].TempMember",1,0,3.5)<=0) Then
                    $Msg = "Alarm not Set"
                End If
                If($TagsDBSetTrend("TempTag[0].TempMember",0,1)<=0) Then
                    $Msg = "Trend not Set"
                End If
            Else
                $Msg = "Tag not created"
            End If
        Else
            $Msg = "Class Member not added"  
        End If
    Else
        $Msg = "Class not created"
    End If
Else
    $Msg = "Tag functions not enabled"
End If
```

In particular, please note how the script begins with the function `TagsDBBeginEdit` and then ends with the function `TagsDBEndEdit`. Also, see how the nested `If…Then…Else` structures ensure that each function is executed successfully (i.e., returns a value of 0) before the next one is attempted.

Here is another example that shows how to remove the alarm, trend, tag, class member, and class, in reverse order from how they were added in the previous example:

```vb
If($TagsDBBeginEdit()=0) Then
    If($TagsDBRemoveAlarm("TempTag",1)<=0) Then
        $Msg = "Alarm not removed"
    End If
    If($TagsDBRemoveTrend("TempTag")<=0) Then
        $Msg = "Trend not removed"
    End If
    If($TagsDBRemoveTag("TempTag")=0) Then
        If($TagsDBRemoveClassMember("TempClass","TempMember")<>0) Then
            $Msg = "Class member not removed"
        End If
        If($TagsDBRemoveClass("TempClass")<>0) Then
            $Msg = "Class not removed"
        Else
            $Msg = "Tag not removed"
        End If
    Else
        $Msg = "Tag not removed"
    End If
Else
    $Msg = "Tag functions not enabled"
End If
```

It is not absolutely necessary to remove the alarm and trend before removing the tag they are on, because they are discarded with everything else when the tag is removed. They are included in the example simply to be thorough. In contrast, the class member and class cannot be removed until the tag is removed successfully.
The most basic function performed by IWS is to provide a window into the process. The ability to display the status of the process by interacting with instrumentation (or computers), is described as the Human-Machine Interface (HMI).

IWS allows you to create projects that can monitor processes using high-resolution color screens.

The IWS graphic tools consist of two modules:
- The Screen/Worksheet Editor in the IWS development environment (used to create or import graphics); and
- The runtime project Viewer.

You can use animations to create dynamic graphic objects or symbols. Animations cause objects and symbols to change appearance to reflect changes in the value of a tag or an expression. Each screen is an association of static and animated objects.

Screens can have an optional bitmap that acts as a background in the object window. On the following screen for example, the static images can be part of a bitmap in the background object and objects with animation in the animation object layer can reflect the changes in the plant, giving the illusion that the screen is three-dimensional.

All IWS configuration tasks require a Windows-compatible pointing device, such as a mouse or touch pad. You can run a project in the Viewer without a pointing device if you configure keypad or keyboard keys for all commands.
Graphics tab

The Graphics tab of the Project Explorer contains all of the screens, screen groups, and symbols in your project.

![Graphics tab of the Project Explorer](image)

The folders on the Graphics tab are described in the following sections:

**Screens**
You create screens to provide a graphical interface for your project. Each screen can contain many buttons, sliders, dials, indicators, graphs, and so on.

**Screen Groups**
You can combine individual screens into screen groups, so that they all open together at the same time.

**Thin Clients**
You can deploy your project as a web application to be accessed by thin clients such as desktop web browsers, tablets, and smartphones. You can even deploy different versions of your project with different levels of functionality for each type of client.

**Project Symbols**
This folder contains all of the custom symbols that you create for your project. A symbol is a group of interconnected screen objects that work together to perform a single function — for example, lines, rectangles, and text fragments that have been arranged to make a slider control.

**Graphics Script**
You can use this worksheet to define VBScript sub-routines that are called only when the graphics module starts (i.e., when a client station connects to the server and displays the graphical interface), while it is running, and when it ends.

**Symbols**
The symbols library contains not only the custom symbols that you create (see Project Symbols above), but also a large selection of premade symbols that are installed with the development application.

**Layout**
The layout editor displays all of the screens the are currently open for editing. You can use it to visualize how the screens are arranged together and reuse screens in multiple layouts — for example, to create a common navigation bar across your entire project.
**Screens folder**

The Screens folder is located in the Graphics tab of the Project Explorer. It contains all of your Screen worksheets, both completed and still in development.

To create a new Screen worksheet, do one of the following:
- On the Insert tab of the ribbon, in the Graphics group, click Screen;
- Right-click the Screens folder in the Project Explorer, and then click Insert on the shortcut menu; or
- Click the Application button, click New on the Application menu, click the File tab in the New dialog, select Screen from the list of worksheet types, and then click OK.

When a Screen worksheet is opened for the first time, the Screen Attributes dialog for that worksheet is automatically displayed. For more information, see Screen Attributes dialog.

To open an existing Screen worksheet, expand the Screens folder and then double-click the worksheet.

**SCREEN ATTRIBUTES DIALOG**

The Screen Attributes dialog is used to configure runtime settings — such as size, location, title bar, security level, and screen logic — for a specific project screen.

**Accessing the dialog**

The Screen Attributes dialog is automatically displayed when you add a new Screen worksheet.

You can also access the dialog for an existing Screen worksheet (assuming the worksheet is open for editing) by doing one of the following:
- On the Graphics tab of the ribbon, in the Screen group, click Attributes; or
- Right-click anywhere in the Screen worksheet and then click Screen Attributes on the shortcut menu.

**The dialog in detail**

![Screen Attributes dialog](image)

**Elements in Screen Attributes dialog**

<table>
<thead>
<tr>
<th>Group / Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A brief description of the project screen. This is not shown anywhere during runtime.</td>
</tr>
<tr>
<td>Background Picture</td>
<td>Enable Background</td>
</tr>
<tr>
<td></td>
<td>Enables the background picture layer and specifies the file type of the picture.</td>
</tr>
</tbody>
</table>
When this option is selected, a new BMP file with the same name as the screen is automatically saved in the Screen sub-folder of your project folder (e.g., `\project_name\Screen\screen_name.BMP`). You can then edit this image using a third-party image editor. For more information, see Modifying a screen's background color or image on page 177.

<table>
<thead>
<tr>
<th>Group / Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screens and Graphics</td>
<td>Uses the specified image file located in the Screen sub-folder of your project folder.</td>
</tr>
<tr>
<td></td>
<td>If you want to specify a tag/expression that provides this value, so that you can programmatically change the value during runtime, then the tag/expression must be enclosed in curly brackets. For example: <code>{myTag}</code>. Do not include the extension in the file name. If the file format is not BMP, then use the list to the right of the Enable Background option above to select the correct format.</td>
</tr>
</tbody>
</table>

Note: Only BMP files are supported in projects developed for Windows Embedded target systems.

<table>
<thead>
<tr>
<th>Group / Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Width</td>
</tr>
<tr>
<td></td>
<td>Height</td>
</tr>
</tbody>
</table>

| Location | Top | The default distance (in pixels) between the top of the computer display and the top of the screen when the screen is initially displayed. The user can change the location during runtime if the screen is set to have a border and title bar; see Border below. |
| Left | The default distance (in pixels) between the left side of the computer display and the left side of the screen when the screen is initially displayed. The user can change the location during runtime if the screen is set to have a border and title bar; see Border below. |

<table>
<thead>
<tr>
<th>Runtime Properties</th>
<th>Style</th>
<th>The general runtime behavior of the screen:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overlapped</td>
<td>Opens the screen without closing any other screens.</td>
</tr>
<tr>
<td></td>
<td>Popup</td>
<td>Forces the screen in front of all other screens but does not close them.</td>
</tr>
<tr>
<td></td>
<td>Replace (Partial)</td>
<td>Opens the screen and closes all other Replace screens that it partially covers. This is the default.</td>
</tr>
<tr>
<td></td>
<td>Dialog</td>
<td>Similar to Popup, except that the other screens are also disabled until the dialog is closed by the user.</td>
</tr>
<tr>
<td></td>
<td>Replace (Complete)</td>
<td>Similar to Replace (Partial), except that it closes only other Replace screens that it completely covers.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Border</th>
<th>The type of border around the screen:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Thin</td>
</tr>
<tr>
<td></td>
<td>Resizing</td>
</tr>
</tbody>
</table>

<p>| Titlebar | Shows the window's title bar with the specified window name. |</p>
<table>
<thead>
<tr>
<th>Group / Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Menu</td>
<td>Provides a menu of basic window commands at the left end of the title bar.</td>
</tr>
<tr>
<td>Maximize Box</td>
<td>Shows the Maximize button at the right end of the title bar.</td>
</tr>
<tr>
<td>Minimize Box</td>
<td>Shows the Minimize button at the right end of the title bar.</td>
</tr>
<tr>
<td>Don't Redraw</td>
<td>While this tag/expression evaluates as TRUE, the screen's graphics are not updated.</td>
</tr>
<tr>
<td>Disable Commands</td>
<td>While this tag/expression evaluates as TRUE, the screen is locked against user interaction but the graphics continue to be updated.</td>
</tr>
<tr>
<td>Security</td>
<td>The minimum security level that a user must have to access this screen.</td>
</tr>
<tr>
<td>Screen Logic</td>
<td><em>On Open</em> Lists expressions to be evaluated once when the screen is opened, similar to a Math worksheet.</td>
</tr>
<tr>
<td></td>
<td><em>While Open</em> Lists expressions to be continuously evaluated while the screen is open, similar to a Math worksheet.</td>
</tr>
<tr>
<td></td>
<td>If you also configure a tag/expression in <em>Trigger</em>, then instead of being continuously evaluated, the listed expressions will be evaluated once each time the value of the trigger changes while the screen is open.</td>
</tr>
<tr>
<td></td>
<td><em>On Close</em> Lists expressions to be evaluated once when the screen is closed, similar to a Math worksheet.</td>
</tr>
<tr>
<td>Multi-Touch Settings</td>
<td>Customizes the Multi-Touch settings for this screen. For more information, see About the Multi-Touch settings for project screens on page 258.</td>
</tr>
<tr>
<td>Focus</td>
<td><em>Receive focus on open</em> When the screen is opened, the focus will automatically go to the first object in the screen (according to Object ID) that can receive focus, as if the user tabbed into the screen.</td>
</tr>
<tr>
<td></td>
<td><em>Share tab order with other screens</em> When the user tabs through the last object in the screen, the focus will go to the next open screen (according to Tab Order below) rather than back to the first object in the current screen.</td>
</tr>
<tr>
<td></td>
<td><em>Tab Order</em> Similar to Object ID for screen objects, this determines the tab order between screens when multiple screens are open. When the user tabs through the last object in a screen, the focus will go to the open screen with the next higher Tab Order number. Each screen should have a unique Tab Order number between 0 and 32767.</td>
</tr>
<tr>
<td></td>
<td><em>Background screen</em> When the user clicks on the screen, it remains in the background and is not brought in front of the other open screens. If more than one screen has this option selected, then the screens are arranged in tab order with the greatest Tab Order number being the farthest back.</td>
</tr>
<tr>
<td>Performance Optimization</td>
<td><em>Hide screen instead of closing it</em> Closing the screen (by any user action or system process) in fact only makes it hidden, and reopening the screen makes it visible again. This makes the screen appear to open very quickly.</td>
</tr>
<tr>
<td></td>
<td><em>Keep screen file in memory</em> When the screen is closed, the screen file is kept in memory so that it does not need to be reloaded from the hard drive when the screen is reopened. The screen still needs to be redrawn, however. This is not as fast as making a hidden screen visible again (see above), but it still makes the screen appear to open quickly.</td>
</tr>
</tbody>
</table>

**MODIFYING A SCREEN'S BACKGROUND COLOR OR IMAGE**

A project screen can have either a solid background color or an editable background image.

**Selecting a screen's background color**

By default, a newly created project screen has a solid white background. To change this background color:

1. Make sure the screen file is open for editing.
2. On the **Graphics** tab of the ribbon, in the **Screen** group, click **Background Color**. A standard color picker is displayed as a shortcut menu.

3. Use the color picker to select a color. The color is applied to the entire project screen.

   **Tip:** If you want to set a background color for only part of a screen, draw a *shape object* and then send it to the back.

---

**Enabling a screen's background image**

To enable the background image for a screen and then edit it:

1. Make sure the screen file is open for editing.

2. On the Graphics tab of the ribbon, in the Screen group, click **Attributes**. The **Screen Attributes** dialog box is displayed.

3. Select **Enable Background**. A new BMP file with the same name as the screen is automatically saved in the Screen folder in your project folder (e.g., `[...]\projectname\Screen\screenname.BMP`).

4. Click **OK** to close the **Screen Attributes** dialog box.

5. On the **Graphics** tab of the ribbon, in the **Screen** group, click **Background Image**. Microsoft Paint is run automatically and the BMP file is opened for editing.

6. Use Microsoft Paint to edit the background image as needed.

   **Tip:** To use an image editor other than Microsoft Paint, manually edit the file `[...]\InduSoft Web Studio v7.1\Bin\Program Settings.ini` to add the following setting:

   ```
   [Options]
   ImageEditor=filepath
   ```

---

**Specifying an existing image file as the background**

To select an existing image file, especially if it is in a format other than BMP:

1. Copy the image file that you want to use to the Screen folder in your project folder.

2. In the development environment, make sure the screen file is open for editing.

3. On the **Graphics** tab of the ribbon, in the **Screen** group, click **Attributes**. The **Screen Attributes** dialog box is displayed.

4. Select **Enable Background**, and then in the list to the right of the option, select the image file format.

   **Note:** Only BMP files are supported in projects developed for Windows Embedded target systems.

5. Select **Shared Image**, and then in the box to the right of the option, type the name of the image file. Do not include the file extension, because that is controlled by the list selection in the previous step. You can specify folders within the Screen folder. For example, if you type `MyBackgrounds\Background1`, the program will look for the image file at `[...]\projectname\Screen\MyBackgrounds\Background1`. You can also specify a tag or expression in curly brackets (e.g., `{MyTag}`), in order to programmatically change the background image during run time.

6. Click **OK** to close the **Screen Attributes** dialog box. If the program can find and load the specified image file, the image will be displayed in the project screen. If not, then a warning message will be displayed.

---

**Screen Group Folder**

The **Screen Group** folder combines individual screens from the **Screens** folder into more manageable groups. To open a specific screen group, open the **Screen Group** folder and right-click on the subfolder.
To remove a specific screen group, right-click on its subfolder and click the prompt screen to delete.

To create a new screen group:

1. On the Insert tab of the ribbon, in the Graphics group, click Screen Group to open the Insert Screen Group dialog:

   ![Insert Screen Group dialog]

2. Type a name for the new folder into the Name field.

3. Create a group of screens for this folder by selecting screens from the List of screens list. To select multiple screens press the Ctrl key as you click on the screen names. Release the Ctrl key when you finish.

   This list contains only those screens currently located in Screens folder.

4. Click OK to close the Insert Screen Group dialog.

**Layout Tool**

Click the Layout tool to open the Layout tab.

This interface displays all screens currently open in the Screen Editor and allows you to:

- **Modify the Screen Attributes**: Right-click on the screen displayed on the Layout tab and use the alignment options or the Screen Attributes link to modify the screen position. You can also click and drag the screen to change its position (Top and Left) or resize it (Width and Height).

- **Visualize how the screens fit together during runtime**. This option is especially useful when creating pop-up/dialog screens or groups of screens.

**Note**: The screens open in the Layout tab according to the order that they are open in the development environment. When you change the position of the screen tabs in the development environment (to the left or to the right), you will change the order in which these screens will be displayed in the Layout tab.

**Tip**: Right-click on the title of the Layout tab to display the option to enable/disable the Auto Scale. If you enable this option, the screens will be auto-scaled automatically to fit in the Layout tab.
Using Screen Objects and Animations

*Editing*

The *Editing* group provides tools for general screen editing.

![Editing group](image)

**SELECTION**

On the Graphics tab of the ribbon, in the *Editing* group, click **Selection** to display a mouse cursor that you can use to select and move objects on the screen.

**DISABLING DRAG IN A SCREEN**

You can disable the dragging of objects in the screen editor, to prevent accidental moves after you’ve laid out the screen exactly as you want it.

On the **Graphics** tab of the ribbon, in the **Editing** group, click **Disable Drag**.

**REPLACING PROJECT TAGS IN A DOCUMENT OR SCREEN OBJECT**

To replace all occurrences of a tag in the current document, do one of the following:

- On the Home tab of the ribbon, in the Tags group, click **Replace**;
- On the Graphics tab of the ribbon, in the **Editing** group, click **Replace**.

To replace all occurrences of a tag in a screen object, double-click the object to open its **Object Properties** dialog and then click **Replace**.

All of these methods will open the **Replace** dialog, which is described below.

You can replace one or more tags by clicking the **Whole Tag Name** tab. Current tags used are displayed. The original tag names are shown in the **From** column on the left, and you can enter your new tag names in the **To** column on the right.

![Whole Tag Name tab](image)

Note that this does not **rename** or **delete** any tag — it only replaces the tags used in the object with other tags from the database.
You can also replace one or more strings (e.g., button captions, descriptive text) by clicking the **String Value** tab.

![String Value tab](image)

When you are done, click **OK**.

**OBJECT PROPERTIES DIALOG**

This dialog shows the configurable properties of a screen object or animation. Each type of screen object and animation has its specific properties, but all types have a few properties in common.

**Accessing the dialog**

To access the *Object Properties* dialog for a specific screen object, do one of the following:

- Select the screen object, and then on the **Graphics** tab, in the **Editing** group, click **Properties**;
- Select the screen object, and then press **Alt+Enter**;
- Right-click the screen object, and then click **Properties** on the shortcut menu; or
- Double-click the screen object.

**The dialog in detail**

All *Object Properties* dialogs contain the following elements:

<table>
<thead>
<tr>
<th>Area / Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pushpin</td>
<td>When the pin button is released, the focus is passed to the object on the screen as soon as that object is selected. When the pin button is pressed, the focus is kept on the <em>Object Properties</em> window even when you click the objects on the screen.</td>
</tr>
<tr>
<td>Replace</td>
<td>Launches the <em>Replace</em> dialog, where you can replace strings, tags or properties for the selected object or group of objects.</td>
</tr>
</tbody>
</table>
| Hint                | Tooltip displayed during runtime, when the user hovers the mouse cursor over the object. This can be used to provide quick-help to the user.  

The text in the **Hint** field is also temporarily written to the *Hint* system tag, so that you can trigger actions based on the value of this tag when the mouse cursor is moved over a specific object.  

To show hints/tooltips during runtime, the **Enable Tooltip** option must be selected in the *Project Settings* dialog. You can enable/disable this feature separately for full project viewers (**Viewer** on the Project tab of the ribbon) and for thin clients (**Thin Client** on the Project tab of the viewer). |
| Curve               | The combo-box at the right side of the dialog allows you to select the specific object, group of objects, or animation that must be edited. |

**GRID SETTINGS**

To show/hide the grid in the screen editor, click **Grid Settings** on the Graphics tab of the ribbon and then click **View Gridlines** on the shortcut menu.

To edit the grid settings, do one of the following:
• Click **Grid Settings** on the Graphics tab of the ribbon and then click **Grid Settings** on the shortcut menu; or
• Right-click anywhere in the screen editor and then click **Grid Settings** on the shortcut menu.

Either method will open the **Grid Settings** dialog:

![Grid Settings dialog](image)

**UNDO**

Select **Undo** to cancel the last action performed (and up to 20 actions taken prior to the last action) while working on a screen. (**Object Properties** actions do not increase **Undo** steps.)

**Note:** Using the **Undo** menu option is the same as using **Undo** tool located on the **Standard** toolbar.

**FORMAT TAB**

The **Format** tab of the ribbon is used to format and arrange objects in a project screen.

![Format tab of the ribbon](image)

**Note:** This tab is available only when you've selected one or more objects in a project screen.

The tools are organized into the following groups:

• **Arrange:** Arrange objects in a project screen, including *bring to front and send to back, group, align, and rotate*.
• **Position:** Precisely adjust the *position* of a screen object in a project screen.
• **Size:** Precisely adjust the *size* of a screen object.
• **Style:** Change the *fill* and *line color* of a screen object.
• **Fonts:** Change the *caption font* of a screen object.

**Shapes**

The **Shapes** group provides the following tools, which you can use to create polygons, rectangles, lines, and other objects for your screen.

![Shapes group](image)
LINE OBJECT

On the Graphics tab, in the Shapes group, click Line to draw an orthogonal line in the drawing area, as follows:

1. Click the left mouse button to set the starting point of the line.
2. Drag the cursor to adjust the line size.
3. Click again to place the object.
4. To view the object properties, double-click on the object. The Object Properties dialog displays as follows.

![Object Properties: Line](image)

Use the Object Properties dialog to specify the following parameters for the orthogonal line:

- **Line**: Specify a line style by clicking the No Line, Solid Line, or Dashed Line button.
- **Color**: Specify a line color by clicking the Color button. When the Color dialog opens, click a color to select it and then close the dialog.
- **Weight**: Specify the line width (in pixels) by typing a number representing the line width into the text box.

**Note**: Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

OPEN POLYGON OBJECT

On the Graphics tab, in the Shapes group, click Open Polygon to draw an open polygon with a border in the specified foreground color.

To draw an open polygon in the drawing area:

1. Click the left mouse button to set the starting point of the polygon.
2. Move the cursor to a new location and click again to place the second vertex.
3. Repeat this process until you create the desired polygon shape.
4. Double-click to stop drawing the polygon.

To change the shape of a polygon after you’ve drawn it, select it and drag any of its points.

**Tip**: If a polygon’s individual points are not draggable, they may be grouped. To ungroup the points, right-click on the polygon and choose Ungroup from the shortcut menu.
To view the object properties, double-click on the polygon object and the *Object Properties* dialog is displayed as follows.

![Object Properties Dialog: Open Polygon]

**Object Properties: Open Polygon**

Use the *Object Properties* dialog to specify the following parameters for the polygon:

- **Line**: Specify a border line style by clicking the *No Line*, *Solid Line*, or *Dashed Line* button.
- **Color**: Specify a border line color by clicking the *Color* button. When the Color dialog opens, click on a color to select it and then close the dialog.
- **Weight**: Specify the borderline width (in pixels) by typing a number representing the line width into the text box.

**Note**: Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

**CLOSED POLYGON OBJECT**

On the *Graphics* tab, in the *Shapes* group, click *Closed Polygon* to draw a closed polygon, using a border in the specified foreground color.

To draw a closed polygon in the drawing area:

1. Click the left mouse button to set the starting point of the polygon.
2. Move the cursor to a new location and click again to place the second point.
3. Repeat this process until you create the desired polygon shape.
4. Double-click or right-click to stop drawing the polygon.
5. To view the object properties, double-click on the polygon object.

To change the shape of a polygon after you’ve drawn it, select it and drag any of its points.

**Tip**: If a polygon’s individual points are not draggable, they may be grouped. To ungroup the points, right-click on the polygon and choose *Ungroup* from the shortcut menu.

The *Object Properties* dialog is displayed as follows.

![Object Properties Dialog: Closed Polygon]

**Object Properties Dialog: Closed Polygon**

Use the *Object Properties* dialog to specify the following parameters for the polygon:

- **Line**: Specify a border line style by clicking the *No Line*, *Solid Line*, or *Dashed Line* button.
• **Color**: Specify a border line color by clicking the **Color** button. When the **Color** dialog opens, click a color to select it and then close the dialog.

• **Weight**: Specify the borderline width (in pixels) by typing a number representing the line width into the text box.

**Note**: Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

• **Fill**: To specify whether the polygon is filled, click **No Fill** or **Fill**.

  If you enable the Fill option, you can specify a fill Color by clicking on the **Color** button. When the **Color** dialog displays, click a color to select it and close the dialog.

**RECTANGLE OBJECT**

On the **Graphics** tab, in the **Shapes** group, click **Rectangle** to create rectangles, as follows:

1. Click in the drawing area and drag the mouse/cursor to draw the rectangle.
2. Release the mouse button when the rectangle is the size you want.
3. Double-click on the object to view the **Object Properties** dialog.

![Object Properties: Rectangle](image)

**Object Properties: Rectangle**

Use the **Object Properties** dialog to specify the following parameters for the orthogonal line:

• **Type**: Specify a border line style by clicking on **None**, **Solid**, **Dashed**, **Etched**, **Raised** or **Sunken**.

• **Color**: Specify a border line color by clicking the **Color** button to open the **Color** dialog. Click the color to select it, and then close the dialog.

• **Weight**: Specify a border line width by typing a number representing the line width (in pixels) into the text box provided.

**Note**: Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

• **Fill**: Specify whether to fill the rectangle by clicking **No Fill** or **Fill**.

  If you select the Fill option, specify a fill color by clicking on the Color rectangle. When the Color dialog displays, click a color to select it and close the dialog.

• **Color**: Specify a fill color by clicking the **Color** button to open the **Color** dialog. Click a color to select it, then close the dialog.
• **Caption**: Press this button to open the Caption dialog where you can edit the text that can be written inside the rectangle object:

![Caption dialog](Image)

**Caption dialog**

- **Caption**: Enter the text that you want to display inside the rectangle object. You can include a tag by enclosing it in curly brackets (e.g., `{tagname}`).
- **Fonts**: Specify a font style for the caption by clicking the Fonts button.
- **Align**: Specify the alignment for the caption of the rectangle.
- **Multiline**: Allow the caption of the rectangle to be shown in more than one line, when checked.
- **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e.g., `{1.2345}`) or a tag of Real type (see Caption above), then the value will be formatted according to the virtual table created by the SetDecimalPoints function.
- **Wrap Text**: When checked, the object automatically wraps the text when necessary.
- **Auto gray out**: Turns the caption of the rectangle to gray when the Command animation applied to the rectangle is disabled by the Disable field or due to the Security System.
- **Enable translation**: Click (check) to enable translation during runtime using the Translation Tool.

**ROUNDED RECTANGLE OBJECT**

On the Graphics tab, in the Shapes group, click Rounded Rectangle to draw rounded rectangles (empty or filled), as follows:

1. Click in the drawing area and drag the mouse/cursor to create the rectangle.
2. Release the mouse button to stop drawing the object.
3. Double-click on the object to view the Object Properties dialog.

![Object Properties: Rounded Rectangle](Image)

- **Note**: You cannot use the rounded rectangle tool to create a bar graph for projects running on Windows Embedded target systems.
- **Tip**: A rounded rectangle has one extra handle in the bottom-right corner, which enables you to modify the arc angle.

Use the Object Properties dialog to specify the following parameters for the orthogonal line:
• **Line**: Specify a borderline style by clicking the *No Line*, *Solid Line*, or *Dashed Line* button.

• **Color**: Specify a borderline color by clicking the *Color* button to open the *Color* dialog. Click the color to select it and then close the dialog.

• **Weight**: Specify a borderline width by typing a number representing the line width (in pixels) into the text box provided.

**Note**: Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

• **Fill**: Specify whether the rectangle is filled by clicking *No Fill* or *Fill*.

  If you select the **Fill** option, specify a fill color by clicking on the **Color** button. When the **Color** dialog displays, click a color to select it and close the dialog.

• **Color**: Specify a fill color by clicking the **Color** button to open the **Color** dialog. Click a color to select it, then close the dialog.

• **Caption**: This option is not enabled for this object.

### ELLIPSE OBJECT

On the **Graphics** tab, in the **Shapes** group, click **Ellipse** to draw ellipses, chords, arcs, and rings (see the following figures).

![Ellipse, Chord, Arc, and Ring](image)

**Tip**: The **Ring** style is particularly useful when you are creating plumbing drawings.

To create an ellipse, use the following steps:

1. Click in the drawing area and drag the mouse/cursor to create an ellipse shape.
2. Release the mouse button to stop drawing the ellipse.
3. Use the **Object Properties** dialog to change the shape to a chord, arc, or ring.
4. Double-click on the object to view the **Object Properties** dialog.

![Object Properties: Ellipse](image)

Use the **Object Properties** dialog to specify the following parameters for the ellipse:

• **Style**: Specify the object style by selecting **Ellipse**, **Arc**, **Chord**, or **Ring** from the drop-down list. Next, select **Left-Bottom**, **Left-Top**, **Right-Bottom**, or **Right-Top** from the **Style** list to choose the quadrant into which the ellipse is drawn.

  For example to represent a half-circle pipe, create two **Ring** objects. Specify one as **Left-Bottom** and the other as **Right-Bottom** then join the two objects to create a half-pipe.

• **Line**: Specify a line style for the ellipse border by clicking the *No Line*, *Solid Line*, or *Dashed Line* button.
• **Color**: Specify the ellipse borderline color by clicking the **Color** button to open the **Color** dialog. Click the color to select it, then close the dialog.

• **Weight**: Specify a line width for the ellipse border by typing a number representing the line width (in pixels) into the text box provided.

**Note**: Dashed lines/borders of weight greater than 1 are not supported in projects running on Windows Embedded.

• **Fill**: To specify whether the ellipse is filled, click **No Fill** or **Fill**.

If you select the **Fill** option, specify a fill color by clicking on the **Color** button. When the **Color** dialog displays, click on a color to select it and close the dialog.

**PASTE A BITMAP IMAGE INTO A SCREEN**

To paste a bitmap image into a screen, copy it to the clipboard and then paste it directly into the Screen worksheet.

This task assumes that you have a Screen worksheet open for editing.

Please note that by default, using this method to add an image to a project screen will embed the image data in the screen file. This keeps everything in a single file, which is more convenient in some situations, but it increases the screen file size each time you reuse the image and it is also less flexible than linking to an external image file. Nevertheless, if this is what you want to do, then proceed with the task below.

If this is not what you want to do, however, then you have two other options.

First, if you want all bitmap images to be linked as external files, then you should first select the option **Save pictures in separate files**, in the project settings. When that option is selected and you paste a bitmap image into a screen, the image is automatically saved as a separate file in your project folder and then placed as a **Linked Picture** object in the project screen. For more information about the option **Save pictures in separate files**, see **Project Settings: Viewer**.

Second, if you want only a specific bitmap image to be linked as an external file, then you should not paste it into the screen. Instead, you should manually place it as a **Linked Picture** object. This leaves you the ability to paste and embed other bitmap images, according to the default settings.

The following table summarizes your options:

<table>
<thead>
<tr>
<th>Action</th>
<th>Save pictures in separate files...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...is selected</td>
</tr>
<tr>
<td>Paste image into screen</td>
<td>Image is placed as a Linked Picture object</td>
</tr>
<tr>
<td>Link to external image file</td>
<td>Image is placed as a Linked Picture object</td>
</tr>
</tbody>
</table>

To paste a bitmap image into a screen:

1. Open the desired image in an appropriate image editor, such as Microsoft Paint.
2. Select part or all of the image, either by using the **Select** tool in Paint or by pressing **Ctrl-A**.
3. Copy the selection to the clipboard, either by using the **Copy** tool in Paint or by pressing **Ctrl-C**.
4. Switch to the development application, and then make sure that you have the correct Screen worksheet open for editing.
5. On the **Home** tab of the ribbon, in the **Clipboard** group, click **Paste**, or press **Ctrl-V**. The bitmap image is placed as a screen object.
6. Click and drag the object to where you want it to be positioned in the screen. Also, click and drag the object’s handles to resize it, if necessary.
7. Double-click the screen object.

If **Save pictures in separate files** is selected in the project settings, then the bitmap image was placed as a Linked Picture screen object. Skip the remaining steps of this task and instead proceed to **Link to an external image file** on page 232.
On the other hand, if **Save pictures in separate files** is not selected, then the bitmap image was placed as a Bitmap screen object. Proceed with the remaining steps below.

The **Object Properties: Bitmap** dialog is displayed.

![Object Properties: Bitmap](image)

8. If you want some part of the picture to be transparent to the screen background and other objects, then select a transparent color:
   a) Select **Transparent**.
   b) Click and drag the tracker on the screen object until it is positioned over a sample of the desired transparent color.
      The tracker is an additional handle on the screen object that initially appears just inside the bottom-right corner of the object. Moving the tracker on the object does not move or resize the object itself.

9. Close the **Object Properties** dialog.

**CHANGE THE PROPERTIES OF MULTIPLE SCREEN OBJECTS**

This task describes how to select two or more screen objects and then change the properties that are common to the selected objects.

Before you begin this task, you must have a project screen open in the screen editor.

Which properties you can change depends on whether you select multiple objects of the same type or of different types. If the objects are of the same type, you can change the properties that are specific to that type. For example, if you select multiple Button objects, then you can change the properties that are specific to Button objects.

![Object properties of multiple selected Button objects](image)

For more information about the properties of a specific type of object, see the documentation for that object.

**Note:** You can only use this method to change the properties of Shapes and Active Objects. You cannot use this method to change the properties of Data Objects, Animations, Library items, or objects in a group.

In contrast, if you select multiple objects of different types, you can change the properties that are common to all of the objects. This includes not only cosmetic properties like Border and Background, but also functional...
properties like Disable, Security, Enable Translation, and E-Sign. (Some properties may not apply to all objects. For example, Button objects do not have Border and Rectangle objects do not have Security.)

In both cases, the dialog box shows the current values of the properties of the last selected object. It is only when you actually change the value of a property that the change is applied to the selected objects. All other properties are left unchanged, regardless the values shown in the dialog box.

To change the properties of multiple screen objects:

1. In the screen editor, do one of the following:
   • Press and hold either Shift or Ctrl on the keyboard, and then click each object that you want to change; or
   • Use the cursor to draw a selection box around all of the objects that you want to change.
   The objects are selected.

2. Do one of the following:
   • On the Graphics tab of the ribbon, in the Editing group, click Properties;
   • Right-click the selected objects, and then on the shortcut menu, click Properties; or
   • Press Alt+Enter on the keyboard.
   
   Note: You cannot double-click to open the Object Properties dialog box as you otherwise would, because clicking like that clears the selection.

   The Object Properties dialog box is displayed for the selected objects.

3. Change the property values that you want to change, and then close the dialog box.
   The changes are applied to all of the selected objects.
**Active Objects**

The *Active Objects* toolbar provides the following tools, which you can use to create interactive objects. Active objects typically require more parameters than simple shapes.

![Active Objects group]

**TEXT OBJECT**

On the *Graphics* tab, in the *Active Objects* group, click *Text* to create text objects, as follows:

1. Click in the drawing area. When a cursor displays, you can type a line of text.
2. After entering the text string, double-click on the new text object to view the *Object Properties* dialog.

![Object Properties: Text]

Use the *Object Properties* dialog to specify the following properties:

- **Caption**: Specify a text string by typing a caption in the text box.
- **Text data link** button: Click to apply the *Text Data Link animation* to the Text object.
  
  If the caption doesn’t include any placeholder characters (###) for the text-data link, then clicking this button also automatically appends those characters.
- **Align**: Align the text by selecting *Left*, *Center*, or *Right* from the combo-box.
- **Fonts**: Specify a font style for the text by clicking the *Fonts* button. When the *Fonts* dialog displays, you can specify the following parameters:
  
  - Font (typeface)
  - Font style
  - Size
  - Effects
  - Color
  - Script
- **Border**: Specify a text border by clicking the *Border* box.
  
  To select a border color, click the *Color* rectangle. When the *Color* dialog displays, click a color to select it, then close the dialog.
- **Background**: Specify a background color by clicking the *Color* button. When the *Color* dialog displays, click a color to select it, then close the dialog.
- **Enable translation** (optional): Specify an external translation file for the text by clicking (checking) this box.
DRAW AND CONFIGURE A TEXT BOX OBJECT

Draw a Text Box screen object to create a way to input and/or output text, including multiple lines.

Before InduSoft Web Studio v7.1, the only way to input and output text on screen was to draw a Text object and then apply a Text Data Link animation to it. This was restricted to a single line of text per object, however, and it required additional steps to combine the object, animation, and any background or border shapes.

The new Text Box object is an OS-style text input/output box that can be configured to show multiple lines. When the object is associated with a String array, each line of the box corresponds to an array element: Line 1 is Array Index 0, Line 2 is Array Index 1, Line 3 is Array Index 2, and so on.

Otherwise, the Text Box object works much like the Text Data Link animation.

To draw and configure a Text Box object:

1. Open a screen for editing.
2. On the Graphics tab of the ribbon, in the Active Objects group, click Text Box. The mouse cursor changes to a crosshair for drawing.
3. Draw the object where you want it on the screen, and then further move or resize it if necessary.
4. Double-click the object. The Object Properties dialog box is displayed.

5. In the Hint box, type a hint or tooltip that you want to have displayed during run time when the mouse hovers over the object.

6. In the Tag/Expression box, type the name of a project tag or an expression to associate it with the object.

   **Tip:** Click the browse button (…) on the right to use the Object Finder to form the tag/expressions.

A project tag can be used for either input or output. An expression can be used only for output.

If you want to input or output multiple lines of text — that is, if you select Multi-line, which is described below — then type the name of a String array with an index of 0 (e.g., MyStringArray[0]). If you try to start at any other position, then you may see unexpected behavior during run time. For example, if you type MyStringArray[4], then the first line of the object will display index 4 of the array but the second line will start over at index 0.

You should not use any other type of array (i.e., Boolean, Integer, or Real) with the Multi-line option.

7. In the Format list, select how the numerical value (if any) of the specified tag/expression will be formatted and displayed on-screen. Available options include Decimal, Hexadecimal, Binary and Auto. If you select Auto, then the value will be formatted according to the virtual table created by the SetDecimalPoints function. This option does not actually change the specified tag/expression in any way. For example, Tag/Expression is set to an Integer tag, Input Enabled is selected, and Format is set to Hexadecimal. You may input a new value in hexadecimal format, but it is saved in your project database as an integer.

8. By default, Input Enabled is selected. If you do not want to allow input, clear this option.
9. In the **Mask/Count** box, type a value that will restrict the input:

- To mask a numerical value so that it matches a specific format, type one or more # characters. Each # represents one character of input/output. You can also combine # characters with regular text: $###.##

  It is important to remember that the project runtime will always display the most significant digits of a numerical value, regardless of the number or placement of # characters in the text. That means if you do not have sufficient # characters to display the value, then it will be transformed in some way depending on the format of the value (as set by the **Format** option described below):

  - In **Decimal** format, the number of decimal places is determined by the position of the decimal separator in the ### text. However, if you do not have enough # characters to the left of the decimal separator to display the whole value, then the whole value will overrun the fractional value. For example, if you try to display a value of 112.64 in #.#, you will see 112.

  - In **Hexadecimal** and **Binary** formats, if you have more # characters than you need to display the value, then the runtime project will fill in with leading zeroes. If you have less characters than you need, then the value will simply be truncated.

  - In **Auto** format, the runtime project will ignore the number of # characters and display the entire numeric or string value. Numeric values will be displayed in decimal format with their complete whole and fractional values, regardless of the placement of the decimal separator in the ### text. Given an exceptionally large value or long string, this may disrupt the layout of your screens.

- To limit a string to specific character count, type a value between 0 and 1024. 1024 is the maximum limit because it is the maximum size of the String data type. If you enter more than 1024 characters, then the string will be truncated and the remaining characters will be discarded.

  Please note that this character count limit is per line, so if you configure the object to have multiple lines (i.e., if you select **Multi-line**) and associate it with a String array, then each line of text / array element may be up to the limit.

10. In the **Minimum Value** and **Maximum Value** fields, type the minimum and maximum numerical values (if any) that will be accepted from the user. This is optional.

11. In the **Disable** box, type the name of project tag or an expression. This is optional.

    When the value of the tag/expression is TRUE (1), the object is disabled.

12. To make the object accept/display multiple lines of text, select **Multi-line**.

    When **Multi-line** is selected, the **Scroll Bar** and **Word Wrap** options also become available and the **Password** option becomes unavailable.

13. To make the object obfuscate text input (e.g., *********), select **Password**.

    **Tip:** During run time, if you want to insert a line break in a multi-line text box, press **Shift-Return**, **Ctrl-Return**, or **Alt-Return**.

---

**BUTTON OBJECT**

The Button object

On the **Graphics** tab, in the **Active Objects** group, click **Button** to create custom-sized buttons, as follows:

1. Click in the drawing area and drag the mouse/cursor to create the button shape.
2. Release the mouse button when the button is the size you want.
3. Double-click on the object to view the *Object Properties* dialog.

![Object Properties: Button]

Use the *Object Properties* dialog to specify the following parameters for the button:

- **Caption**: Specify a caption by typing the text into the text box. You can include a tag by enclosing it in curly brackets (e.g., `{tagname}`).

- **Style**: Select a style for the button:
  - **3D Sharp**: A raised, rounded button with somewhat sharpened corners, suitable for touchscreen displays.
  - **3D Soft**: A raised, rounded button with softened corners, suitable for touchscreen displays.
  - **OS Like**: A button styled to match the operating system on which the project client is running, suitable for Windows desktops running the Web Thin Client or Secure Viewer.
  - **Standard**: The standard, flat button from the previous versions of IWS.

![Examples of button styles]

- **Background color**: Select a background color for the button.

- **Align**: Select the alignment for the caption of the button.

- **Fonts**: Specify a font style for the caption by clicking the *Fonts*.
  
  When the *Fonts* dialog is displayed, specify the following parameters:
  
  - **Font (typeface)**
  - **Font style**
  - **Size**
  - **Effects**
  - **Color**
  - **Script style**

- **Images**: Insert an image file into the button by clicking the *Images* button.
  
  When the *Images* dialog is displayed, specify the following parameters:
  
  - **File**: Type the file path to the image file. You can also click the browse button to the right of the box, to open a standard Windows file browser.
Tip: Icon files (*.ICO) cannot be kept in memory, which means they must be loaded from the hard drive every time the project screen is opened or redrawn, and that may decrease your run-time performance. If you use many icon files in your project, consider replacing them with new files in another format such as GIF, JPG, or PNG. For more information, see Configure the performance control settings on page 101.

- By default, the image is displayed at its actual size. To change this, in the Size list, select Custom, and then configure the desired Width and Height (in pixels) of the image.
- **Position**: Select where the image should be positioned in relation to the caption.
- **Offset**: Specify the offset (in pixels).
- **Transparent Color**: Select which color in the image should be transparent. The background color (see above) will show through these areas.
- **Advanced**: Specify advanced settings for the button by clicking the Advanced button.

When the Advanced dialog is displayed, specify the following parameters:

- **Enable translation** *(optional)*: Specify an external translation file for the button label by clicking (checking) the box.
- **Multiline**: Allow the caption of the button to be shown in more than one line, when checked.
- **Wrap Text**: When checked, the object automatically wraps the text when necessary.
- **Auto gray out**: Turns the caption of the button to gray when the Command animation applied to the button is disabled by the Disable field or due to the Security System.
- **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e.g., \{1.2345\}) or a tag of Real type (see Caption above), then the value will be formatted according to the virtual table created by the SetDecimalPoints() function.
- **Command**: Click to automatically apply a Command animation to the button and then switch to the animation's properties.

### PUSHBUTTON OBJECT

On the Graphics tab, in the Active Objects group, click Pushbutton to create a Pushbutton object using the Command animation with an object or pre-configured pushbuttons.

IWS provides the following pre-configured button types, all of which mimic the standard panel buttons of the same name:

- **Momentary** *(default)*: Changes state *(Open or Closed)* when you press the button and reverts to its initial state when you release the button. This button type always displays in its normal position when you open the screen.
- **Maintained**: Changes state *(Open or Closed)* when you press the button but does not revert to its initial state when you release the button. You must press the button again to change its present state. This button type maintains its state across screen changes.
- **Latched**: Changes state *(Open or Closed)* when you press the button and remains in this state until you release it by changing the Reset tag.

IWS also provides the following button styles:

- Rectangular with a faceplate and indicator light
- Rectangular without a faceplate or indicator light *(default)*
- Rectangular with a 3-D
- Rectangular with a floating appearance

To add one or more pre-configured buttons to a screen:

1. Click the Pushbutton tool, and position the mouse (pointer) on the screen.
2. Click and drag to create/adjust the size of the rectangular button.
The button size and text font characteristics determine how much text you can display and how much area you can touch on a touch screen. You can resize the button and change the font characteristics later to permit longer messages to be shown in a given space.

3. Double-click on the object to open the Object Properties dialog.

Tip: Alternatively, you can right-click on the pushbutton object or highlight the object, press Alt + Enter, and select Properties from the resulting shortcut menu to open the Object Properties dialog.

Object Properties: Pushbutton

You can use this dialog to specify the following parameters:

- **Type** drop-down list: Click to select the pushbutton type (Momentary (default), Maintained, or Latched).

- **State** drop-down list: Click to specify a default state for the pushbutton (Normally Open (default) or Normally Closed).

  Click the button to toggle between its default and non-default state (according to its specified Type). For example, in the button’s initial state, it may conform to characteristics specified in the Open area of the Configuration dialog (see below). Click the button again to toggle to the opposite state, which in this example is Closed, and conform to characteristics specified in the Closed area.

- **Tag/Exp** text box: Type a tag or an expression to accomplish the following:
  - Type in a tag to receive the Write Value from the appropriate state (Open or Closed) area in the Configuration dialog.
  - Type an expression to execute On Down, when you press the pushbutton down.

  Note: IWS does not write the result of any expression in the Tag/Exp field into a tag.

- **Indicator** text box: Type a tag to define an indicator that causes the button to change to a specified color when the tag value matches one of two specified values. You must define both the colors and tag values in the Configuration dialog. If you leave this field blank, the indicator changes color automatically when you press the button.

- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.

- **Reset** text box (active for Latched pushbutton type only): Type a tag to control the button’s latched state, as follows:
  - Type a zero and the button will remain in a latched state after you press it.
  - Type a nonzero value and a latched button will become unlatched after you press it. You must reset the tag value to zero before you can press the button again.

- **Key** area: Specify a keyboard key or create a key combination to toggle a pushbutton when you have no pointing device (mouse or touch screen) or if you want to create shortcut keys in addition to pushbuttons.

- **Key** drop-down list: Type a key in the text box or select a non-alphanumeric key from the drop-down list. Enter a single character or key only. Numbers are not valid entries for this field.
Click (check) the Shift, Ctrl, or Alt box to create a combination key, meaning the Shift, Ctrl, or Alt key must be pressed with the key specified in the drop-down list.

Click the browse button ... to open the Key Modifier dialog, which enables you to modify your combination keys. You can choose Left, Right or Left or Right to specify the position on the keyboard of the Shift, Ctrl or Alt key in the combination key. If you choose Left or Right, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

Tip: If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

- **Disable** text box: Type a tag using a nonzero value to disable this pushbutton so that pressing the button has no effect. This box is empty by default, which also enables the Command animation.
- **Ext Trans.** checkbox: Click (check) to translate the text automatically using pre-configured translation worksheets. (See the Translation Tool for more information.)
- **Security** text box: Type a value to specify a security level (0 to 255) for this button. If the user does not have the specified security level, the button becomes inactive. If the user has the appropriate security level, or you leave this field blank, the button remains active.
- **Config** button: Click to open the Configuration dialog, which allows you to specify style and state parameters for the pushbutton:

![Configuration dialog](image)

This dialog provides the following parameters:

- **Style** combo-box: Click the combo-box button to select a pushbutton style (Rectangle (default) or Rectangle with Indicator).
- **Effect** combo-box: Click to select a 3-D effect for the pushbutton.
  - **Floating** (default): Buttons resemble a flat object with a shadow
  - **3D**: Buttons have beveled edges and appear to "depress" into the screen when pressed.

You can use the **Style** and **Effect** parameters in combination to create four different buttons, as shown in the following figures:

![Pushbutton Styles](image)

- **Align**: Specify the alignment for the caption of the pushbutton.
- **Button Color** box: Click to specify a default color for the button area of a pushbutton object that includes an indicator and a faceplate. When the Color dialog displays, click on a color to select it, and close the dialog.
• **Legend Plate Color** box: Click to specify or change a default color for the legend plate area of a pushbutton object that includes an indicator. When the Color dialog displays, click on a color to select it, and close the dialog.

A legend plate encloses a button and indicator light. This field becomes inactive if the pushbutton Style does not include an indicator.

• **Open** and **Closed** areas: The following parameters are used to configure the appearance of a pushbutton object in its open and closed states.

  • **Color** box: Click to specify a default color for an indicator in each State. When the Color dialog displays, click on a color to select it, and close the dialog.

    If you selected a pushbutton style that does not include an indicator, you can use this field to specify a button color for each State.

  • **Blink** combo-box: Click to specify whether the color you specified in the Color box blinks and how fast it blinks for each state (None (no blinking), default, Slow, and Fast).

    If you set the color to blink, it alternates between the color specified in the Color box and the Legend Plate Color (if an indicator) or the Button Color (if a button).

  • **Caption** text box: Use this text box to enter the caption of the button. Alternatively, if the button style includes an indicator, the legend plate. You can include a tag by enclosing it in curly brackets (e.g., {tagname}).

  • **Fonts** button: Click to open the Font dialog, which you can use to specify or change the message font characteristics for each state.

  • **Text Blink** combo-box: Click to specify whether the text you specified blinks and how fast it blinks for each state (None (no blinking), default, Slow, and Fast). Unlike a blinking color, blinking text appears and disappears.

  • **Write Value** combo-box: Click to select a value in either field. When the pushbutton is in the appropriate state (Open or Closed), IWS writes this value to the tag specified in the Tag/Exp field (Object Properties dialog).

• **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e.g., {1.2345}) or a tag of Real type (see Caption above), then the value will be formatted according to the virtual table created by the SetDecimalPoints() function.

**CHECK BOX OBJECT**

The Check Box object is useful to create interfaces where the users can enable/disable an option on the display.

On the Graphics tab, in the Active Objects group, click Check Box to create a Check Box object on your screen:

1. Click in the drawing area and drag the mouse/cursor to draw the check box and its label.
2. Release the mouse button when the object is the size you want.
3. Double-click on the object to view the Object Properties dialog.

```
[Objects]
```

**Tip:** To change the default size of the check box, edit your project file (project_name.app) to add the following setting:
CheckBoxSize=height_in_pixels

Doing this will change the size of all check boxes in your project.

Use the Object Properties dialog to specify the following parameters for the Check Box object:

- **Caption**: Specify a caption by typing the text into the text box. You can include a tag by enclosing it in curly brackets (e.g., `{tagname}`).
- **Fonts**: Specify a font style for the caption by clicking the Fonts button.
- **E-Sign**: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.
- **Confirm check box**: Click (check) this box to ensure IWS prompts you to confirm the action at runtime.
- **Key drop-down list**: Select a key from the list to associate that keyboard key with the object or group of objects. You can then press this key to check/uncheck the check box.

Click (check) the Shift, Ctrl, or Alt box to create a combination key, meaning the Shift, Ctrl, or Alt key must be pressed with the key specified in the drop-down list.

Click the browse button ... to open the Key Modifier dialog, which enables you to modify your combination keys. You can choose Left, Right or Left or Right to specify the position on the keyboard of the Shift, Ctrl or Alt key in the combination key. If you choose Left or Right, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

**Tip**: If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

- **Disable field**: Type a tag or expression into this field to enable and disable the object. You disable the Check Box object when you enter a value different from 0.
- **Security field**: Type a value in this field to specify a security level for the object, as defined under Security. When a user logs on, and does not have the specified security level, IWS disables the object.
- **Tag field**: When the user clicks on the check box during runtime, the value of this tag is updated. If no Feedback was specified, the value of this tag is also used to indicate the current status of the object.
- **True Value field**: Specify a value that will be used to change the control to TRUE state and to indicate that the control is in TRUE state. For more information about states, please refer to the states table.
- **Advanced button**: Press this button to open the Advanced dialog:

```
Advanced dialog

- **Tri-State**: If enabled the control has a third state. The third state will be displayed when the tag configured in the Feedback field assumes the value specified in the Tri-State field. If the Feedback field is left in blank, the third state will be displayed when the tag configured in the Tag field assumes the value specified in the Tri-State field.

**Caution**: The Tri-State field must not be configured with the same value as the True Value field, nor with an empty string value.

- **Feedback**: Value that indicates the state of the object (FALSE, TRUE, or TRI-STATE). When the value of the tag configured in Feedback is equal to the value of the tag configured in True Value, the state is set to TRUE. When the value of the tag configured in Feedback is equal to the value of the tag configured in Tri-State, the state is set to TRI-STATE. When none of these conditions are satisfied, the state is set to
FALSE. If the Feedback field is left in blank, then the tag configured in the Tag field will be used as the Feedback tag.

- **Ext Trans.**: When this option is checked, the caption of the object supports the translation.
- **Auto gray out**: Turns the caption of the object to gray when it is disabled by the Disable field or due to the Security System.
- **Force**: Click (check) this box to force the Tag Database to recognize a tag change when the user clicks on the object, even if the value of the tag in question does not change.
- **Enable Focus**: When this option is checked, the object can receive the focus during runtime by the navigation keys.
- **Push Like**: When this option is checked the control is displayed as a button, instead of the standard check box standard shape.
- **Fill Color**: Specify the fill color for the button. This option is enabled only when the Push Like option is checked.
- **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e.g., \{1.2345\}) or a tag of Real type (see Caption above), then the value will be formatted according to the virtual table created by the SetDecimalPoints function.

**Modes of Operation**

The Check Box object can operate in two different modes: Normal and Tri-State. For more information, see Modes of operation for Check Box and Radio Button objects.

**RADIO BUTTON OBJECT**

The radio button object is useful to create interfaces where the users can chose one option from multiple options on the display.

On the Graphics tab, in the Active Objects group, click Radio Button to create a radio button object on your screen:

1. Click in the drawing area and drag the mouse/cursor to draw the radio button and its label.
2. Release the mouse button when the object is the size you want.
3. Double-click on the object to view the Object Properties dialog.

![Object Properties: Radio Button](image)

**Tip**: To change the default size of the radio button, edit your project file (project_name.app) to add the following setting:

```plaintext
[Objects]
RadioButtonSize=height_in_pixels
```

Doing this will change the size of all radio buttons in your project.

Use the Object Properties dialog to specify the following parameters for the radio button object:

- **Caption**: Specify a caption by typing the text into the text box. You can include a tag by enclosing it in curly brackets (e.g., \{tagname\}).
- **Fonts**: Specify a font style for the caption by clicking the Fonts button.
- **E-Sign**: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.

- **Confirm** checkbox: Click (check) this box to ensure IWS prompts you to confirm the action at runtime.

- **Key** drop-down list: Select a key from the list to associate that keyboard key with the object or group of objects. You can then press this key to check/uncheck the radio button.

  Click (check) the **Shift**, **Ctrl**, and/or **Alt** boxes to create a combination key, meaning the Shift, Ctrl, and/or Alt key must be pressed with the key specified in the drop-down list.

  Click the browse button ... to open the **Key Modifier** dialog, which enables you to further modify your combination keys. You can choose **Left**, **Right**, or **Left or Right** to specify the position on the keyboard of the Shift, Ctrl, or Alt key in the combination key. If you choose **Left or Right**, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

  **Tip**: If you have defined custom keys for your project, you can select them in this list. For more information, see **Define custom keys for selected screen objects**.

- **Disable** field: Type a tag or expression into this field to enable and disable the object. You disable the radio button object when you enter a value different from 0.

- **Security** field: Type a value in this field to specify a security level for the object, as defined under Security. When a user logs on, and does not have the specified security level, IWS disables the object.

- **Tag** field: When the user clicks on the radio button during runtime, the value of this tag is updated. If no **Feedback** was specified, the value of this tag is also used to indicate the current status of the object.

- **True Value**: Specify a value that will be used to change the control to TRUE state and to indicate that the control is in TRUE state. For more information about states, please refer to the states table.

- **Advanced**: Press this button to open the **Advanced** dialog:

  ![Advanced dialog](image)

  - **Tri-State**: If enabled the control has a third state. The third state will be displayed when the tag configured in the **Feedback** field assumes the value specified in the **Tri-State** field. If the **Feedback** field is left in blank, the third state will be displayed when the tag configured in the **Tag** field assumes the value specified in the **Tri-State** field.

    **Caution**: The **Tri-State** field must not be configured with the same value as the **True Value** field, nor with an empty string value.

  - **Feedback**: Value that indicates the state of the object (FALSE, TRUE, or TRI-STATE). When the value of the tag configured in **Feedback** is equal to the value of the tag configured in **True Value**, the state is set to TRUE. When the value of the tag configured in **Feedback** is equal to the value of the tag configured in **Tri-State**, the state is set to TRI-STATE. When none of these conditions are satisfied, the state is set to FALSE. If the **Feedback** field is left in blank, then the tag configured in the **Tag** field will be used as the **Feedback** tag.

  - **Ext Trans.**: When this option is checked, the caption of the object supports the translation.

  - **Auto gray out**: Turns the caption of the object to gray when it is disabled by the **Disable** field or due to the Security System.

  - **Force**: Click (check) this box to force the Tag Database to recognize a tag change when the user clicks on the object, even if the value of the tag in question does not change.
• **Enable Focus**: When this option is checked, the object can receive the focus during runtime by the navigation keys.

• **Push Like**: When this option is checked the control is displayed as a button, instead of the standard radio button standard shape.

• **Fill Color**: Specify the fill color for the button. This option is enabled only when the **Push Like** option is checked.

• **Auto Format**: When checked, if the caption includes a decimal value enclosed by curly brackets (e.g., \{1.2345\}) or a tag of **Real** type (see **Caption** above), then the value will be formatted according to the virtual table created by the **SetDecimalPoints()** function.

**Modes of Operation**

The Radio Button object can operate in two different modes: Normal and Tri-State. For more information, see **Modes of operation for Check Box and Radio Button objects**.

**COMBO BOX OBJECT**

On the **Graphics** tab, in the **Active Objects** group, click **Combo Box** to select a single label from a combo-box list of labels.

If the list is longer that the space allotted, a scroll bar is enabled for the list. During runtime, if you select a label from the list, the combo-box hides itself and the selected label displays in the combo box.

Double-click on the combo-box object to open the **Object Properties** dialog box.

![Object Properties: Combo Box](image)

You can use this dialog box to set the following parameters:

• **Label** text box: Type a string tag to receive the value of the label currently displayed in the combo box.

• **Position** text box: Type an integer tag, which corresponds to the label currently displayed in the combo box. Changing this tag value changes the label being displayed.

• **Disable** text box: Type a tag with a nonzero value to disable this combo box. Type a zero, or leave the field blank (default) to enable the Command animation. If you disable the combo box, it appears grayed out during runtime.

• **Data Sources** button: Click to open the **Data Sources** dialog box (see below).

• **Advanced** button: Click to open the **Combo Box - Advanced** dialog box:

![Combo Box - Advanced](image)
• **Align** combo-box: Click to specify the label alignment (Left, Center, or Right) which affects the alignment in both the combo box and its list.

• **Color** box: Click to specify a background color for the combo box. When the Color dialog box opens, click a color to select it, then click **OK** to close the dialog box.

• **Drop List Size (Items)** field: Specify the number of items that should be displayed at one time when the user clicks on the combo box. The higher the number of items, the longer the drop list will appear.

  **Note:** If this number is less than the total number of items in the list, then the drop list will also scroll.

• **Decimal Points**: Select how decimal values will be displayed on-screen:
  - **Auto Format**: Decimal values will be formatted according to the virtual table created by the `SetDecimalPoints` function.
  - **Custom**: Enter the number of decimal places to display (e.g., 2) for all decimal values.

• **Fonts** button: Click to open a standard **Font dialog box**. Use this dialog box to change the characteristics of a message font.

• **Security** text box: Type a security level for the command (0 to 255). If an operator logs on and does not have the specified security level, the command becomes inactive. If an operator logs on and does have the specified security level, or you leave this field blank, the Command animation remains active.

• **Require confirmation** checkbox: Click (check) to prompt an operator to confirm a command during runtime.

• **Enable translation** checkbox: Click (check) to enable automatic translation of the combo box labels using the Translation Table.

• **Sort** checkbox: Click (check) to display the contents of your array of labels in alphabetical order. This parameter is available only when you select the **Array Tag** type.

• **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.

• **Input Enabled** checkbox: Click (check) to allow an operator to select a label by typing the contents of that label into a tag in the **Label** field.

• **Virtual keyboard**: Virtual Keyboard type used for this object. You need to select the Virtual Keyboard option in the **Viewer** settings (**Viewer** on the Project tab of the ribbon) before configuring the Virtual Keyboard for this interface.

### Data Sources

Use the **Data Sources** dialog box to configure the items/labels that will be displayed in the Combo Box object.

![Data Sources dialog box](image)

• **Type** combo box: Select the type of data source that you want to use, and click **Settings** to configure the source. Each type of source is described in detail below.

• **Field** field (for Text File and Database only): Specify which field/column of the data source to read from.

• **Reload** field (for Text File and Database only): Enter a tag name. When the value of the specified tag changes, the combo box will reload the labels from the data source.

**Type: Static Labels**
When **Type** is set to **Static Labels**, you can configure the following settings:

![Static Labels dialog box](image)

Enter your labels — with one label per line — just as if you were editing a plain text file. You can include tags or expressions in a label by enclosing them in curly brackets. For example: Tank #1 in area {AreaNameTag}.

The labels are not sorted in any way, so be sure to put them in the order you want them displayed during runtime. The first line is position 0, the second line is position 1, and so on.

Click **OK** when you're done.

**Type: Array Tag**

When the **Type** is set to **Array Tag**, you can configure the following settings:

![Array Tag dialog box](image)

- **Array Tag**: Enter the name of a String array that contains the items for the combo box.
- **Number of Items**: Specify how much of the array should be displayed in the combo box. Keeping in mind that the combo box counts *array index 0* as the first item, if you enter a value of 4, then the combo box will display *array index 0* through *array index 3*.

Click **OK** when you're done.

**Type: Text File**

When the **Type** is set to **Text File**, you can configure the following settings:

![Grid Data – Text File dialog box](image)

- **File**: Enter the name of the text file source. You can either type the file name and its path or click the ... button to browse for it. (If the file is stored in your project folder, then you can omit the path in the name.)

Tip: You can configure tag names between curly brackets (e.g., `{tagname}`) in the **File** field.
Delimiters: Set the delimiter(s) used in the data source file. For instance, if the data will be read from a CSV (comma separated values) file, you would select the Comma option. You can even choose a custom delimiter by checking the Other option and typing the custom delimiter in the field beside it.

Click OK when you’re done.

**Type: Database**

When the Type is set to Database, you can configure the following settings:

![Database Configuration dialog box](image)

For more information, see Database Configuration on page 98.

**LIST BOX OBJECT**

The List Box screen object displays a list of messages or menu items for the user to select from. When the user selects a message, its corresponding numerical value is written to a project tag.

If the list of messages is too long to fit within the viewable area of the List Box object, the object provides scroll bars.

The user can browse the list and make a selection by doing one of the following, depending on how you design your project interface:

- Use mouse or touchscreen input to click/tap the list’s scroll bar and then select a message;
- Press the Up, Down, Esc, Tab and/or Enter keys on the keyboard or keypad; or
- Use on-screen controls (e.g., Button objects, linked symbols) that have been configured with the PostKey function to post the equivalent key codes.

Generally, when you run a project, the active List Box object displays a list of messages. On a screen containing only one List Box object and no text input boxes, the List Box object will be active automatically.

On a screen containing multiple List Box objects and text input boxes, you can use a cursor (pointing device) or the Tab key to select and activate a List Box object.

To add a List Box object to a screen:

1. On the Graphics tab of the ribbon, in the Active Objects group, click List Box.
2. Draw the List Box object in the screen, and then drag object’s handles to adjust its size.

   The height of the object and the font size determine how many messages are visible. The width of the object determines how much of the message length is visible.

   After you draw the object, you can adjust the size and font characteristics to allow more messages to display in the given space.
3. Double-click on the object to open the **Object Properties** dialog.

![Object Properties: List Box](image)

**Object Properties: List Box**

You can use this dialog to specify the following parameters:

- **Value** drop-down list (located below the Replace button): Click to select one of the following the tag values used to index the message list.
  - Boolean
  - Integer (default)
  - LSB (Least Significant Bit)

  **Note:** For more information, see the discussion about the **State** field on the **Messages Configuration** dialog.

- **Messages** button: Click to open the **Messages Configuration** dialog (see below).
- **User Enable** text box: Type a tag, expression, or a (nonzero) number to select a message in the runtime project. The default is 1 (true or enabled).
- **Control Enable** text box: Type a tag, expression, or a (nonzero) number to select a message in the runtime project — depending on the current value of the **Read/Search Tag**. The default is 1 (true or enabled).

  IWS bases this parameter on the **Value** field (in the **Messages Configuration** dialog) that you associate with the selected message. Enabling this field allows tag changes triggered by the process to affect which messages you can select.

- **Read/Search Tag** text box: Type an integer or a Boolean tag to point to a selected message based on the message **Value** field (in the **Messages Configuration** dialog). You can use the **Control Enable** and **User Enable** fields to control whether the operator or a process can alter this tag.

- **Write Tag** text box (optional): Type a string tag to receive the **Message** value of the last-selected message. When you close and reopen the screen containing a List Box object, IWS uses this tag value to determine the last message selected in the list box.

- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature when using this object.

- **Row** checkbox: Click (check) to include set up and set down arrows in the List Box object scroll bar.

- **Page** checkbox: Click (check) to include page up and page down arrows in the List Box object scroll bar.

- **Start/End** checkbox: Click (check) to include home and end arrows in the List Box object scroll bar.

- **List wrap** checkbox: Click (check) to continue displaying and scrolling the message list (starting at the opposite end) after you scroll to the beginning or end of the list.

- **Require enter for selection** checkbox: Clicking (checking) this box requires the user to press **Enter** (or post the equivalent key code) to make a selection. If this option is not checked, then a selection is made whenever focus changes to another screen object (i.e., the user tabs out of the list or clicks/taps on another object).

- **Color** boxes: Click a color box to open the **Color** dialog or the 16-color **Color Selection** dialog. Either dialog allows you to specify or change colors for the List Box object. Click a color to select it and then click **OK** to close the dialog.
• **Highlight Color** box: Specify a color for highlighting messages (default is blue).
• **Text Color** box: Specify a color for highlighting message text (default is black).
• **Win Color** box: Specify a color for the list box background (default is white).
• **Border Color** box: Specify a color for the list box border (default is black).

**Messages Configuration Dialog**

![Message Configuration dialog](image)

Use the parameters on this dialog as follows:

- **State** field (*read-only*): Use this field to view the indexed individual messages. IWS numbers this field based on the Read/Search Tag type you selected:
  - **Boolean**: Provides two valid states, labeled 0 and 1
  - **Integer**: Provides 256 valid states, labeled 0 to 255
  - **LSB**: Provides 32 valid states (i.e., the 32 bits in an integer value), labeled 0 to 31
- **Message** field: Enter the string to be displayed in the List Box object. You can include tags in a message by enclosing them in curly brackets (e.g., `{tagname}`).
- **Value** field: Type a message value matching the specified Read/Search Tag value. (Also, the same value written to the write tag.)
  
  If you specify **LSB** for the **Value** field, IWS uses the value specified in the **State** field for both the Read/Search Tag and the write tag.
- **Text Foreground** color field: Click to specify a color for the message text foreground. The color is displayed only when the message is not selected.
- **Text Blink** checkbox: Click (check) to cause a message to blink, once per second, when it is selected.
- **Fonts** button: Click to open the Font dialog, which allows you to change the characteristics (style, size, and so forth) of the message font.
- **Enable translation**: Click (check) to enable translation during runtime using the Translation Tool.
- **Auto Format**: When checked, if a message includes a decimal value enclosed by curly brackets (e.g., `{1.2345}`) or a project tag of Real type (see **Message** above), then the value will be formatted according to the virtual table created by the SetDecimalPoints function.
SMART MESSAGE OBJECT

The Smart Message screen object displays messages and images that can be changed during run time by updating the value of the associated project tag.

The following smart message object types are available:

- **Message Display**: Enables you to display any one of multiple messages within a single screen object.
- **Multistate Indicator**: Enables you to display any one of multiple messages within a single screen object, and also has the ability to display bitmap images with the messages.
- **Multistate Pushbutton**: Enables you to display messages and bitmap images. This object also resembles a multi-position switch in that it allows you to increment through the messages by clicking on the object during run time.

These smart message object types vary in their ability to display messages and graphics, write to a tag, and control how many messages and graphics display on the screen. However, all of the object types can receive process input (Read Tag value) to determine which message to display.

To add a smart message object to the screen:

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Smart Message**.
2. Draw the Smart Message object in the screen, and then drag object's handles to adjust its size.
   
   You use the object’s height, width, and font size to determine how much text and how large a bitmap image you can display on the screen.
3. Double-click on the object to open the **Object Properties** dialog.

   ![Object Properties: Smart Message]

You can use this dialog to specify the following parameters:

- **Type** combo-box: Click to select the smart message object type. The object type sets the behavior of the object during run time and the features supported by it:
  
  - **Message Display** (default)
  - **Multistate Indicator**
  - **Multistate Pushbutton**

- **Value type** drop-down list: Select the type of values used to index the message list:
  
  - **Boolean**: Provides two valid states. Use this selection when you want to display either one of two different messages, based on a boolean value (0 or 1).
  - **Integer** (default): Provides 500 valid states. Use this selection when you want to display different messages based on specific values from an Integer tag.
  - **LSB** (least significant bit): Provides 32 valid states (i.e., 32 bits in an integer value). Use this selection when you want to display different messages based on which bit from an integer tag is set. If more than one bit from the Integer tag is set simultaneously, the message associated with the least significant bit that is set (value 1) will be displayed.

   **Note:** If **Multistate Pushbutton** is the **Smart Message** type, only 16 different messages can be associated with the object, even for **Integer** or **LSB** values.
• **Switch event** drop-down list (available for **Multistate Pushbutton** only): Select one of the following options to specify when the message is changed:
  - **On Down**: Switch to the next message when you click on the object (default).
  - **While Down**: Switch to the next message continuously while you hold the mouse button down on the object.
  - **On Up**: Switch to the next message when you release the mouse button on the object.
• **Align**: Select the alignment of the text displayed by the Smart Message object.
• **Fonts**: Launches the Fonts dialog, where you can configure the font settings for the text displayed in the object.
• **Check expression on each message line** check box: When this option is cleared, only **Read Tag/Expression** is evaluated during run time and the resulting value determines which message is displayed. When this option is selected, each message has its own tag/expression.
• **Read Tag/Expression** text box: Enter a project tag or expression. The value determines which message is displayed by the object during run time.
• **Messages**: Displays the **Configuration** dialog box, where you can configure the messages for the object. See "Messages Configuration" below.
• **Write Tag** text box (optional and available for **Multistate Pushbutton** only): Enter the name of a project tag. The value associated with the message currently displayed by the object is written to this tag.
• **E-Sign** (available for **Multistate Pushbutton** only): When this option is selected, the user will be prompted to enter the Electronic Signature before executing the animation.
• **Hide border**: When this option is selected, the line border of the object is not visible.
• **Border weight**: Defines the thickness (in pixels) of the line drawn around the object.
• **Security** (available for **Multistate Pushbutton** only): System Access Level required for the object/animation.
• **Key area** (optional and available for **Multistate Pushbutton** only): Shortcut used to go to the next message (step) using a keyboard when the Multistate Pushbutton type is selected. This option is especially useful when creating projects for runtime devices that do not provide a mouse or touch-screen interface, when the keyboard is the only physical interface available to interact with your project during run time.

| **Tip**: If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects. |
Messages Configuration
The behavior of the object, and thus the layout of this dialog box, depend on whether Check expression on each message line is selected.

Configuring messages for the resulting values of Read Tag/Expression

Configuring a different tag/expression for each message line

Data Source list box: The source of the messages to be displayed by the object. Select one of the following:

- **Static**: The messages are configured and stored directly on the object. Configure a worksheet row for each possible state. For more information, see the table "Properties for Data Source: Static" below.
- **Text File**: The messages are loaded from an external source file. In the Source File box, type the path (relative to your project folder) and name of the file.

Tip: If you want to change the file path during run time, type the name of a string tag in curly brackets (e.g., \{MyTag\}) and then store the file path in that tag.

For more information about how to create and format the file, see "Source File Format" below.

The following table describes the meaning of the properties associated with each message, regardless of the Data Source:

**Properties for Data Source: Static**
When Type (in the Object Properties) is Multistate Pushbutton, the resulting value is also written to the tag configured in Write Tag.

The tag/expression associated with each Text/Message. All of these are continuously evaluated during run time, and the first one in the list that evaluates as TRUE (non-zero) determines which message is displayed on the object. If none of these evaluate as TRUE (non-zero), the message configured in the first row (Error) is displayed.

Foreground color for the messages displayed during run time.

Background color for the messages displayed during run time.

If selected, the message text will blink during run time.

Line color (Border) for the rectangle behind the message.

Background (Fill) color for the rectangle behind the message.

If selected, the rectangle behind the message will blink during run time.

Path and name of the bitmap file (*.BMP) (if any) that will be displayed when the message associated with it is selected during run time. If you do not specify the path, the bitmap file must be stored in your project folder.

Select the color that will be transparent in the graphic file, if the En. Transparent checkbox is selected.

If selected, the color selected in the Transparent field will be set as transparent in the graphic file.

Enable translation: Click (check) to enable translation during run time using the Translation Tool.

Auto Format: When selected, if a message includes a decimal value enclosed by curly brackets (e.g., `{1.2345}`) or a project tag of Real type (see Text/Message below), then the value will be formatted according to the virtual table created by the function SetDecimalPoints.

Note: The properties Graphic File, Transparent and En. Transparent are not available for the Message Display type.

Tip: You can copy data from this dialog and paste it into an Excel worksheet, and vice versa.

Source File Format

This section describes the format of the text file supported by the Smart Message object when the Data Source is Text File. The main advantage of using an external text file instead of static values is that it gives you the flexibility to change the messages during run time, by pointing to a different text file, or even by changing the content of the text file dynamically.

The text file must be created in the CSV format (comma separated values), where the comma character (`,`) is used to divide the columns (data) in each line (row) of the file. Therefore, you can use any CSV editor such as Microsoft Notepad and Microsoft Excel to create the CSV file with the messages and their properties for the Smart Message object.

The description of each property associated with the messages is provided in the Smart Message section. The order of the data in the CSV file is described in the following table:

<table>
<thead>
<tr>
<th>Column #</th>
<th>Property</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Text/Message</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Value</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Text (FG)</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Text (BG)</td>
<td>16777215</td>
</tr>
<tr>
<td>5</td>
<td>Text Blink</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Rec (FG)</td>
<td>8421376</td>
</tr>
</tbody>
</table>
When configuring text messages that have the comma character as part of the message, you must configure the whole message between quotes (e.g., "Warning, Turn the motor Off"); otherwise, the comma will be interpreted as a data separator instead of as part of the message.

- The first line of this file is equivalent to the State = Error. In other words, if there is no message associated with the current value of the tag configured in the Read Tag field, the message configured in the first row (State = Error) is displayed during run time.
- The data configured in the Value column of the first row from this file is irrelevant. This row must always be configured, regardless of the object type (even for Multistate Pushbutton).
- Only the Text/Message and Value columns are mandatory. The other columns are optional, and the default values will be used if you do not specify any value for them (see table).
- The fields Text(FG), Text(BG), Rec(FG), Rec(BG) and Transparent can be configured with the code of the color associated with it. The code can be entered directly in decimal format (e.g., 255) or in hexadecimal format using the syntax #value (e.g., #0000FF).
- The fields Text Blink, Rec Blink and En. Transparent can be configured with Boolean values 0 or 1 (0 = Unselected; 1 = selected), or with the keywords FALSE or TRUE (FALSE = Unselected; TRUE = selected).

Example:

Error Message,,0,16777215,1,8421376,16777215,1,error.bmp,0,0
Message Zero,0,0,16777215,0,8421376,16777215,0,open.bmp,65280,1
Message Ten,10,0,16777215,0,8421376,16777215,0,closed.bmp,65280,1
Message Twenty,20,0,16777215,0,8421376,16777215,0,0,0
Message Thirty,30,0,16777215,0,8421376,16777215,0,0,0

**Tip:** You can use the Smart Message editor (Data Source is Static) to configure the messages, values and colors. To do so, select the configuration, copy it and paste it into an Excel worksheet. Then, you can save the Excel worksheet as a CSV file (File > Save As). This procedure provides you with a user friendly interface for configuring the color codes.

**CHANGE THE PROPERTIES OF MULTIPLE SCREEN OBJECTS**

This task describes how to select two or more screen objects and then change the properties that are common to the selected objects.

Before you begin this task, you must have a project screen open in the screen editor.

Which properties you can change depends on whether you select multiple objects of the same type or of different types. If the objects are of the same type, you can change the properties that are specific to that type.
For example, if you select multiple Button objects, then you can change the properties that are specific to Button objects.

For more information about the properties of a specific type of object, see the documentation for that object.

**Note:** You can only use this method to change the properties of Shapes and Active Objects. You cannot use this method to change the properties of Data Objects, Animations, Library items, or objects in a group.

In contrast, if you select multiple objects of different types, you can change the properties that are common to all of the objects. This includes not only cosmetic properties like Border and Background, but also functional properties like Disable, Security, Enable Translation, and E-Sign. (Some properties may not apply to all objects. For example, Button objects do not have Border and Rectangle objects do not have Security.)

In both cases, the dialog box shows the current values of the properties of the last selected object. It is only when you actually change the value of a property that the change is applied to the selected objects. All other properties are left unchanged, regardless the values shown in the dialog box.

To change the properties of multiple screen objects:

1. In the screen editor, do one of the following:
• Press and hold either **Shift** or **Ctrl** on the keyboard, and then click each object that you want to change; or
• Use the cursor to draw a selection box around all of the objects that you want to change.
The objects are selected.

2. Do one of the following:
   • On the **Graphics** tab of the ribbon, in the **Editing** group, click **Properties**;
   • Right-click the selected objects, and then on the shortcut menu, click **Properties**; or
   • Press **Alt+Enter** on the keyboard.

**Note:** You cannot double-click to open the **Object Properties** dialog box as you otherwise would, because clicking like that clears the selection.

The **Object Properties** dialog box is displayed for the selected objects.

3. Change the property values that you want to change, and then close the dialog box.
The changes are applied to all of the selected objects.

**Libraries**

**SYMBOLS LIBRARY**

The Symbols library is a visual browser for all of the symbols that are available to be used in your project screens. To open the library: on the **Graphics** tab of the ribbon, in the **Libraries** group, click **Symbols**.

The Symbols library is displayed:

![Symbols library](image.png)

The library is divided into two main folders: the **Project Symbols** folder contains your **user-made symbols** for the current project, and the **System Symbols** folder contains all of the premade symbols (sorted by type) that are installed with the InduSoft Web Studio software.

To select a symbol and place it in a project screen:

1. Find the symbol you want in the library, and then double-click it. The mouse cursor will change to indicate that you have a symbol waiting to be placed.
2. Return to the project screen where you want to place the symbol.
3. Click anywhere in the project screen to place the selected symbol.
4. Edit the symbol’s object properties as needed, including any custom properties.
For more information, see Save your own project symbols on page 215.

**Save your own project symbols**

A symbol is a set of one or more screen objects that is saved in the Symbols library, so that you can reuse it in your projects.

Every time you reuse a symbol, you actually make a copy that is linked to the master symbol in the Symbols library. (These linked copies are also called “instances” of the symbol.) Thereafter, if you make any changes to the master symbol, those changes automatically propagate to every linked copy in every project.

You can customize each linked copy of the master symbol by defining **Custom Properties**. For example, when you create a gauge that displays tank levels and then save that gauge as a master symbol, you can define custom properties on the symbol that will allow each linked copy to display the level of a different tank.

![Project Symbols folder in the Graphics tab](image)

**Create a master symbol**

To create a master symbol and save it to the Symbols folder:

1. Design your symbol just as you would normally draw a project screen, using any combination of Static and Active Objects. For example, three check boxes in a rectangular pane:

![Drawing objects in a project screen](image)

2. Select the object(s) or **Group** that you want to save as a symbol.

![Selecting the objects](image)

**Note:** It is not necessary to make a Group out of two or more objects before saving them as a symbol. Saving the objects together as a symbol effectively groups them.

There is a situation, however, where you may want to group the objects first. A symbol can have only one hint. If more than one object has a hint configured on it (in the **Object Properties**), those hints are not shown when the objects are saved together as a symbol. To specify a hint for the symbol as a whole, you
must first group the objects and then configure the hint on the Group. That hint will carry through when you save the Group as a symbol.

3. Right-click on the selection, and then click **Create Linked Symbol** on the shortcut menu.

![Creating a linked symbol](image)

4. A standard **Save As** dialog box is displayed, and you are prompted to give the new symbol a file name. Symbol files (*.sym) are saved in the \Symbol folder of your project.

![Saving the symbol file](image)

5. Click **Save** to save the file. The symbol appears in the Project Symbols folder, in the **Graphics** tab of the Project Explorer.

![Symbol file in the Project Explorer](image)

The symbol also appears in the **Project Symbols** folder of the **Library**.

The symbol is now ready to be reused in your project, but the way it is currently saved, every copy will have identical properties. You must now define custom properties on the symbol — that is, the properties you want to be able to customize each time you reuse the symbol.

**Edit the master symbol**

You can edit a master symbol after you’ve initially saved it, to add or delete objects in the symbol or to define custom properties on it. Remember that any changes you make to the master symbol will automatically propagate to every linked copy in every project.
Note: There is one exception. If you change the hint on a symbol — as described in "Create a master symbol" — then the change will appear only on new instances of the symbol. Existing instances will be unchanged.

To edit a symbol:

1. Right-click on the symbol file in the Symbols folder, and then choose Edit from the shortcut menu.

   ![Editing the symbol file]

Tip: You can also right-click on any instance of the symbol and choose Edit Linked Symbol from the shortcut menu.

The symbol file is opened for editing in its own window. This symbol editor works in the same way as a regular screen editor, except that every object in the window is part of the symbol. If you add, move or delete objects in the symbol editor, you may change the size or shape of the symbol and disrupt the layout of any Screens where it is used.

Besides adding, moving or deleting objects in the symbol, you can also edit the Object Properties as you normally would. You may want some properties to be the same in every instance of the symbol, but other properties need to be customized according to where and how the symbol is used. In this example, you probably want to customize the captions for the three check boxes, the tags with which the check boxes are associated, and the caption for the pane itself.

2. Select the first object in the symbol and open its Object Properties. For example, the first check box:

   ![Object Properties dialog box for the first Check Box object]

3. In any field where you would normally configure a tag, expression, or value, you can instead define a custom property using the syntax…

   # [Category.]Property: [Value]

   ...where:
• **Category** is an optional name for a collection of related properties, such as all captions or all Check Box values. If you do not specify **Category** for a property, it will be automatically listed under the “Main” category.

• **Property** is a label to identify the specific property. **Property** is required for each property, and it must always be followed by a colon (:).

• **Value** is an *optional* default value for the property.

**Note:** All Tag/Expression syntax rules apply to Value, including tag names, pointers, arrays, strings, numerical and boolean values, and scripting functions.

In the following example, we want to be able to customize which tag will be set when the Check Box is selected or cleared. So, in the **Tag** field, type `#Option1.Tag:` as shown.

![Defining a custom property for the Tag property](image)

When you go to complete the properties on an instance of the symbol, `#Option1.Tag:` will appear like this:

![Custom properties on a symbol](image)

But more about that later...

4. Depending on the context, some object properties require a specific type of value like a String, a Boolean or a numerical value. For these properties, you must enclose the custom property declaration in curly brackets `{}`.

In this example, the **Caption** field requires a String, so type `{#Option1.Caption:"Option 1"}` as shown.

![Defining a custom property for the Caption property](image)
Again, when you go to complete the properties on an instance of the symbol, they will appear like this:

![Custom properties on a symbol](image)

5. Repeat steps 2 through 4 as needed, to define the rest of the custom properties on the symbol.
   In this example, the finished symbol has all of the following properties:

![Custom properties on a symbol](image)

6. Save the symbol and close the symbol editor.

7. On the Home tab of the ribbon, in the Tools group, click **Verify**. This will update all existing instances of the symbol in your project.

**Add tooltips to custom properties**

You can configure a description for each custom property available in the symbol. After creating a symbol, open it with the symbol editor, right-click in the symbol editor (not on the symbol itself) and choose **Edit Symbol Properties** from the shortcut menu.
When assigning values to the custom properties of the symbol on the screens, the user can read the description as Tooltips just by moving the mouse cursor on the property name, as illustrated on the following picture:

![Tooltip showing the description of the property](image)

**Password protect a symbol**

You can put a password on any of your user-made symbols to prevent them from being edited or analyzed by other users. To protect a symbol:

1. In the Symbols folder, right-click on the desired symbol file (.sym) and then choose **Password Protection** from the shortcut menu. A **Password Protection** dialog box is displayed.
2. In the **New Password** field, type your password.
3. In the **Confirm Password** field, type your password again.
4. Click **OK**.

Once this is done, you will be prompted for the password whenever you attempt to edit the symbol or unlink an instance of the symbol.

**Make a user-made symbol available to other projects**

User-made symbols are normally available only in the project where they were initially created and saved. However, you can send a user-made symbol to the **System Symbols** folder of the **Library**, to make it available to all of your projects:

1. In the Symbols folder of the **Project Explorer**, right-click the desired symbol file (.sym) and then choose **Send to System Symbols** from the shortcut menu. A standard **Save As** dialog box is displayed, automatically pointing to the \Symbol sub-directory of the IWS program directory instead of the \Symbol sub-folder of your project folder.

![Saving a symbol](image)

2. Choose a location in which to save the symbol file. You can choose one of the existing categories/folders, or you can create a new one.
3. Click **Save**. The symbol file is saved in the specified location and the symbol is displayed in the **System Symbols** folder of the Symbols library.

For more information, see Using the Library.

**Insert a symbol in a project screen**

To insert a symbol in a project screen and then complete its custom properties:

1. Open the desired project screen from the **Screens folder**, or insert a new screen. The screen is opened for editing.

2. Open the Symbols Library by doing one of the following:
   - On the Graphics tab of the ribbon, in the Libraries group, click **Symbols**;
   - Double-click **Symbols** in the Project Explorer; or
   - Right-click in the screen where you want to insert the symbol, and then click **Insert Linked Symbol** on the shortcut menu.

3. Select the symbol from the Symbols Library, and then click in the screen:

Once the symbol is inserted, you can manipulate it like any other object in the screen. You can align and distribute it with other objects, and you can apply Animations to it. However, the first thing to do is complete the custom properties for this instance of the symbol.

4. Open the Object Properties for the symbol.
5. Click **Expand** to open the *Symbol Properties* dialog box.

![Symbol Properties dialog box for the symbol](image)

To see all of the properties at the same time, select the **Display properties from all categories** check box.

![Displaying properties from all categories](image)

6. Enter the property values as needed. In this example, the three check boxes are used to determine whether to alert Tom, Dick and/or Harry. The captions are updated accordingly, and the check box tags are configured with the first three elements of a Boolean array called `AlertOptions`.

![Completed properties for the symbol](image)

7. Click **OK** to close the *Symbol Properties* dialog box, and then close the *Object Properties* dialog box. The custom properties are resolved during runtime, as shown below.

![Custom properties](image)
**ACTIVEX CONTROL OBJECT**

On the **Graphics** tab, in the **Libraries** group, click **ActiveX Control** to open the **Insert ActiveX Control** dialog, which you can use to place ActiveX components on your screen.

When the dialog opens (as in the following figure), it contains a list of all ActiveX components that are registered on your PC.

![Insert ActiveX Control dialog](image)

**Note:** When you use ActiveX controls in your project, your runtime stations should have the same controls already installed and registered. Stations often have “auto download” and “auto install” features disabled for security reasons, so they may not be able to get ActiveX controls that are called by your project. Consult your hardware manufacturer and ActiveX controls provider for more information about how to manually install controls.

If you still want to enable automatic download of ActiveX controls, you can do so by manually editing your project file (`project_name.app`) to include the following settings:

```
[UsedControls]
EnableDownload=1
Count=number of controls

[UsedControl1]
CLSID=class ID of the ActiveX control
Version=version of the ActiveX control
Codebase=URL of the ActiveX control file, or of the .CAB file that contains the ActiveX control files
RegFile1=name of a specific .OCX or .DLL file within the .CAB file; see below
RegFile2=name of a specific .OCX or .DLL file within the .CAB file; see below
```

**Note:** Remember, the completed custom properties on each instance of a symbol are independent from every other instance of that symbol, but if you make any changes to the master symbol file, those changes automatically propagate to every instance.
The CLSID and Version settings are required for each ActiveX control, and they must match the ID and version of the actual control file(s) to which Codebase links. This allows a runtime station to check the control against those that are already registered. If the settings do not match, then the runtime station may unnecessarily download the same control again.

If you don't know the CLSID and Version settings for an ActiveX control, you can find them in the registry key of an already installed and registered control. Search for the control file in HKEY_CLASSES_ROOT\CLSID in the Windows Registry.

Also, the URL for the Codebase setting can be either absolute or relative to the Web server's "home" directory. For example:

```
Codebase=http://server_address/AddOns/IndDateTimePick.ocx
```

...or...

```
Codebase=AddOns/IndDateTimePick.ocx
```

Finally, the Regfile settings are required only if Codebase links to a .CAB file. If it does, then use one or more Regfile settings to name the specific files within the .CAB file that must be downloaded and registered.

ActiveX controls are components designed according to a standard. Because InduSoft Web Studio is an ActiveX container, you can configure and run ActiveX controls in the screens created with IWS. ActiveX controls can provide the following interfaces:

- **Properties**: Variables whose values can be read and/or written for your project (e.g., Object Color, FileName, URL, and so forth)
- **Methods**: Functions from the ActiveX object that can be triggered by your project (e.g., open a dialog, execute a calculation, and so forth)
- **Events**: Internal messages that can trigger the execution of expressions in your project (e.g., Mouse_Click, Download_Completed, and so forth)

The name of the properties, methods and events supported by each ActiveX depends on its own implementation.

There are two different ways to interface your project with the ActiveX control:

- By using the ActiveX functions XGet(), XSet() and XRun()
- OR

- By using the Object Properties window to configure the object

Double-click on the ActiveX Control to open the Object Properties dialog.

The Object Properties window displays the name of the ActiveX control. Generally, each ActiveX control is either a *.dll or a *.ocx file registered in your local computer. You must assign a name (alias) to the ActiveX

---

[UsedControln]

---

...
control on the Name field (e.g., MyControl). This name is used to reference the object when calling one of the ActiveX functions that are provided in the Built-in Scripting Language.

**Note:** You should not configure two ActiveX controls on the same screen with the same name. For instance, if you insert two “Windows Media Player” ActiveX controls on the same screen, and assign the name MyMP1 to one object (Name field), you cannot assign the same name to the second object on the same screen. You would have to assign the name MyMP2, for example, to the second object.

The **Property Pages** button opens the standard window for configuring the Static Properties (if any). The layout and the options in this dialog depend on the implementation of each ActiveX Control. Use this interface to set properties that should not be changed during runtime (fixed properties).

The **Configuration** button on the **Object Properties** window opens dialogs that allow you to do the following:

- Associate tags to properties of the ActiveX Control;
- Trigger methods from the ActiveX Control based on tag change; and
- Configure scripts, which are executed when Events from the ActiveX Control occur.

The following sections describe how to configure these interfaces.

**Note:** Although the Configuration dialog displays the list of all properties, methods and events, you only have to configure the items that you need for your project.

The screen shots used in the following sections depict the Acrobat 3D Office control. The names of the properties, methods and events vary for each ActiveX control, but the configuration interface is the same. The concepts described here apply to all controls.

**Configuring Properties**

The **Properties** tab provides a grid with the following fields:

**Configuration Dialog – Properties Tab**

- **Property**: Lists all properties available from the ActiveX object, and indicate their types:

<table>
<thead>
<tr>
<th>Property Icon</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td>String</td>
</tr>
</tbody>
</table>

- **Tag/Expression**: The tag configured in this field is associated with the respective property of the ActiveX object. The Action column will define whether the value of this tag will be written to the ActiveX property, or if the value of the ActiveX property will be written to this tag (or both).
**Note:** You can configure an expression in this field if you want to write the result of an expression to the property of the ActiveX object. However, in this case, the value of the property cannot be read back to one tag (unless you use the XGet() function). Therefore, an expression is configured in this field, the Scan field is automatically set to Set.

- **Action**: Defines the direction of the interface between the tag or expression configured in the Tag/Expression field and the ActiveX property, according to the following table:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td>Read the value of the ActiveX property and write it to the tag configured in the Tag/Expression field.</td>
</tr>
<tr>
<td>Set</td>
<td>Write the value from the tag or expression configured in the Tag/Expression field into the ActiveX property.</td>
</tr>
<tr>
<td>Get+Set</td>
<td>Executes both actions (Get and Set). However, when opening a screen with the ActiveX object, IWS executes the Get command before executing any Set command. That is, the tag configured in the Tag/Expression field is updated with the value of the ActiveX property when IWS opens the screen where the ActiveX is configured.</td>
</tr>
<tr>
<td>Set+Get</td>
<td>Executes both actions (Get and Set). However, when opening a screen with the ActiveX object, IWS executes the Set command before executing any Get command. That is, the ActiveX property is updated with the value of the tag configured in the Tag/Expression field when IWS opens the screen where the ActiveX is configured.</td>
</tr>
</tbody>
</table>

**Note:** When the value of the property is "Read-only" (cannot be overwritten by your project), the Action field is automatically set to Get.

- **Scan**: Defines the polling method to get values from the ActiveX properties, according to the following table:

<table>
<thead>
<tr>
<th>Scan</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>The value of the ActiveX property is read and written to the tag configured in the Tag/Expression field, only when the screen with the ActiveX object is open, and when the ActiveX object sends a message to IWS to update this tag.</td>
</tr>
<tr>
<td>Always</td>
<td>IWS keeps polling the value of the ActiveX property and updating the tag configured in the Tag/Expression field with this value.</td>
</tr>
</tbody>
</table>

**Note:** Some ActiveX controls are designed to send messages to their containers (e.g., your project) indicating that a property changed value and the new value should be read (Get) again. However, other ActiveX controls do not implement this algorithm. In this case, the only way to get the updated values of the ActiveX properties is to keep polling these values from the ActiveX control (Scan=Always).

**Configuring Methods**

The Methods tab provides a grid with the following fields:

- **Method**: List all methods available from the ActiveX object.
• **Parameters**: The tags configured in this field are associated with the parameters of the method of the corresponding ActiveX object. If the method does not support any parameter, the fixed text <None> is displayed in the Parameters field. Otherwise, you can type the tags associated in the parameters of the ActiveX object. When the method has more than one parameter, you can type one tag for each parameter, separating them by a comma (,). For example, TagA, TagB, TagC. When the method is executed, either the value of the tags are written to the parameters of the method (input parameters), or, after the method is executed, the ActiveX writes the value of the parameters to the tags (output parameters).

  **Tip**: When you click the Browse button ( ), it will display the list of parameters supported by the method, allowing you to associate one tag with each parameter.

• **Trigger**: When the tag configured in this field changes value, the respective method of the ActiveX control is executed.

• **Return**: The tag configured in this field receives the value returned by the method (if any).

**Configuring Events**

The Events tab provides a grid with the following fields:

- **Event**: List all events available from the ActiveX object.
- **Parameters**: The tags configured in this field are associated with the parameters of the event of the corresponding ActiveX object. If the event does not support any parameter, the fixed text <None> is displayed in the Parameters field. Otherwise, you can type the tags associated with the parameters of the ActiveX object. When the event has more than one parameter, you can type one tag for each parameter, separating them by a comma (,). For example, TagA, TagB, TagC. When the event is generated, either the value of the tags are written to the parameters of the event (input parameters), or the parameter values are written to the tags (output parameters).

  **Tip**: When you click the Browse button ( ), it will display the list of parameters supported by the event, allowing you to associate one tag with each parameter.

- **Script**: The script configured in this field will be executed when the event is triggered by the ActiveX control.

  **Tip**: When you click the Browse button ( ), it will display a dialog with the complete script associated with the event. The main dialog displays only the expression configured in the first line of the script.

**.NET Control Object**

.NET Components are designed according to the Microsoft .NET Framework, which is a standard for modular programming technologies. Because IWS is a .NET container, you can configure and run .NET Components in your project screens. The actual functions of a .NET Component are contained within a .NET Control object, which provides the configuration dialogs.

.NET Components include the following interfaces:
- **Properties**: Variables whose values can be read and/or written for your project (e.g., Object Color, FileName, URL, and so forth)

- **Methods**: Functions from the .NET Component that can be triggered by your project (e.g., open a dialog, execute a calculation, and so forth)

- **Events**: Internal messages that can trigger the execution of expressions in your project (e.g., Mouse_Click, Download_Completed, and so forth)

The properties, methods and events supported by each .NET Component vary according to the component's implementation.

⚠️ **Caution**: When using .NET Components in your project, make sure that the target system (runtime station) can support the same components and that they are properly installed and registered. Your project includes links to the .NET Components; however, the installation of these components on the target system must be done separately. Furthermore, when .NET Components are used on screens open in remote Thin Clients, the .NET Components must also be manually installed on the Thin Client stations. The Microsoft Windows operating system installs a large selection of components by default, but additional components are offered by third-party providers. Consult your .NET Component provider for further information about how to install.

**Selecting and Placing a .NET Control Object**

To select and place a .NET Control object in your project screen:

1. On the **Graphics** tab, in the **Libraries** group, click **.NET Control**.

2. When the **.NET Framework Components** dialog opens (as in the following figure), it contains a list of all .NET Components that are registered on your computer.

![.NET Framework Components dialog](image)

3. Select a component from the list, and then click **OK** to place it in your project screen. You can also click the **Browse...** button to find an unregistered component on your computer.

**Tip**: Registered .NET Components are typically stored in the following directory:

`C:\WINDOWS\Microsoft.NET\Framework\`

However, you can have the application include unregistered components in the **.NET Framework Components** dialog by editing the `project_name.APP` file to add this parameter:

```
[Execution Environment]
DotNetControlPath=OptionalPath
```

For example:

```
[Execution Environment]
DotNetControlPath=OptionalPath
```
DotNetControlPath=C:\DOTNET CONTROLS BACKUP

Thereafter, the .NET Framework Components dialog will list all registered components and all components found in the specified directory.

4. By default, a new .NET Control object is placed in the upper-left corner of your project screen. Click on the object and drag it to where you want it placed.

5. Once the object is placed, double-click on it to open its Object Properties dialog.

Object Properties: .NET Control

The Object Properties dialog shows the name of the .NET Component. You must assign a name (alias) to the component in the Name box (e.g., CheckBox1). This name is used to reference the component when using the scripting languages (VBScript and built-in scripting).

Note: You should not configure two .NET Control objects on the same screen with the same name. For instance, if you place two CheckBox components on the same screen and assign the name CheckBox1 to one object (Name field), you cannot assign the same name to the second object on the same screen. You would have to assign the name CheckBox2, for example, to the second object.

The Property Pages button opens the standard window for configuring the Static Properties (if any). The layout and the options in this dialog depend on the implementation of each .NET Component. Use this interface to set properties that should not be changed during runtime (fixed properties).

The Members button on the Object Properties dialog opens additional dialogs that allow you to do the following:

• Associate tags to properties of the .NET Component
• Trigger methods from the .NET Component based on tag change
• Configure scripts, which are executed when Events from the .NET Component occur

The following sections describe how to configure these interfaces.

Note: Although the Members dialog displays the list of all properties, methods and events, you only have to configure the items that you need for your project.

The screen shots used in the following sections depict the CheckBox component. Although the names of properties, methods and events vary by component, the configuration interface is the same for any .NET Component. The concepts described here apply to all of them.
Configuring Properties

The Properties tab provides a grid with the following fields:

- **Property**: List all properties available from the .NET Component, and indicate their types:

<table>
<thead>
<tr>
<th>Property Icon</th>
<th>Property Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Integer</td>
</tr>
<tr>
<td>Real</td>
<td>String</td>
</tr>
</tbody>
</table>

- **Tag/Expression**: The tag configured in this field is associated with the respective property of the .NET Component. The Action column will define whether the value of this tag will be written to the property, or if the value of the property will be written to this tag (or both).

- **Action**: Defines the direction of the interface between the tag or expression configured in the Tag/Expression field and the .NET property, according to the following table:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td>Read the value of the property and write it to the tag configured in the Tag/Expression field.</td>
</tr>
<tr>
<td>Set</td>
<td>Write the value from the tag or expression configured in the Tag/Expression field into the property.</td>
</tr>
<tr>
<td>Get+Set</td>
<td>Executes both actions (Get and Set). However, when opening a screen with the .NET Component, IWS executes the Get command before executing any Set command. That is, the tag configured in the Tag/Expression field is updated with the value of the property when IWS opens the screen where the .NET Component is configured.</td>
</tr>
<tr>
<td>Set+Get</td>
<td>Executes both actions (Get and Set). However, when opening a screen with the .NET Component, IWS executes the Set command before executing any Get command. That is, the property is updated with the value of the tag configured in the Tag/Expression field when IWS opens the screen where the .NET Component is configured.</td>
</tr>
</tbody>
</table>

- **Scan**: Defines the polling method to get values from the properties. For .NET Components, all properties scan Always by default. That is, IWS keeps polling the value of the property and updating the tag configured in the Tag/Expression field with this value.

*Note:* When the value of the property is "Read-only" (cannot be overwritten by your project), the Action field is automatically set to Get.
Configuring Methods

The *Methods* tab provides a grid with the following fields:

- **Method**: Lists all methods available from the .NET Component.
- **Parameters**: The tags configured in this field are associated with the corresponding method. If the method does not support any parameter, then the fixed text `<None>` is displayed. Otherwise, you can enter the tags that you want to associate with the parameter. When the method has more than one parameter, you can enter one tag for each parameter, separating them by a comma (,). For example, TagA, TagB, TagC.

  **Tip**: When you click the Browse button ( ), it will display the list of parameters supported by the method, allowing you to associate one tag with each parameter.

When the method is executed, either the value of the tags are written to the parameters of the method (input parameters), or, after the method is executed, the .NET Component writes the value of the parameters to the tags (output parameters).

- **Trigger**: When the tag configured in this field changes value, the respective method of the .NET Component is executed.
- **Return**: The tag configured in this field receives the value returned by the method (if any).

Configuring Events

The *Events* tab provides a grid with the following fields:

- **Event**: Lists all events available from the .NET Component.
- **Parameters**: The tags configured in this field are associated with the corresponding event. If the event does not support any parameter, then the fixed text `<None>` is displayed. Otherwise, you can enter the tags that
you want to associate with the parameter. When the event has more than one parameter, you can enter one tag for each parameter, separating them by a comma (,). For example, TagA, TagB, TagC.

**Tip:** When you click the Browse button ( ), it will display the list of parameters supported by the event, allowing you to associate one tag with each parameter.

When the event occurs, either the value of the tags are written to the parameters of the method (input parameters), or, after the event occurs, the .NET Component writes the value of the parameters to the tags (output parameters).

- **Script:** The script configured in this field will be executed when the event is triggered by the .NET Component.

**Tip:** When you click the Browse button ( ), it will display a dialog with the complete script associated with the event. The main dialog displays only the expression configured in the first line of the script.

### LINK TO AN EXTERNAL IMAGE FILE

Use a Linked Picture screen object to link to an external image file, so that you can easily reuse the image in your project and/or change the image during run time.

This task assumes that you have a Screen worksheet open for editing.

Also, you must decide where exactly the image file will be stored:

- If you want the image file to be downloaded with the rest of the project files to the target system, it must be saved in your project folder at: \Documents\InduSoft Web Studio v7.1 Projects \projectname\Web
- If the image file will be located elsewhere on the network or target system, note the complete file path.

To link to an external image file:

1. On the **Graphics** tab of the ribbon, in the **Libraries** group, click **Linked Picture**.
   A standard **Open** dialog box is displayed.
2. Use the dialog box to locate and select the image file, and then click **OK**.

The following image file types are supported in projects running on Windows target systems:

- Windows Bitmap (*.BMP)
- AutoCAD DXF (*.DXF), versions 10 and 11 only
- JPEG (*.JPG, *.JPEG)
- PNG (*.PNG)
- TIFF (*.TIF, *.TIFF)
- Windows Metafile (*.WMF, *.EMF)

The following image file types are supported in projects running on Windows Embedded target systems:

- Exchangeable image file (*.Exif), commonly used in digital cameras
- GIF (*.GIF)
- JPEG (*.JPG)
- PNG (*.PNG)
- TIFF (*.TIFF)
- Windows Bitmap (*.BMP)

**Note:** InduSoft Web Studio v7.1+SP3 uses a new drawing system to display images on Windows Embedded devices. This new system requires that your device is running a recent version of Windows Embedded and that all of the supported image file types are included in the device’s run-time image. If an image is not displayed correctly on your device, there are two possible solutions.
First, you may disable the new drawing system and use the legacy drawing system instead. If you do this, however, only JPG, PNG, and BMP images will be supported in your project. To use the legacy system, open your project file (projectname.APP) in a text editor and then edit the following property:

```plaintext
[Objects]
  DrawingSystemID=<0 for new system, 1 for legacy system>
```

Second, you might need to update the run-time image on your Windows Embedded device in order to include all of the supported image file types. (The run-time image is typically set by the device manufacturer.) That is beyond the scope of this documentation, however; please refer to the technical note *Image Requirements for IWS v7.1 Runtime on Embedded Systems*, which is available on the InduSoft Web Studio installation DVD and the InduSoft website (www.indusoft.com).

The image is added to the worksheet as a Linked Picture screen object.

3. Double-click the screen object.
   The *Object Properties: Linked Picture* dialog box is displayed.

4. In the *Link File* box, examine the link. If the image file is located in the Web folder, the link is a relative file path. If the image file is located elsewhere, the link is an absolute file path.

   You can specify folders within the Web folder. For example, if you type `MyPictures\Picture1.BMP`, the program will look for the image file at `projectname\Web\MyPictures\Picture1.BMP`.

   **Tip:** The file extension is not always required for the link to work. In projects running on Windows target systems, if no file extension is specified, the extension PNG is used by default. To change this, open your project file (projectname.APP) in a text editor and then edit the following property:

   ```plaintext
   [Viewer]
    DefaultLinkedPictureExtension=<image file extension>
   ```

   In projects running on Windows Embedded target systems, if no file extension is specified, the extension BMP is used by default. This cannot be changed.

5. If you want to change the link — and therefore change the picture — during run time, replace the file path with a project tag:
   a) In the *Link File* box, select the file path, and then copy it to the clipboard.
   b) Replace the file path in the *Link File* box with the name of a String tag in curly brackets (e.g., `{MyLinkedPicture}`).
   c) If the tag does not exist, you will be prompted to create it. Make sure that you create it as a String tag.
   d) Set the tag's startup value to be the file path that you copied to the clipboard, either by pasting the file path into the tag's Startup property (in the Project Tags Datasheet View) or by configuring the *Startup Script* to set the tag value when the project is run.
With a properly configured project tag, the link will be refreshed whenever the tag value changes during run time. Keep in mind that the tag value must have the same format as a normal link: a relative file path for a file located in the Web folder, or an absolute file path for a file stored elsewhere on the network or target system.

6. If you want some part of the picture to be transparent to the screen background, select a transparent color:
   a) In the Transparent Color group, select either Color Code or Tracker in the list.
   b) If you selected Color Code, type a tag/expression that will provide the 24-bit color code of the desired transparent color.
      For more information about Windows color codes, see Color Interface and WdColor Enumeration.
   c) If you selected Tracker, click and drag the tracker on the screen object until it is positioned over a sample of the desired transparent color.
      The tracker is an additional handle on the screen object that initially appears just inside the bottom-right corner of the object. Moving the tracker on the object does not move or resize the object itself.

   ![Note: Transparent color is not supported for Windows Metafiles (*.WMF, *.EMF)]

7. Close the Object Properties dialog box.

**Applying animations to screen objects**

Use the Animations group to apply animations to a screen object or group of objects. Animations enable you to modify object properties on the fly (during runtime) according to tag values. Some animations also enable you to execute commands or insert values (set points) to the tags.

![Animations group]

**COMMAND ANIMATION**

On the Graphics tab, in the Animations group, click Command to add the animation to a selected object or group of objects. The animation enables you to click on the object or press a pre-defined key to execute the command at runtime. Double-click on the object to view its object properties.
The Command animation provides one tag for each one of the events supported by it. Notice that more than one event can be configured simultaneously for the same Command animation:

- **On Down**: Executes the command/script once when the user clicks on the object with the left mouse button.
- **On While**: Keeps executing the command/script continuously while the mouse pointer is pressed on the object. The period (in milliseconds) of execution for the command/script is set in the Rate field from the Configuration dialog screen, except for the VBScript option, which is executed as fast as possible.
- **On Up**: Executes the command/script once when the user releases the left mouse button on the object.
- **On Right Down**: Executes the command/script once when the user clicks on the object with the right mouse button.
- **On Right Up**: Executes the command/script once when the user releases the right mouse button on the object.
- **On Double Click**: Executes the command/script once when the user double-clicks on the object with the left mouse button.
- **On Touch, On Touch Start, On Touch Delta, On Touch Complete**: These events are used for multi-touch gestures. For more information, see About Touch Events on page 269.

**Note**: The events **On Right Down**, **On Right Up** and **On Double Click** are not supported in projects running on Windows Embedded target systems.

**Tip**: An asterisk (*) on an event tab indicates that something is configured for that event. This makes it easier to see at a glance which events are configured.

**Type menu**: This setting defines the type of action that must be executed by the event of the Command animation. Notice that each event has its own type. Therefore, the same Command animation can be configured with different types of action for different events. The following types are supported:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in Language</td>
<td>Allows you to configure a script using the IWS built-in language. When this type is selected, the user can configure up to 12 expressions for each event in the <strong>Expression</strong> column. The expressions are executed sequentially from the first row until the last one when the event is triggered. The result of each expression is written to the tag configured in the <strong>Tag</strong> column (if any). Consult the Built-in Scripting Language chapter for more information.</td>
</tr>
<tr>
<td>VBScript</td>
<td>Allows you to configure a script using the standard VBScript language. When this type is selected, the user can configure a script in the VBScript editor for the Command animation. Consult the VBScript chapter for further information about the VBScript language.</td>
</tr>
<tr>
<td>Open Screen</td>
<td>Allows you to configure the Command animation to open a specific screen when the event is triggered during runtime. This type is equivalent to the Open function. You can either type the screen name in the Open Screen field or browse it. Furthermore, you can type a string tag between curly brackets {TagName} in this field. When the event is executed, the project will attempt to open the named screen.</td>
</tr>
<tr>
<td>Close Screen</td>
<td>Allows you to configure the Command animation to close a specific screen when the event is triggered during runtime. This type is equivalent to the Close function. You can either type the screen name in the Close Screen field or browse it. You can also type a string tag between curly brackets {TagName} in this field. When the event is executed, the project will attempt to close the named screen.</td>
</tr>
<tr>
<td>Set Tag</td>
<td>Allows you to configure the Command animation to set a tag when the event is triggered during runtime. You can either type the tag name in the Set Tag field or browse it. When the event is executed, the project will write the value 1 to the tag configured in this field.</td>
</tr>
<tr>
<td>Reset Tag</td>
<td>Allows you to configure the Command animation to reset a tag when the event is triggered during runtime. You can either type the tag name in the Reset Tag field or browse it. When the event is executed, the project will write the value 0 to the tag configured in this field.</td>
</tr>
<tr>
<td>Toggle Tag</td>
<td>Allows you to configure the Command animation to toggle a tag when the event is triggered during runtime. You can either type the tag name in the Toggle Tag field or browse it. When the event is executed, the project will toggle the value of the tag configured in this field.</td>
</tr>
</tbody>
</table>

**Config button**: Launches the Configuration dialog, where the Command animation can be fully configured.

**Back to button**: Click to go back to the object properties of the underlying Button object.
Configuration dialog

This dialog allows you to fully configure the Command animation...

The event tabs (e.g., On Down, On While, etc.) and the Type menu are the same as in the Object Properties dialog described above. The remaining settings are shared for all events:

- **Options pane:**
  - **Enable Focus** checkbox: When this option is checked, the object that the Command animation was applied to can receive the focus during runtime by the navigation keys.
  - **Force** checkbox: When this option is selected, any project tag that receives a value will trigger events as if the tag changed, even if the new value is equal to the old value. For example, if a tag has a value of 0 and the Command animation runs a procedure that writes 0 to that tag, all other tasks in the project runtime will recognize that the tag changed, even though it did not. This option is useful for making sure that events triggered by tag changes (e.g., **Write on Tag Change** on a communication driver) are always triggered when the Command animation is used.

Please keep in mind that if the tag's value does not actually change, the tag's timestamp (tagname->Timestamp) is not updated either.

**Force** applies to both the procedure run by the Command animation itself and any global procedures called in that procedure, as long as they are run on the project runtime client where the Command animation is used (i.e., on the device where the button is pushed).

**Force** does not apply to global procedures that are run on the project runtime server using the function RunGlobalProcedureOnServer, even if the function is called in the procedure run by the Command animation. If you want to force tag changes in global procedures run on the server, use the function ForceTagChange.

**Note:** For projects created with InduSoft Web Studio v6.1+SP3 or earlier, **Force** is enabled by default and cannot be disabled.

- **Beep** checkbox: When this option is checked, a short beep is played when the Command is executed. This option is useful to provide an audio feed-back to the user, indicating that the Command was executed. It does not indicate, however, if the action triggered by the Command animation was successful or not.
• **Release** checkbox: When this option is checked, the On Up event is executed when you drag the cursor (or your finger) out of the object area (whether the button was released or not). This option is useful to make sure that the On Up event will always be executed after an On Down event, even if the user releases the mouse cursor out of the object area before releasing it.

• **Confirm** checkbox: When this option is checked, user will have to answer a confirmation question before executing the command. This option is useful for decreasing the accidental triggering of critical events during runtime.

• **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.

• **Key Only** checkbox: When this option is checked, the user can only use the keyboard shortcut (configured in the **Key** pane described below) to execute commands.

• **Disable**: Disables action by the user when the result of the expression configured in this field is TRUE (value different from 0).

• **Security**: Security access level required to use the Command animation.

• **Key group**: Shortcut used to trigger the events On Down, While Down and On Up using a keyboard. (In other words, pressing this keyboard shortcut is the same as clicking the left mouse button.) This option is especially useful when creating projects for runtime devices that do not provide a mouse or touch-screen interface — the keyboard is the only physical interface available to interact with your project during runtime.

    • **Shift, Ctrl, or Alt** boxes: Click to create a key combination key, meaning the Shift, Ctrl and/or Alt key must be pressed with the key specified in the drop-down list.

    • Click the browse button (…) to open the **Key Modifier** dialog, which enables you to modify your combination keys. You can choose **Left**, **Right** or **Left or Right** to specify the position on the keyboard of the Shift, Ctrl or Alt key in the key combination. If you choose **Left or Right**, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

    **Tip:** If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.

**HYPERLINK ANIMATION**

On the **Graphics** tab, in the **Animations** group, click **Hyperlink** to add the animation to a selected object or group of objects. Applying this animation allows you to click on the object(s) during execution to launch the default browser and load the specified URL.

Double-click on the object to open the **Object Properties** dialog.

![Object Properties: Hyperlink](image)

You can use this dialog to specify the following parameters:

• **Hyperlink Type** combo-box: Click the combo-box button to select a URL protocol from the list. The project uses this protocol when it loads the URL.

• **URL** field: Type the URL address you want to load.
Tip: You are not required to enter the protocol type in the URL field. When you select a protocol type from the Hyperlink Type list, the project automatically adds the protocol's prefix to the URL address.

- **Disable** field: Type a value greater than zero into this field to disable the hyperlink Command animation for the selected object(s).
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.
- **Security** field: Type a value into this field to specify a security level for the object(s). If a user logs on but does not have the required security level, the project disables the hyperlink command for the object(s).

**BARGRAPH ANIMATION**

On the Graphics tab, in the Animations group, click Bargraph to add bar graph properties to a selected object, then double-click on the object to open the Object Properties dialog.

![Object Properties: BarGraph](image)

Use the Object Properties dialog to specify the following parameters:

- **Tag/Expression** field: Type a tag or expression that evaluates the bar graph level. You also can click the icon to browse your directories for an existing tag or expression.

- **Minimum Value** field: Type a numeric constant or a tag value into this field to define the minimum value used to calculate the height (if vertical) or width (if horizontal) of the bars.

- **Maximum Value** field: Type a numeric constant or a tag value into this field to define the maximum value used to calculate the height (if vertical) or width (if horizontal) of the bars.

If you do not specify a value for this field, the application opens a dialog requesting you confirm creation of the tag.

Tip: The application also allows you to enter constants in tag/numeric value fields. Constant values (defined by the # character) are equivalent to numeric values, except that constants display in the Tag Replace dialog. You may find constants useful for documentation purposes or for creating generic objects.

For example: \#Name:100.

Where the value (100) following the semicolon (:) is the constant, and Name is a constant mnemonic only and not added to database.

- **Foreground Color**: To specify a fill color for the bars, click the combo-box button. When the Color dialog displays, click on a color to select it, and then close the dialog.

- **Direction** area: Click the Vertical or Horizontal radio button to specify the direction of the bar graph.

- **Orientation** area: Click the Up, Center, or Down radio button to specify the orientation of the maximum and minimum values when drawing the bars.
TEXT DATA LINK ANIMATION

On the Graphics tab, in the Animations group, click Text Data Link to add the animation input or output text property to a selected Text object. Applying the Text Data Link property allows you to insert and display tag values in real time if you are using the keyboard or on-screen keypad to run a project.

Note: You can only apply this animation to Text objects that include one or more # characters. Each # represents one character of input/output. You can combine # characters with regular text in the same Text object — for example, MyLabel #### or $###.##.

It's important to remember that the runtime project will always display the most significant digits of a numeric value, regardless of the number or placement of # characters in the text. That means if you do not have sufficient # characters to display the value, then it will be transformed in some way depending on the format of the value (as set by the Fmt option described below):

- In Decimal format, the number of decimal places is determined by the position of the decimal separator in the ### text. However, if you do not have enough # characters to the left of the decimal separator to display the whole value, then the whole value will overrun the fractional value. For example, if you try to display a value of 112.64 in .##, you will see 112.
- In Hexa and Binary formats, if you have more # characters than you need to display the value, then the runtime project will fill in with leading zeroes. If you have less characters than you need, then the value will simply be truncated.
- In Auto format, the runtime project will ignore the number of # characters and display the entire numeric or string value. Numeric values will be displayed in decimal format with their complete whole and fractional values, regardless of the placement of the decimal separator in the ### text. Given an exceptionally large value or long string, this may disrupt the layout of your screens.

Double-click on the object to open the Object Properties dialog. You can use this dialog to specify the following parameters:

Object Properties: Text Data Link

- Tag/Expression text field: Type one of the following into the field:
  - The name of a tag on which to perform an input or output operation; or
  - An expression on which to perform an output operation only.

You can also click the browse button ... to open the Object Finder to find an existing tag or expression.

Note: If the configured tag/expression is invalid, then during runtime, the placeholder characters (###) will be displayed instead.

- Format combo-box: Click to select how the numeric value (if any) of the specified tag or expression will be formatted and displayed on-screen. Available options include Decimal, Hexa (i.e., hexadecimal), Binary and Auto. If you select Auto, then the value will be formatted according to the virtual table created by the SetDecimalPoints function.

This option does not actually change the specified tag or expression in any way. For example, Tag/Expression is set to a tag of Integer type, Input Enabled is checked, and Fmt is set to Hexa. You may input a new value in hexadecimal format, but it is saved in your project database as an integer.

- Input Enabled checkbox: Click (check) this option to allow user input to the specified tag. Disable (uncheck) this option to only display the output from the specified tag or expression.
• **Back to text**: Click to go back to the object properties of the underlying Text object.

• **Minimum Value** field: Enter a minimum value for the tag associated with this Text object. A user will not be permitted to input a number lower than this value.

• **Maximum Value** field: Enter a maximum value for the tag associated with this Text object. A user will not be permitted to input a number greater than this value.

• **Password** checkbox: Click (check) this option to hide password text entries by replacing the text with asterisks (*).

• **Confirm** checkbox: Click (check) this option to require users to confirm any new values set during runtime.

• **Auto Size** checkbox: Click (check) this option to automatically resize the Text object to fit the output. This option is not available if **Input Enabled** is checked (see above).

• **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before changing the tag value.

• **VK**: Virtual Keyboard type used for this object. You need to select the Virtual Keyboard option in the **Viewer** settings (**Viewer** on the Project tab of the ribbon) before configuring the Virtual Keyboard for this interface.

• **Disable** field: Type a value greater than zero in this field to disable the tag’s data input property.

• **Security** field: Type a value in this field to specify the security level for a specific data input object (as defined in the **Security** section).

**COLOR ANIMATION**

On the **Graphics** tab, in the **Animations** group, click **Color** to add the animation to a selected object. The Colors animation allows you to modify the color of a static object during runtime based on the value of a tag or expression.

Double-click on the object to open the Object Properties dialog.

![Object Properties: Colors](image)

You can use this dialog to specify the following parameters:

• **Type**: Determines the mode in which this animation works:
  
  • **By Limit**: When selecting this type, you can specify up to four limits (Change Limit) for this animation and a color for each limit. When the value of the tag or expression configured in the Tag/Expr field reaches the limits, the color associated with the respective limit is applied to the object.

  • **By Color**: When selecting this type, you can specify the code of the color that must be applied to the object directly in the Tag/Expr field. Using this code, you can apply any color supported by your device to the object.

  **Tip**: You can configure the **RGBColor** function in the Tag/Expr field when **Type** = **By Color**. This allows you to configure the color by its RGB codes. See **Color Interface** for a table with the codes for the most commonly used colors.

• **Tag/Expression** field: Type the name of a tag or expression you want to monitor. When **Type** = **By Limit**, IWS compares the result of the tag/expression with the specified Change Limits to determine the proper color for the selected object. When **Type** = **By Color**, the result of this field sets the color that will be applied to the object.
• **Change Limit** field: Type a limit value (a numeric constant or tag) for the color change. The numbers must be configured in ascending order according to the following sequence of the fields displayed on the Object Properties dialog: Upper left, lower left, upper right, and lower right field. If you click on the More button, you can configure up to 16 different limits for the color animation.

![Color Limits dialog](image)

- **Color** combo box: Click the combo-box button to associate a color with each color change limit. When the Color dialog opens, click a color to select it, and then close the dialog.
- **Blink** combo-box: Click the combo-box button to specify whether the color change will blink, and how fast it will do so.

**Note:** The following fields are automatically disabled (grayed out) when *Type* = By Color: **Change Limit**, **Color** and **Blink**.

**VISIBILITY/POSITION ANIMATION**

The Visibility/Position animation allows you to move an object horizontally and/or vertically during run time. On the **Graphics** tab, in the **Animations** group, click **Visibility/Position** to add the animation to an object. Double-click on the object to open its **Object Properties** dialog.

![Object Properties: Visibility/Position](image)

Use the dialog to configure the following properties:

- **Visibility** box: Configure a tag/expression in this box to control the visibility of the object. When the value of the tag/expression is 0 (FALSE), the object is hidden, and when the value is non-zero (TRUE) or the box is left empty, the object is visible.
Objects that are hidden cannot be clicked/tapped and therefore cannot execute any Command animations applied to them.

For some types of screen objects (i.e., all Shapes, Standard-style Buttons, and Linked Pictures), Visibility controls not just the visibility but the opacity of the object, and the value (from 0 to 1) of the tag/ expression determines the percentage of opacity. For example, a value of 0.8 would give the object 80% opacity. The value can change during run time, so you can use it to make objects appear to fade in and out. Please note that you must have Enable Enhanced Graphics selected in the project settings in order to use this feature. For more information, see Project Settings: Viewer.

• **Horizontal** and **Vertical**: Configure these settings to determine how the object moves in the screen:
  
  • **Tag/Expression** boxes: Configure a tag/expression that will determine the position of the object during run time; as the value changes, the object is moved in the screen. Whether you can configure a tag or an expression depends on whether the Slider/Gestures option (see below) is selected:
    
    • If the Slider/Gestures option is not selected, then configure either a tag or an expression in this box.
    
    • If the Slider/Gestures option is selected, then configure only a project tag (Integer or Real type) in this box. When the end user manually moves the object, the new value is written back to the tag.

  For the horizontal position, the value increases as the object moves to the right and it decreases at the object moves to the left. For the vertical position, the value increases as the object moves to the bottom and it decreases as the object moves to the top.

  • **Value range** boxes: Enter the minimum and maximum values for the tag/expression. If the actual value goes outside of its range, then the value is ignored and the limit is used instead.

  • **Position** boxes: Enter values to specify how far (in pixels) the object can move from its starting position. The starting position is equal to “0,0”. Values greater than 0 allow the object to move right and down, and values less than 0 allow the object to move left and up.

  During run time, the object’s position is proportional to the tag/expression value within its range. For example, if Position is 0 to 100 and Range is 0 to 10, then each increment in the value will move the object 10 pixels. This is true for both Horizontal and Vertical.

  • **Reference** drop-down lists: Select a reference point on the object. The following table shows how your selections for Horizontal and Vertical work in combination:

<table>
<thead>
<tr>
<th></th>
<th>LEFT</th>
<th>CENTER</th>
<th>RIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
</tr>
<tr>
<td>CENTER</td>
<td><img src="image4" alt="Image" /></td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
</tr>
<tr>
<td>BOTTOM</td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
<td><img src="image9" alt="Image" /></td>
</tr>
</tbody>
</table>

This reference point is meaningful only if you have the **Resize animation** added to the same object. The position of the object is always based on this reference point, regardless of the size or shape of the object.

• **Slider/Gestures**: When multi-touch gestures are enabled, the end user can use one- or two-finger "slide" gestures to move this object during run time. The changes in position are written back to the project tags configured in the Tag/Expression boxes above.

  • **Enable** option: Select to enable gestures on this specific object.

  Please note that Multi-Touch must also be enabled for the project and screen.

  • **Inertia** option: Select to apply inertia to this object, so that it slows down naturally rather than stops abruptly when the end user stops touching it.
• **Disable** box: Configure a tag/expression. When its value is TRUE (i.e., not 0), then gestures are disabled on this object.

• **Security** box: Type the minimum security level that the end user must have to use gestures on this object.

For more information, see *About using Multi-Touch in your project* on page 257.

**RESIZE ANIMATION**

The Resize animation allows you to increase or decrease the size of an object during runtime.

On the **Graphics** tab, in the **Animations** group, click **Resize** to add the animation to an object. Double-click on the object to open its **Object Properties** dialog.

![Object Properties: Resize](image)

Use the dialog to configure the following properties:

• **Height** and **Width**: Configure these settings to determine how the object moves in the screen:
  
  • **Tag/Expression** boxes: Configure a tag/expression that will determine the size of the object during runtime; as the value changes, the object is resized in the screen. Whether you can configure a tag or an expression depends on whether the **Gesture** option (see below) is selected:
    
    • If the **Gesture** option is not selected, then configure either a tag or an expression in this box.
    
    • If the **Gesture** option is selected, then configure only a project tag (Integer or Real type) in this box. When the end user manually resizes the object, the new value is written back to the tag.
  
  • **Value range** boxes: Enter the minimum and maximum values for the specified tag(s). If a tag’s actual value goes outside of its range, then the value is ignored and the limit is used instead.
  
  • **Size range (%)** boxes: Enter the minimum and maximum values for the size of the object. The minimum value can be as low as 0% (making the object effectively invisible), and the maximum value can be as high as you want. 100% is the original size of the object when you draw it in the screen worksheet, 200% is double the original size, and so on.

  During run time, the object’s size is proportional to the tag value within its range. For example, if **Size range (%)** is 0 to 100 and **Value range** is 0 to 10, then each increment in the value will increase the object size by 10%. This is true for both Height and Width.

• **Reference** drop-down lists: Select a reference point to determine the directions in which the object will change size. The following table shows how your selections for Height and Width work in combination:

<table>
<thead>
<tr>
<th></th>
<th>LEFT</th>
<th>CENTER</th>
<th>RIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td><img src="image" alt="image" /></td>
<td><img src="image" alt="image" /></td>
<td><img src="image" alt="image" /></td>
</tr>
<tr>
<td>CENTER</td>
<td><img src="image" alt="image" /></td>
<td><img src="image" alt="image" /></td>
<td><img src="image" alt="image" /></td>
</tr>
</tbody>
</table>
• **Gesture**: When multi-touch gestures are enabled, the end user may use two-finger “pinch” and “stretch”
gestures to resize this object during run time. The changes in size are written back to the project tags
configured in the **Tag/Expression** boxes above.

  • **Enable** option: Select to enable gestures on this specific object.

  Please note that Multi-Touch must also be enabled for the project and screen.

  • **Inertia** option: Select to apply inertia to this object, so that it slows down naturally rather than stops
  abruptly when the end user stops touching it.

  • **Disable** box: Configure a tag/expression. When its value is TRUE (i.e., not 0), then gestures are disabled
  on this object.

  • **Security** box: Type the minimum **security level** that the end user must have to use gestures on this
  object.

  For more information, see About using Multi-Touch in your project on page 257.

**ROTATION ANIMATION**

Use the Rotation animation to rotate screen objects.

On the **Graphics** tab, in the **Animations** group, click **Rotation** to add the animation to a **Line**, **Open Polygon**, **Closed Polygon**, **Bitmap**, or **Linked Picture** object.

**Note**: The Rotation animation does not work in a group of objects. If the animation is added to an
object and then that object is grouped with others, it will be disabled.

Also, the Rotation animation cannot be added to Bitmap and Linked Picture objects in projects
running on Windows Embedded devices. For more information, see About target platforms and
product types on page 91.

After the animation is added, double-click the object to open the **Object Properties** dialog box.

Use this dialog box to edit the following properties:

• **Tag/Expression** box: Specify a tag/expression that will determine the angle of the object during run time;
as the value changes, the object is rotated in the screen. Whether you can specify a tag or an expression
depends on whether the **Gesture** option (see below) is selected:

  • If the **Gesture** option is not selected, specify either a tag or an expression in this box.

  • If the **Gesture** option is selected, specify only a project tag (Integer or Real type) in this box. When the end
  user manually rotates the object, the new value is written back to the tag.

  • **Range area**: Enter the **Minimum** and **Maximum** values allowed for **Tag/Expression**. Values less than the minimum
  and greater than the maximum are disregarded.
• **Rotation (degrees)** area: Enter the **Start** and **End** positions (in degrees) of the object. The actual rotation is proportional to the value of **Tag/Expression** within **Range**. An object can rotate up to 360 degrees, and it rotates clockwise by default.

  **Tip:** For example, a Rotation animation has the following settings: **Minimum** is 0, **Maximum** is 100, **Start** is 0, and **End** is 180. If the current value of **Tag/Expression** is 50 (i.e., halfway between **Minimum** and **Maximum**), then the actual rotation of the object is 90 degrees (i.e., halfway between **Start** and **End**). A value of 25 is equal to 45 degrees, a value of 75 is equal to 135 degrees, and so on.

• **Reference** combo-box: Select one of the following as a pivot point on which to rotate the object:
  - **Left-Top**: Upper-left corner of the object.
  - **Left-Bottom**: Lower-left corner of the object.
  - **Center**: Center of the object.
  - **Right-Top**: Upper-right corner of the object.
  - **Right-Bottom**: Lower-right corner of the object.

  You can fine tune the pivot point by configuring the **Offset** settings described below.

• **Advanced** button: Click to open the **Advanced** dialog box, where you can edit the following settings:

  ![Object Properties: Rotation – Advanced](image)

  - **Offset (pixels)** area: Enter the number of pixels by which to offset the **Reference** (i.e., pivot point) on the **X axis** and/or **Y axis**.
  - **Counter Clockwise** checkbox: Click (enable) this option to make the object rotate counterclockwise instead of clockwise.
  - **Gesture**: When multi-touch gestures are enabled, the end user may use two-finger "turn" gestures to rotate this object during run time. The changes in angle are written back to the project tag specified in the **Tag/Expression** box above.
    - **Enable** option: Select to enable gestures on this specific object.
      Please note that Multi-Touch must also be enabled for the project and screen.
    - **Inertia** option: Select to apply inertia to this object, so that it slows down naturally rather than stops abruptly when the end user stops touching it.
    - **Disable** box: Specify a tag/expression. When its value is TRUE (i.e., non-zero), gestures are disabled on this object.
    - **Security** box: Type the minimum **security level** that the end user must have to use gestures on this object.

For more information, see **About using Multi-Touch in your project** on page 257.
Format tab

The Format tab of the ribbon is used to format and arrange objects in a project screen.

![Format tab of the ribbon]

**Note:** This tab is available only when you've selected one or more objects in a project screen.

The tools are organized into the following groups:

- **Arrange:** Arrange objects in a project screen, including bring to front and send to back, group, align, and rotate.
- **Position:** Precisely adjust the position of a screen object in a project screen.
- **Size:** Precisely adjust the size of a screen object.
- **Style:** Change the fill and line color of a screen object.
- **Fonts:** Change the caption font of a screen object.

**Change the properties of multiple screen objects**

This task describes how to select two or more screen objects and then change the properties that are common to the selected objects.

Before you begin this task, you must have a project screen open in the screen editor.

Which properties you can change depends on whether you select multiple objects of the same type or of different types. If the objects are of the same type, you can change the properties that are specific to that type. For example, if you select multiple Button objects, then you can change the properties that are specific to Button objects.

![Object properties of multiple selected Button objects]

For more information about the properties of a specific type of object, see the documentation for that object.

**Note:** You can only use this method to change the properties of Shapes and Active Objects. You cannot use this method to change the properties of Data Objects, Animations, Library items, or objects in a group.

In contrast, if you select multiple objects of different types, you can change the properties that are common to all of the objects. This includes not only cosmetic properties like Border and Background, but also functional
properties like Disable, Security, Enable Translation, and E-Sign. (Some properties may not apply to all objects. For example, Button objects do not have Border and Rectangle objects do not have Security.)

In both cases, the dialog box shows the current values of the properties of the last selected object. It is only when you actually change the value of a property that the change is applied to the selected objects. All other properties are left unchanged, regardless the values shown in the dialog box.

To change the properties of multiple screen objects:
1. In the screen editor, do one of the following:
   • Press and hold either Shift or Ctrl on the keyboard, and then click each object that you want to change; or
   • Use the cursor to draw a selection box around all of the objects that you want to change.
   The objects are selected.
2. Do one of the following:
   • On the Graphics tab of the ribbon, in the Editing group, click Properties;
   • Right-click the selected objects, and then on the shortcut menu, click Properties; or
   • Press Alt+Enter on the keyboard.

   **Note:** You cannot double-click to open the Object Properties dialog box as you otherwise would, because clicking like that clears the selection.

   The Object Properties dialog box is displayed for the selected objects.
3. Change the property values that you want to change, and then close the dialog box.
   The changes are applied to all of the selected objects.
Set the tab order of screen objects

Set the tab order of screen objects to make a screen easier to use from a physical keyboard/keypad rather than from a mouse or touchscreen.

When you press Tab on the keyboard/keypad during project run time, the focus moves to the next object or field on the screen. (This is also known as an object becoming active.) Focus makes it possible to interact with that object or field using only the keyboard/keypad. For example:

- When the focus is on a button, you may press Return to click/tap that button;
- When the focus is on a text box, you may type a value into that box and then press Return to enter the value; and
- When the focus is on list or menu, you may use the arrow keys to navigate the menu and then press Return to make a selection.

If you repeatedly press Tab, then the focus will move through all of the objects in a screen according to the screen’s tab order. By default, the tab order is the same as the layer order, starting with the layer farthest back (ID: 0) and proceeding to the front (ID: n). However, you can draw objects anywhere, rearrange them, and adjust their layers as you develop the screen, so the default tab order of a finished screen may seem to jump around at random rather than move from left to right and/or top to bottom as the user would expect.

You can set the tab order to make it move through the screen exactly how you want it to, independent of the layer order.

To set the tab order:

1. Open a screen for editing.
2. Right-click on the screen background, and then click Tab Order on the shortcut menu.

   **Note:** If Tab Order is not available on the shortcut menu, it may be because you actually right-clicked on a screen object or you have a screen object selected. Make sure you have no screen object selected and you are right-clicking on the screen background.

The screen enters Tab Order mode, with the order number of each screen object shown in an orange badge at the top-right corner of the object.

![Example of Tab Order mode](image)

A group of objects has a single order number for the entire group (e.g., 4), and then the objects within the group have their own sub-order (e.g., 4.1 and 4.2). This continues as deeply as necessary to accommodate
nested groups. If you do not see the sub-order numbers, right-click again in the screen and then click Expand All Groups on the shortcut menu.

3. To quickly set the tab order for all objects, simply click on the objects in the order that you want. The order numbers will update as you click on the objects.

4. To set the order number for a specific object:
   a) Double-click on the object’s badge.
      The badge changes to a text input box.
   b) Type the order number for that object. Be sure to included any necessary sub-orders.
   c) Press Return to apply the change.

   **Note:** You cannot change the sub-order of objects within a Linked Symbol, because it is only a copy of a Master Symbol that is shared across the entire project. Instead, you must edit the Master Symbol itself. For more information, see Save your own project symbols on page 215.

5. When you have finished setting the tab order and want to exit Tab Order mode, right-click on the screen and then click **Tab Order** on the shortcut menu.

**Bring to front / Send to back**

Bring a screen object to the front or send it to the back of other overlapping objects, as part of arranging the objects on the project screen.

Before you begin this task, you must have a project screen open for editing and the screen must have two or more objects already on it.

**Note:** In this section, "object" refers to both individual screen objects and object groups, but it does not refer to multiple objects that have been selected together but not grouped.

Screen objects are automatically assigned ID numbers, starting with ID 0, as you add the objects to a project screen. (To check the ID number of an object, simply select it. The ID number is displayed on the status bar at the bottom of the development application window.) The object with ID 0 is the furthest back, behind all other objects on the screen, and each additional object is displayed in front of it.

In other words, objects with higher ID numbers are displayed in front of objects with lower ID numbers.

On a finished screen, you will probably have objects arranged in overlapping layers, and you may want rearrange the objects so that they are displayed correctly. You can bring an object to the front, so that it is displayed in front of all other objects, or you can send an object to the back, so that it is displayed behind all other objects. You can also move an object forward or backward one layer at a time, if necessary.

![Diagram of Bringing the red circle to the front](image)

**Bringing the red circle to the front**

![Diagram of Sending the blue rectangle to the back](image)

**Sending the blue rectangle to the back**

When you rearrange objects, their ID numbers are automatically adjusted to reflect their new order. For example, if you have four objects on a screen and you send the frontmost object (ID 3) to the back, then that object becomes ID 0 and the ID numbers of the other three objects are increased accordingly.
Please note that windowed objects — that is, screen objects that include some kind of window displaying other data — are handled differently than shapes (e.g., Line, Rectangle) and simple active objects (e.g., Button, Check Box). The windowed objects on a screen will always be in front; you cannot send them behind the other types of objects. This is to ensure that the windowed objects will be rendered properly during project run time.

Windowed objects include:
- Text Box object
- Combo Box object
- Alarm/Event Control object
- Trend Control object
- Grid object
- Most ActiveX and .Net Control objects

If you have more than one windowed object on the same screen, then you can arrange those objects relative to each other. For example, you can send a Grid object behind a Trend Control object. Both windowed objects will still be in front of the other types of objects, however, and in most cases, you should not have overlapping windowed objects anyhow.

To bring a screen object to the front or send it to the back:

1. Select the screen object that you want to move forward or backward. "Handles" are displayed at the object’s corners to show that it is selected.

2. To bring the object all the way to the front (i.e., increase it to the highest ID number), do one of the following:
   - On the Format tab of the ribbon, in the Arrange group, click Bring to Front; or
   - Right-click the object itself, and then click Bring to Front on the shortcut menu.

3. To bring the object one layer forward (i.e., to increase its ID number by 1), do one of the following:
   - On the Format tab of the ribbon, in the Arrange group, click and hold Bring to Front until the shortcut menu appears, and then click Bring Forward on the shortcut menu; or
   - Right-click the object itself, and then click Bring Forward on the shortcut menu.

4. To send the object all the way to the back (i.e., decrease it to ID 0), do one of the following:
   - On the Format tab of the ribbon, in the Arrange group, click Send to Back; or
   - Right-click the object itself, and then click Send to Back on the shortcut menu.

5. To send the object one layer backward (i.e., to decrease its ID number by 1), do one of the following:
   - On the Format tab of the ribbon, in the Arrange group, click and hold Send to Back until the shortcut menu appears, and then click Send Backward on the shortcut menu; or
   - Right-click the object itself, and then click Send Backward on the shortcut menu.

**Group and ungroup screen objects**

Use the Group command to group screen objects together.

A group of objects can be selected, moved, copied, and in some cases modified as if it is a single object, which makes it easier to manage in a busy project screen. Moreover, a group can be grouped with other groups to create increasingly complex groups.

To group two or more screen objects, select the objects that you want to group and then do one of the following:
- On the Format tab of the ribbon, in the Arrange group, click Group; or
• Right-click the selected objects, and then click **Group** on the shortcut menu.

![Grouping the selected objects](image)

**Tip:** If the **Format** tab of the ribbon is not visible, or if the **Group** command is not available on the shortcut menu, it is because you have not selected any objects in the screen editor.

To edit the object properties on a group, use the **Object Properties** dialog box just as you would on a single object. A group has more than one set of object properties, however, like an object that has animations added to it. As such, use the list in the top-right corner of the dialog box to select each set of object properties.

![Selecting a set of object properties in a group](image)

Once you have created a group — also called a symbol — you may choose to save a master of it in your Symbols Library and then reuse it elsewhere in your project. Each copy will be linked to the master so that if you change the master, all of the linked copies will also be changed. For more information, see Save your own project symbols on page 215.

To ungroup a group of objects, select the group and then do one of the following:

• On the **Format** tab of the ribbon, in the **Arrange** group, click and hold **Group** to access the menu, and then click **Ungroup** on the menu; or

• Right-click the selected group, and then click **Ungroup** on the shortcut menu.

**Tip:** A complex group can comprise several subgroups, so to ungroup it completely, first ungroup the group and then ungroup the subgroups.

### Align, Center and Distribute Tools

When you select a series of objects (two or more), you can align those objects based on the location of the last object selected. As you select objects, solid handles display on the last object selected, and the handles on all previously selected objects become empty (unfilled) boxes.
Use the following alignment tools to align a series of objects.

Click the **Align left** tool to align all selected objects to the left edge of the last object selected. For an example, see the following figure:

![Aligning Objects Left](image)

Click the **Align right** tool to align all selected objects to the right edge of the last object selected. For an example, see the following figure:

![Aligning Objects Right](image)

Click the **Align top** tool to align all selected objects to the top edge of the last object selected. For an example, see the following figure:

![Aligning Object Tops](image)

Click the **Align bottom** tool to align all selected objects to the bottom edge of the last object selected. For an example, see the following figure:

![Aligning Object Bottoms](image)
Click the **Center Vertically** tool to align all selected objects to the vertical center of the last object selected. For an example, see the following figure:

![Centering Objects Vertically](image)

Click the **Center Horizontally** tool to align all selected objects to the horizontal center of the last object selected. For an example, see the following figure:

![Centering Objects Horizontally](image)

Click the **Evenly distribute horizontally** tool to put an equal amount of horizontal space between a series of objects (two or more). For an example, see the following figure:

![Distributing Objects Horizontally](image)

Click the **Evenly distribute vertically** tool to put an equal amount of vertical space between a series of objects (two or more). For an example, see the following figure:

![Distributing Objects Vertically](image)

**Note:** The distribution tools may move the last object selected (with solid handles) by no more than a few pixels to equally space all of the objects.
**Rotate Tool**

Click the Rotate tool to rotate the selected object 90 degrees (a quarter turn) clockwise.

![Diagram showing rotation](image)

**Note:** You can use this tool only with a single selected object or grouped object. You cannot use this tool with multiple objects selected.

**FLIP VERTICALLY TOOL**

Click the Flip Vertically tool to invert the selected object vertically. The object rotates around an imaginary line through its vertical center until it is a mirror image of the original object. For an example, see the following figure:

![Diagram showing vertical flipping](image)

**Note:** You can use this tool only with a single selected object or grouped object. You cannot use this tool with multiple objects selected.

**FLIP HORIZONTALLY TOOL**

Click the Flip Horizontally tool to invert the selected object horizontally. The object rotates around an imaginary line through its horizontal center until it is a mirror image of the original object. For example, see the following figure:

![Diagram showing horizontal flipping](image)

**Note:** You can use this tool only with a single selected object or grouped object. You cannot use this tool with multiple objects selected.
Resize Tools

Use the following ribbon options for resizing:

- Click the **Resize width** tool to set the width of all selected objects to the width of the last object selected, or to resize one selected object so that its width equals its height.
- Click the **Resize height** tool to set the height of all selected objects to the height of the last object selected, or to resize one selected object so that its height equals its width.

**Tip:** You can use **Resize width** and **Resize height** to turn an ellipse into a circle or a rectangle into a square. Make sure you have only one object selected, however.

You also can use the mouse pointer and arrow keys to resize objects. When you select an object (or group of objects) with the pointer, handles are displayed at each corner and at the midpoint of each side. You can use these handles as follows:

- **To enlarge an object**, drag a handle in the direction you want to resize the object. Dragging a side handle resizes the object in one direction only (height only or width only). Dragging a corner handle resizes the entire object (height and width).

  When you drag a corner handle, the object’s proportions are constrained by default. To freely resize the object, hold down the **SHIFT** key as you drag the handle.

- **To resize an object one pixel at a time**, click and hold a handle and then press the arrow keys. For the corner handles and the left and right side handles, press the **LEFT ARROW** and **RIGHT ARROW** keys. For the top and bottom handles, press the **UP ARROW** and **DOWN ARROW** keys.

- **To resize an Open or Closed Polygon**, draw a selection box around all of the polygon’s points and **group** them. You can then resize the polygon like a normal object.

**Note:** When you resize a **Symbol**, a **Group**, or any other collection of selected objects, all of the objects in the collection are resized in the same direction and to the same degree.

Fill Color Tool

Click the **Fill Color** tool to specify a default fill color for the following objects:

- **Closed Polygons**
- **Ellipses**
- **Rounded Rectangles**
- **Rectangles**

**Tip:** To save development time, select several objects (of any type specified in the preceding list) and use **Fill Color** to specify a default fill color for all of them at once.

Line Color Tool

Click the **Line Color** tool to specify a line color for selected objects or to set a default color for new objects, including the following:

- **Open Polygons**
- **Closed Polygons**
- **Lines**
- **Ellipses**
- **Rounded Rectangles**
- **Rectangles**

When you click the **Line Color** tool, the **Line Selection** dialog displays. Use this dialog to specify line styles and color for the selected objects.
**Tip:** To save development time, you can select several of the preceding objects and use the **Line Color** tool to specify a line color for all of the objects at once.

### Fonts Tool

Click the **Fonts** tool to specify the font and color for selected **Text** objects, or to specify a default font and color for new **Text** objects.

**Tip:** To save development time, select several **Text** objects and use the **Fonts** tool to specify font and color settings for all of the objects at once. (You cannot use this function for **grouped** **Text** objects however.)
About using Multi-Touch in your project

You can enable multi-touch gestures in project screens to provide the end user with additional interface options.

On a traditional touchscreen device that supports only a single touch point, the user's touch or tap is directly equivalent to a simple mouse click, and the end user cannot do anything that they could not otherwise do by connecting a mouse to the device.

On a newer touchscreen device that supports multiple touch points, however, the end user may manipulate project screens and screen objects with two or more fingers at the same time. The additional touch points provide context that is not available in a single touch point; two fingers working together can perform different gestures, and different gestures trigger different actions on-screen. For example:

- You may swipe two fingers together in the same direction to quickly pan through a window or scroll through a list;
- You may stretch and pinch with two fingers to resize a screen object or zoom in/out on a project screen; or
- You may "grip" a screen object with two fingers and then rotate it like a dial.

Moreover, if you are experienced with VBScript, then you may use Touch Events to customize the behavior of multi-touch beyond the standard gestures described in this section. These Touch Events are actually VBScript sub-routines that receive the raw touch input data from the Windows API.

Requirements and considerations

These requirements do not apply to the computer that you are using to develop your InduSoft Web Studio project; if you can run the development application and create a project, then you can configure the necessary settings to include multi-touch in your project.

Instead, these requirements apply to the client devices that your end users will use to access your project. To support multi-touch gestures, a client device must have:

- Microsoft Windows 7, Microsoft Windows 8, or Microsoft Windows Embedded Compact 7;
- A touchscreen display capable of reading two or more touch points; and
- Either Web Thin Client or Secure Viewer installed for Windows, EmbeddedView installed for Windows Embedded Standard, or CEView installed for Windows Embedded Compact.

To confirm that the client device supports multi-touch gestures, open the System control panel (Start > Control Panel > System and Security > System) and then look for Pen and Touch. It must say that touch input is available with at least two touch points.

Pen and Touch in the System control panel

If you will be running your project in a mixed environment — that is, if your end users will be using different types of client devices, including some that do not support touch input — then you should be careful about how you include multi-touch gestures in your project. Always provide a second way to manipulate a screen or object, using a keyboard, a mouse, or a single-finger tap.

Support for Windows Embedded Compact

Support for multi-touch gestures on Windows Embedded Compact devices is limited. The specific limitations are described in the respective sections, but the table below provides a summary.
**Support for multi-touch gestures by operating system**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Windows 7 / Windows 8</th>
<th>Windows Embedded Compact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoom and Pan gestures on project screens</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Gestures with Alarm/Event Control object</td>
<td>Supported</td>
<td>&quot;Slide to scroll&quot; supported, &quot;Swipe to select&quot; not supported</td>
</tr>
<tr>
<td>Gestures with Trend Control object</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Gestures with Grid object</td>
<td>Supported</td>
<td>&quot;Slide to scroll/pan&quot; supported, &quot;Swipe to select&quot; not supported</td>
</tr>
<tr>
<td>Gestures with Position animation</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Gestures with Resize animation</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Gestures with Rotation animation</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>Touch Events for up to 10 touch points</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**About the Multi-Touch settings for project screens**

The Multi-Touch settings determine how multi-touch gestures behave in each project screen.

You may configure default settings for your entire project, so that each new project screen has the same settings as all other screens of the same type, and you may also customize the settings for specific screens when the default settings would not be appropriate.

**CONFIGURE THE DEFAULT MULTI-TOUCH SETTINGS FOR ALL SCREENS**

Configure the default Multi-Touch settings for all screens in your project, so that the screens all behave the same way during run time.

The Multi-Touch settings come preconfigured for most common uses, so you need to further configure them only if:

- You are not satisfied with the run-time behavior of these settings; and/or
- You want to use project tags to programmatically change the setting during run time.

Keep in mind that these are the default settings for all screens in your project. If you only want to configure the settings for a specific screen, see Configure the Multi-Touch settings for a specific screen on page 261.

To configure the default Multi-Touch settings for all screens:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Viewer**.
The Project Settings dialog is displayed with the Viewer tab selected.

2. Click Multi-Touch Settings. The Multi-Touch Settings dialog is displayed.

3. In the Screen Style list, click the style for which you want to configure the default settings. Every style has its own settings, so you may need to repeat the following steps for each style. For more information about the different styles of project screens, see Screen Attributes dialog on page 175.

4. In the Enable list, either click an option or type the name of a project tag (Boolean or Integer type). This setting determines whether the Multi-Touch features in general (i.e., gestures, screen zoom/pan, inertia, touch events) are enabled for the selected screen style. If you typed the name of a project tag, then the value of the tag will control the setting during run time.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:No</td>
<td>The Multi-Touch features are disabled for the selected screen style.</td>
</tr>
<tr>
<td>1:Yes</td>
<td>The Multi-Touch features are enabled for the selected screen style.</td>
</tr>
</tbody>
</table>

5. In the Screen Zoom/Pan Gestures area, configure the zoom settings.
a) In the Mode box, either select an option or type the name of a project tag (Integer type).
This setting determines the zoom/pan mode for the selected screen style. If you type the name of a project tag, the value of that tag will determine the mode during run time.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:Disable</td>
<td>Zoom/Pan is disabled for the selected screen style.</td>
</tr>
<tr>
<td>1:Inner</td>
<td>Inner Zoom/Pan is enabled for the selected screen style. Zooming changes the magnification of the viewable area, and panning moves the viewable area inside the screen border. In most cases, the screen itself does not change size or position.</td>
</tr>
<tr>
<td>2:Outer</td>
<td>Outer Zoom/Pan is enabled for the selected screen style. In practice, this is more like Resize/Move; zooming changes the size of the entire screen (thereby magnifying whatever is in the screen) and panning moves the screen in relation to the other open screens.</td>
</tr>
</tbody>
</table>

For more information, see Using multi-touch gestures in project screens on page 263.

b) In the Max Inner Zoom (%) box, either select an option or type the name of a project tag (Integer type). This will be the maximum magnification allowed for the viewable area inside the project screen, when Mode is set to Inner. If you type the name of a project tag, the value of that tag will determine the zoom during run time. Values less than 100 (i.e., 100%) and greater than 1000 (i.e., 1000%) will be ignored.

c) In the Min Outer Zoom (%) box, either select an option or type the name of a project tag (Integer type). This will be the minimum size allowed for the project screen (as a percentage of size specified in the screen attributes), when Mode is set to Outer. If you type the name of a project tag, the value of that tag will determine the zoom during run time. Values less than 20 (i.e., 20%) and greater than 100 (i.e., 100%) will be ignored.

d) In the Max Outer Zoom (%) box, either select an option or type the name of a project tag (Integer type). This will be the maximum size allowed for the project screen (as a percentage of the size specified in the screen attributes), when Mode is set to Outer. The default option is (Auto), which means that the maximum screen size will be equal to the size of the display on which the project is viewed. If you type the name of a project tag, the value of that tag will determine the zoom during run time. Values less than 100 (i.e., 100%) and greater than 1000 (i.e., 1000%) will be ignored.

Note: These settings only apply to projects running on Windows, because multi-touch gestures in project screens are not supported on Windows Embedded Compact.

6. In the Inertia Settings area, configure the deceleration values for the different types of movement. All values are in pixels per second.
   a) In the Deceleration box, either type a value or type the name of a project tag (Real type) that contains the value.
      This value controls the deceleration from "slide to pan" and "slide to move" gestures.
   b) In the Angular Deceleration box, either type a value or type the name of a project tag (Real type) that contains the value.
      This value controls the deceleration from "turn to rotate" gestures.
   c) In the Expansion Deceleration box, either type a value or type the name of a project tag (Real type) that contains the value.
      This value controls the deceleration from "pinch/stretch to resize" and "pinch/stretch to zoom" gestures.

For a detailed explanation of how inertia is used in multi-touch gestures, see "Inertia Mechanics" at the Microsoft Developer Network website: msdn.microsoft.com/library/dd562169.aspx

Note: These settings only apply to projects running on Windows, because Touch Input on Windows Embedded Compact does not include support for inertia.

7. Repeat from Step 3 for each style of screen that you want to configure.

8. Click OK to save the settings and close the dialog.
CONFIGURE THE MULTI-TOUCH SETTINGS FOR A SPECIFIC SCREEN

Configure the Multi-Touch settings for a specific screen if the project defaults would not be appropriate. Before you begin this task, you should have the selected Screen worksheet open for editing.

Keep in mind that the project defaults are there to ensure that the screens all behave the same way during run time. Consistency makes your project easier to use. (For more information, see Configure the default Multi-Touch settings for all screens on page 258.) As such, you should change the settings for a specific screen only when it is absolutely necessary to the purpose of that screen.

To configure the Multi-Touch settings for a specific screen:

1. Do one of the following:
   • On the Graphics tab of the ribbon, in the Screen group, click Attributes; or
   • Right-click anywhere in the Screen worksheet, and then click Screen Attributes on the shortcut menu.
   The Screen Attributes dialog is displayed.

2. Click Multi-Touch Settings.
   The Multi-Touch Settings dialog is displayed, with most of the settings disabled because the screen is using the project defaults.

3. Clear the Use Project Default option.

The remaining settings are enabled for configuring.

4. In the Enable list, either click an option or type the name of a project tag (Boolean or Integer type).
This setting determines whether the Multi-Touch features in general (i.e., gestures, screen zoom/pan, inertia, touch events) are enabled for the selected screen style. If you typed the name of a project tag, then the value of the tag will control the setting during run time.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:No</td>
<td>The Multi-Touch features are disabled for the selected screen style.</td>
</tr>
<tr>
<td>1:Yes</td>
<td>The Multi-Touch features are enabled for the selected screen style.</td>
</tr>
</tbody>
</table>

5. In the **Screen Zoom/Pan Gestures** area, configure the zoom settings.
   a) In the **Mode** box, either select an option or type the name of a project tag (Integer type).
   This setting determines the zoom/pan mode for the selected screen style. If you type the name of a project tag, the value of that tag will determine the mode during run time.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:Disable</td>
<td>Zoom/Pan is disabled for the selected screen style.</td>
</tr>
<tr>
<td>1:Inner</td>
<td>Inner Zoom/Pan is enabled for the selected screen style. Zooming changes the magnification of the viewable area, and panning moves the viewable area inside the screen border. In most cases, the screen itself does not change size or position.</td>
</tr>
<tr>
<td>2:Outer</td>
<td>Outer Zoom/Pan is enabled for the selected screen style. In practice, this is more like Resize/Move; zooming changes the size of the entire screen (thereby magnifying whatever is in the screen) and panning moves the screen in relation to the other open screens.</td>
</tr>
</tbody>
</table>

For more information, see Using multi-touch gestures in project screens on page 263.

   b) In the **Max Inner Zoom (%)** box, either select an option or type the name of a project tag (Integer type).
   This will be the maximum magnification allowed for the viewable area inside the project screen, when **Mode** is set to **Inner**. If you type the name of a project tag, the value of that tag will determine the zoom during run time. Values less than 100 (i.e., 100%) and greater than 1000 (i.e., 1000%) will be ignored.

   c) In the **Min Outer Zoom (%)** box, either select an option or type the name of a project tag (Integer type).
   This will be the minimum size allowed for the project screen (as a percentage of size specified in the screen attributes), when **Mode** is set to **Outer**. If you type the name of a project tag, the value of that tag will determine the zoom during run time. Values less than 20 (i.e., 20%) and greater than 100 (i.e., 100%) will be ignored.

   d) In the **Max Outer Zoom (%)** box, either select an option or type the name of a project tag (Integer type).
   This will be the maximum size allowed for the project screen (as a percentage of the size specified in the screen attributes), when **Mode** is set to **Outer**. The default option is **(Auto)**, which means that the maximum screen size will be equal to the size of the display on which the project is viewed. If you type the name of a project tag, the value of that tag will determine the zoom during run time. Values less than 100 (i.e., 100%) and greater than 1000 (i.e., 1000%) will be ignored.

   ![Note: These settings only apply to projects running on Windows, because multi-touch gestures in project screens are not supported on Windows Embedded Compact.]

6. In the **Inertia Settings** area, configure the deceleration values for the different types of movement.
   All values are in pixels per second.
   a) In the **Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.
   This value controls the deceleration from "slide to pan" and "slide to move" gestures.

   b) In the **Angular Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.
   This value controls the deceleration from "turn to rotate" gestures.

   c) In the **Expansion Deceleration** box, either type a value or type the name of a project tag (Real type) that contains the value.
This value controls the deceleration from "pinch/stretch to resize" and "pinch/stretch to zoom" gestures.

For a detailed explanation of how inertia is used in multi-touch gestures, see "Inertia Mechanics" at the Microsoft Developer Network website: msdn.microsoft.com/library/dd562169.aspx

Note: These settings only apply to projects running on Windows, because Touch Input on Windows Embedded Compact does not include support for inertia.

7. Click OK to save the settings and close the dialog.

About the different types of multi-touch gestures

This section describes the different types of multi-touch gestures and how they can be used in your project. The gestures themselves — swipe, slide, pinch, stretch, and so on — are a standard part of Windows 7 and Windows 8, so you may be already familiar with using them on a smart phone or tablet. And even if you are not, there are illustrations of gestures included.

What this section describes is which gestures may be used in which areas of your project. For example, the same "pinch" and "stretch" gestures may be used to resize a project screen or resize a screen object with the Resize animation or even navigate through a trend graph.

Please note that for the purposes of this documentation, "multi-touch gesture" almost always means a gesture using two fingers. There are some exceptions, such as using a one-finger swipe to select cells in a Grid object, but those exceptions will be descibed in detail in the appropriate topics.

USING MULTI-TOUCH GESTURES IN PROJECT SCREENS

You may use multi-touch gestures to either zoom-and-pan or resize-and-move a project screen during run time, depending on how the screen is configured.

Specifically, it depends on whether Zoom/Pan Mode for a given screen is set to Inner or Outer. For more information, see About the Multi-Touch settings for project screens on page 258.

Note: For these gestures to work, at least two fingers must be touching the same project screen. If only one finger is inside the screen and the others are outside it, then the touch input will be ignored.

Inner Zoom/Pan

When Inner Zoom/Pan is enabled for a project screen, you may use two-finger gestures to zoom and pan the viewable area inside the screen's border. Simply touch your fingers to any part of the screen during run time, and then either pinch and stretch to zoom or slide to pan. The screen itself is not resized or moved in most cases.
This is typically used when you need to view an image or workspace that is larger than your display. You may load the large image into a smaller project screen and then use zoom and pan to manipulate it.

**Note:** If the zoom is at 100% and the end user tries to pan, then the entire screen will move rather than the viewable area in the screen.

These gestures are not supported on Windows Embedded Compact.

**Outer Zoom/Pan**

When Outer Zoom/Pan is enabled for a project screen, you may use two-finger gestures to resize and move that screen on the display, just as you would resize and move windows on the desktop. Simply touch your fingers to any part of the screen during run time, and then either pinch and stretch to resize or slide to move.
This is essentially the same as using one-finger gestures (or a connected mouse) to resize and move the screen, except that one-finger gestures only work on a screen with the Resizing border (as configured in Screen Attributes) and only when you "grab" the border itself.

These gestures are not supported on Windows Embedded Compact.

**USING MULTI-TOUCH GESTURES IN DATA OBJECTS**

You may use multi-touch gestures to manipulate Alarm/Event, Trend, and Grid objects during run time.

![Note: For these gestures to work, all fingers must be touching the same screen object. If only one finger is inside the object and the others are outside it, then the touch input will be ignored.]

**Alarm/Event Control object**

When Multi-Touch is enabled for a project screen, then any Alarm/Event Control object in that screen may be manipulated with multi-touch gestures. Specifically, you may swipe with one finger to select items in the list of alarms/events, and you may slide with one or two fingers to scroll through the list.

![Swipe to select]

![Slide to scroll]

*Gestures for the Alarm/Event Control object*

The "Slide to scroll" gesture (with one finger only) is supported on Windows Embedded Compact. The "Swipe to select" gesture is not.
**Trend Control object**

When Multi-Touch is enabled for a project screen, then any **Trend Control object** in that screen may be manipulated with multi-touch gestures. Specifically, you may either pinch/stretch to zoom or slide to pan the viewable area of the trend.

---

**Gestures for the Trend Control object**

Zooming changes the period and scale of the trend (i.e., the X and Y axes) just as if you clicked any of the **Zoom** tools on the Trend Control object’s toolbar. Consequently, if you click **Cancel Zoom** on the toolbar, then any zooming done by your gestures will be canceled and the period and scale will be reset.

Panning works only when the Trend Control object is configured to show historical data.

Both of these gestures are supported on Windows Embedded Compact.

---

**Grid object**

When Multi-Touch is enabled for a project screen, then any **Grid object** in that screen may be manipulated with multi-touch gestures. Specifically, you may swipe with one finger to select cells in the grid, and you may slide with two fingers to scroll/pan the viewable area of the grid.
The "Slide to scroll" and "Slide to pan" gestures are both supported on Windows Embedded Compact. The "Swipe to select" gesture is not.

**USING MULTI-TOUCH GESTURES IN OBJECT ANIMATIONS**

You may use multi-touch gestures to trigger Position, Resize, and Rotation animations during run time. Gestures for the Position and Resize animations are supported on Windows Embedded Compact. Gestures for the Rotation animation are not.

---

For these gestures to work, all fingers must be touching the same screen object. If only one finger is inside the object and the others are outside it, then the touch input will be ignored.
**Position animation**

When Multi-Touch is enabled for a project screen, any object in that screen with a Position animation may be manipulated with multi-touch gestures. Specifically, you may slide with one or more fingers to move the object.

![Gesture for the Position animation](image)

This is essentially the same as when Multi-Touch is disabled, of course, because the purpose of the Position animation is to make the object movable. In this case, the primary benefit of enabling Multi-Touch is inertia, which makes the movement of the object more natural.

This gesture is supported on Windows Embedded Compact.

**Resize animation**

When Multi-Touch is enabled for a project screen, any object in that screen with a Resize animation may be manipulated with multi-touch gestures. Specifically, you may pinch and stretch with two fingers to resize the object.

![Gesture for the Resize animation](image)

This gesture is supported on Windows Embedded Compact.
Rotation animation

When Multi-Touch is enabled for a project screen, you may use a two-finger gesture to "grip" and turn any object with a Rotation animation in that screen. Simply touch your fingers to the object and turn it.

This gesture is not supported on Windows Embedded Compact.

About Touch Events

Touch Events are predefined VBScript sub-routines that may be added to screen objects and project screens to create custom touch behaviors.

These Touch Events are based directly on the Windows Touch API — specifically, on the _IManipulationEvents interface and the methods ManipulationStarted, ManipulationDelta, and ManipulationCompleted. For more information, see "Touch Input" at the Microsoft Developer Network website: msdn.microsoft.com/library/dd562197.aspx

These methods pass the raw touch input data for up to 10 touch points from the Windows operating system to InduSoft Web Studio, and then IWS passes the data to your project through the Touch Events.

In practice, Touch Events are essentially the same as any other VBScript interface in your project. You select a screen object or project screen, add the appropriate Touch Event depending on when you want the script to be executed (e.g., when the user starts or stops touching), and then develop the script to do whatever you want. The only real differences between Touch Events and the other VBScript interfaces are: Touch Events can only be executed as sub-routines, which means that they do not return values; and they can only receive the selected touch input data as parameters. If you can work within these guidelines, however, you can use Touch Events to develop custom touch behaviors far beyond the standard gestures.

ADD A TOUCH EVENT TO A SCREEN OBJECT

Add a Touch Event to a screen object in order to process touch input on that object.

Before you begin this task, you should already have the screen worksheet open for editing. You should also have created the screen object to which you want to add the Touch Event.

Touch Events are added to a screen object through the Command animation.

To add a Touch Event to a screen object:

1. Click the screen object to which you want to add the Touch Event.
   The screen object is selected.

2. On the Graphics tab of the ribbon, in the Animations group, click Command.
   The Command animation is added to the selected screen object.
3. Double-click the screen object.
   The Object Properties dialog is displayed.

4. If the Command animation properties are not already displayed, then in the properties list (in the top-right corner of the dialog), click **Command**.

5. If the VBScript event tabs are not already displayed, then in the **Type** list, click **VBScript**.

![VBScript event tabs in the Command animation properties](image)

6. In the VBScript event tabs, click the tab for the Touch Event that you want to add.
   You may need to use the arrow buttons to scroll through the tabs.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On Touch</td>
<td>Process the raw data from the touch input. The sub-routine is continuously executed while the user touches the screen object.</td>
</tr>
<tr>
<td>On Touch Start</td>
<td>Perform an action once, when the user starts touching the screen object.</td>
</tr>
<tr>
<td>On Touch Delta</td>
<td>Perform an action each time the user manipulates on the screen object.</td>
</tr>
<tr>
<td>On Touch Completed</td>
<td>Perform an action once, when the user stops touching the screen object.</td>
</tr>
</tbody>
</table>

7. Click in the text box below, and then type your VBScript code.

   At this point, you can develop the Touch Event just as you would develop any other VBScript sub-routine in your project. For more information, see Overview of VBScript on page 943.

**ADD A TOUCH EVENT TO A PROJECT SCREEN**

Add a Touch Event to a project screen in order to process touch input on the screen in general, rather than on a specific object in the screen.

Before you begin this task, you should have the selected Screen worksheet open for editing.

Touch Events are added to a project screen through the **Screen Script worksheet**.

> **Note:** Touch Events are not supported on Windows Embedded Compact.

To add a Touch Event to a project screen:

1. Do one of the following:
   - On the **Graphics** tab of the ribbon, in the **Screen** group, click **Script**; or
   - Right-click anywhere in the screen worksheet, and then click **Screen Script** on the shortcut menu.

   The screen’s associated script worksheet is opened for editing.
2. Right-click anywhere in the script worksheet, and then point to **Add Touch Event** on the shortcut menu. A sub-menu of the available Touch Events is displayed.

3. On the sub-menu, click the Touch Event that you want to add.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub Screen_OnTouch</td>
<td>Process the raw data from the touch input. The sub-routine is continuously executed while the user touches the project screen.</td>
</tr>
<tr>
<td>Sub Screen_OnTouchStart</td>
<td>Perform an action once, when the user starts touching the project screen.</td>
</tr>
<tr>
<td>Sub Screen_OnTouchDelta</td>
<td>Perform an action each time the user moves his or her fingers on the project screen.</td>
</tr>
<tr>
<td>Sub Screen_OnTouchCompleted</td>
<td>Perform an action once, when the user stops touching the project screen.</td>
</tr>
</tbody>
</table>

The Touch Event is inserted as a VBScript sub-routine in the Screen Script worksheet.

4. Click in the worksheet and type your VBScript code.

At this point, you can develop the Touch Event just as you would develop any other VBScript sub-routine in your project. For more information, see **Overview of VBScript** on page 943.

**ONTOUCH**

Use the sub-routine `OnTouch` in VBScript to process the raw touch point data that are provided while the user touches the project screen or screen object.

**Syntax**

```vbnet
Sub OnTouch(arX, arY, arIDs, arFlags, arMask, arTime, arXContacts, arYContacts)
... 
End Sub
```

`arX`  
An array of integer values, from `arX(0)` to `arX(n)`, providing the x-coordinates (in pixels from the left of the screen) of the currently active touch points.

`arY`  
An array of integer values, from `arY(0)` to `arY(n)`, providing the y-coordinates (in pixels from the top of the screen) of the currently active touch points.

`arIDs`  
An array of integer values, from `arIDs(0)` to `arIDs(n)`, providing the unique identifiers of the currently active touch points. Each discrete touch point receives its own identifier, even if it is the same finger touching, then lifting, then touching again. These identifiers are incremented from when the device is turned on, and they include all touches captured by the operating system, not just those captured by your project during run time.

`arFlags`  
An array of integer values, from `arFlags(0)` to `arFlags(n)`, where each value is a set of bit flags that specify various aspects of touch point press, release, and motion.

For more information about the bit flags and their possible values, go to “TOUCHINPUT structure” on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317334.aspx

`arMask`  
An array of integer values, from `arMask(0)` to `arMask(n)`, where each value is a set of bit flags that specify which of the optional parameters (i.e., `arTime`, `arXContacts`, `arYContacts`) contain
valid information. The availability of valid information is device-specific; for example, for the parameter arTime, some devices provide only the time elapsed since the device was turned on, rather than the actual system time.

For more information about the bit flags and their possible values, go to "TOUCHINPUT structure" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317334.aspx

**arTime**

An array of integer values, from arTime(0) to arTime(n), providing the timestamps (in milliseconds) of the currently active touch points.

**arXContacts**

An array of integer values, from arXContacts(0) to arXContacts(n), providing the widths (in hundredths of a pixel) of the contact areas of the currently active touch points. The contact area of a touch point is the area actually touched by the user’s fingertip.

**arYContacts**

An array of integer values, from arYContacts(0) to arYContacts(n), providing the heights (in hundredths of a pixel) of the contact areas of the currently active touch points. The contact area of a touch point is the area actually touched by the user’s fingertip.

**Returned value**

This is a sub-routine (as opposed to a function) in VBScript, so it does not return any value.

**Notes**

This sub-routine is based on the WM_TOUCH system message and the associated TOUCHINPUT data structure in the Windows API. For more information, go to "Windows Touch Input" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317321.aspx

The sub-routine is executed continuously while the user is touching the project screen or screen object. There are no delta or cumulative values, so there is nothing to reset when the manipulation is completed. These are the raw data provided by the Windows API.

You are not required to use the received parameters in your code. They simply make the raw touch input data available to you, for you to use (or not) as you deem necessary.

In all of the parameters described above, the array elements represent the individual touch points on the screen, in the order that the user actually touches the screen. The first array element (position 0) is the first touch point, the second array element (position 1) is the second touch point, and so on up to the maximum number of touch points supported by the device.

Please note that the arrays are dynamically resized to fit to the current number of active touch points. In other words, the array elements do not exist until the user’s fingers actually touch the screen and the corresponding touch points are added, and the array elements are subsequently eliminated when the touch points are removed. This can make it difficult to reference the array elements in your project unless you include the following code (or something similar) in the sub-routine:

```
    n = UBound(arX)
    For i = 0 to n
        $TouchX[i] = arX(i)
        $TouchY[i] = arY(i)
        $TouchID[i] = arIDs(i)
        $TouchTime[i] = arTime(i)
    Next
```

The function **UBound** measures the current size of arX (although any of the parameters may be used), and then the **For** loop copies the values to appropriately named tag arrays (e.g., **TouchX**, **TouchY**) in your project tags database. Once this is done, you can reference the tag arrays rather than the parameters.
Unlike the parameters, the tag arrays are not dynamically resized, so garbage values may be left in the higher array positions when touch points are removed. To clean out those garbage values, you might also include the following code (or something similar) in the sub-routine:

```vbs
s = $TouchX->Size
For i = (n+1) to s
    $TouchX[i] = 0
    $TouchY[i] = 0
    $TouchID[i] = 0
    $TouchTime[i] = 0
Next
```

By this time, you may have noticed that there is no graceful way to handle the elimination of array elements from anything other than the highest array position. If the user touches the screen with two fingers and then lifts their second finger, the second element of the array (position 1) is eliminated without issues. But if the user touches the screen with two fingers and then lifts their first finger, the first element of the array (position 0) is eliminated and the second element (position 1) becomes the first element (position 0).

You can use the unique identifiers provided by arIDs, rather than the array positions that will change as the arrays are dynamically resized, to handle specific touch points over time. The exact method for doing that, however, depends on how you develop the rest of your project and therefore is beyond the scope of this documentation.

**ONTOUCHSTART**

Use the sub-routine `OnTouchStart` in VBScript to perform an action when the user starts touching the project screen or screen object.

**Syntax**

```
Sub OnTouchStart(x,y)
    ...
End Sub
```

**x**

The starting x-coordinate (in pixels from the left of the screen) of the first touch point.

**y**

The starting y-coordinate (in pixels from the top of the screen) of the first touch point.

**Returned value**

This is a sub-routine (as opposed to a function) in VBScript, so it does not return any value.

**Notes**

This sub-routine is based on the method `ManipulationStarted` in the Windows API. For more information, go to "Windows Touch Input" on the Microsoft Developer Network website at: [msdn.microsoft.com/library/dd317321.aspx](msdn.microsoft.com/library/dd317321.aspx)

The sub-routine is executed once when the manipulation is started — that is, when the first touch point is added to the project screen or screen object. Additional touch points after the first do not trigger this sub-routine.

You are not required to use the received parameters in your code. They simply make the raw touch input data available to you, for you to use (or not) as you deem necessary.
ONTOUCHDELTA

Use the sub-routine OnTouchDelta in VBScript to perform an action each time the user manipulates the project screen or screen object.

Syntax

Sub OnTouchDelta(x, y, deltaX, deltaY, deltaScale, deltaExpansion, deltaRotation, cumulativeX, cumulativeY, cumulativeScale, cumulativeExpansion, cumulativeRotation, inertiaEnabled)

End Sub

x
The current x-coordinate (in pixels from the left of the screen) of the first touch point.

y
The current y-coordinate (in pixels from the top of the screen) of the first touch point.

deltaX
The change (in pixels) between the previous x-coordinate and the current x-coordinate of the first touch point.

deltaY
The change (in pixels) between the previous y-coordinate and the current y-coordinate of the first touch point.

deltaScale
The change (as a percentage) in the distance between the first and second touch points.

deltaExpansion
The change (in pixels) in the distance between the first and second touch points.

deltaRotation
The change in the angle of rotation (in radians) indicated by the first and second touch points.

cumulativeX
The total change (in pixels) between the starting x-coordinate and the current x-coordinate of the first touch point.

cumulativeY
The total change (in pixels) between the starting y-coordinate and the current y-coordinate of the first touch point.

cumulativeScale
The total change (as a percentage) in the distance between the first and second touch points, from the start of the manipulation.

cumulativeExpansion
The total change (in pixels) in the distance between the first and second touch points, from the start of the manipulation.

cumulativeRotation
The total change in the angle of rotation (in radians) indicated by the first and second touch points, from the start of the manipulation.

inertiaEnabled
A boolean value indicating whether inertia is enabled for the project screen or screen object.
Returned value
This is a sub-routine (as opposed to a function) in VBScript, so it does not return any value.

Notes
This sub-routine is based on the method ManipulationDelta in the Windows API. For more information, go to "Windows Touch Input" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317321.aspx

The sub-routine is executed once for each discrete movement in the current manipulation. Changes in position require one touch point. Changes in size and/or rotation require two touch points.

You are not required to use the received parameters in your code. They simply make the raw touch input data available to you, for you to use (or not) as you deem necessary.

ONTOUCHCOMPLETED
Use the sub-routine OnTouchCompleted in VBScript to perform an action when the user stops touching the project screen or screen object.

Syntax

Sub OnTouchCompleted(x,y,cumulativeX,cumulativeY,cumulativeScale,cumulativeExpansion,cumulativeRotation)
  ...
End Sub

x
  The ending x-coordinate (in pixels from the left of the screen) of the first touch point.

y
  The ending y-coordinate (in pixels from the top of the screen) of the first touch point.

cumulativeX
  The total change (in pixels) between the starting x-coordinate and the ending x-coordinate of the first touch point.

cumulativeY
  The total change (in pixels) between the starting y-coordinate and the ending y-coordinate of the first touch point.

cumulativeScale
  The total change (as a percentage) in the distance between the first and second touch points, from the start to the end of the manipulation.

cumulativeExpansion
  The total change (in pixels) in the distance between the first and second touch points, from the start to the end of the manipulation.

cumulativeRotation
  The total change in the angle of rotation (in radians) indicated by the first and second touch points, from the start to the end of the manipulation.

Returned value
This is a sub-routine (as opposed to a function) in VBScript, so it does not return any value.

Notes
This sub-routine is based on the method ManipulationCompleted in the Windows API. For more information, go to "Windows Touch Input" on the Microsoft Developer Network website at: msdn.microsoft.com/library/dd317321.aspx
The sub-routine is executed once when the manipulation is completed — that is, when the last touch point is removed from the project screen or screen object.

You are not required to use the received parameters in your code. They simply make the raw touch input data available to you, for you to use (or not) as you deem necessary.
**Import a Studio XML Screen**

Use the Import Wizard to import a Studio XML Screen, which is an external text file created with InduSoft Web Studio’s custom XML schema.

Before you begin this task, you must have a properly formatted Studio XML Screen file that you can import.

A Studio XML Screen file contains the same information as a regular screen file (*.SCR). It is simply formatted as human-readable XML instead of binary data, which makes it more flexible and portable.

XML files can be manually created in any text editor, of course, but our intent is to enable you to quickly and programmatically create large numbers of screens in Microsoft Visual Studio using our custom XML schema and its associated C# API. This often requires additional training and support, so the schema and API are not included in the standard installation of InduSoft Web Studio. Instead, the schema, API, documentation, sample files, and support contract can be purchased as a separate toolkit. For more information, please contact your software vendor.

Once you have created your screens, you can use the Import Wizard to batch import them into your IWS project.

**Tip:** You can also use the function `ImportXML` to import Studio XML Screen files during run time.

To import one or more Studio XML Screens:

1. On the **Home** tab of the ribbon, in the **Tools** group, click **Import Wizard**. The **Import Wizard** dialog box is displayed.
2. In the **Source Type** list, click **Studio XML Screen**, and then click **Next**. The next step of the import wizard is displayed.

   ![Selecting the screens to import](image)

3. Under **Options**, choose whether imported screens should automatically replace existing screens in your project.

   Screens are considered to be duplicates if they have the same file name. For example, **Objects.xml** and **Objects.scr** would be duplicates.

   - If you do not want the imported screens to replace existing screens in your project, select **Do not import duplicate screens**. A warning will be displayed for each duplicate that you try to import.
• If you want the imported screens to automatically replace existing screens in your project, select Replace duplicates with imported screens.

4. Click Browse.
   A standard Open dialog box is displayed.

5. Use the file browser to locate and select the Studio XML Screen files that you want to import, and then click Open.
   You can Ctrl-click and Shift-click to select more than one file.
   The selected file(s) are displayed in the Screen(s) box.

6. Click Next.
   The selected screens are processed, and then the next step of the import wizard is displayed showing the project tags that are included in the selected screens.

7. In the list of project tags, select the tags that you want to import with the screens:
   • For each tag in the list, select or clear the check box to the left.
   • To select all of the check boxes, click Check. To clear all of the check boxes, click Uncheck.
   • To filter the list of tags, click Filter and then configure filter strings for one or more columns. You can use wildcard characters (* and ?) in the filter strings.

8. Click Finish.
   The screens and included tags are imported into your project. Also, the screens are automatically published for Thin Clients and Mobile Access — i.e., the corresponding *.HTML, *.SCC, and *.SSMA files are automatically generated and saved in your project’s Web folder, so the imported screens should be immediately available for you to select.
The Alarm and Trend tasks are used to log historical data, and the Alarm/Event and Trend Control objects are used to display historical data on a project screen.

These two features are normally used together, but they do not need to be; project data may be logged without being displayed during runtime, and the data displayed during runtime may be taken from outside the project.
The Alarms folder enables you to configure alarm groups and tags related to each group. The Alarm worksheet defines the alarm messages generated by the project. The primary purpose of an alarm is to inform the operator of any problems or abnormal condition during the process so he can take corrective action(s).

The Alarm worksheet is executed by the Background Task module (see Execution Tasks). It handles the status of all alarms and save the alarm messages to the history, if configured to do so, but it does not display the alarm messages to the operator; the Alarm/Event Control screen object, available on the Graphics tab of the ribbon, must be created and configured in a screen in order to display alarms.

To create a new Alarm worksheet, do one of the following:
- On the Insert tab of the ribbon, in the Task Worksheets group, click Alarm;
- Right-click the Alarms folder in the Project Explorer, and then click Insert on the shortcut menu; or
- Click New on the Application menu, click the File tab, and then select Alarm Worksheet.

To edit an existing Alarm worksheet, double-click it in the Project Explorer.

You can create multiple Alarm groups (worksheets) and each group can be configured with independent settings, such as message colors, history log enabled/disabled, and so forth.

Each Alarm worksheet is composed of two areas:
- **Header**: Settings applied to all tags and alarms configured in the same alarm group. These settings allow you to configure the formatting of the message and the actions that must be triggered based on alarm events (e.g., print alarms, send alarms by email, and so forth). For more information, see Header Settings.
- **Body**: Configure alarm messages and associate them to conditions linked to tags. For more information, see Body Settings.

**Note:**
- You can configure the Alarm Group to send notifications by Email automatically, based on alarm events. For more information, see Email Settings.
- The alarm properties associated to each tag (configured in the body of the alarm group) can also be edited by the Tag Properties dialog (Properties on the Home tab of the ribbon). However, before associating a tag to an alarm group, it is necessary to create the alarm group and configure the settings on its header, which will be applied to all tags associated to the group.
- As of IWS v6.1+SP2, the Alarm task has been modified to avoid automatically acknowledging alarms by another alarm. For example, the Hi (Lo) alarm should not be automatically...
acknowledged when the HiHi (LoLo) alarm becomes active. To enable the previous behavior, set
the following key in your project (.APP) file:

```
[Alarm]
UseLegacyPriorityAck=1
```

**Note:** The settings configured in the body of each Alarm worksheet are stored in the Tags Database
archive(s). Therefore, changes to the tags database may affect the content of the Alarm worksheets
(body). Notice that each tag/type cannot be available in more than one Alarm group simultaneously
because the Alarm Group is a property associated to each Tag/Alarm Type (e.g., Tag: Level; Alarm
Type: Hi; Alarm Group: 2).

**Alarm Worksheet Header**

The following table describes the Header settings on an Alarm worksheet:

<table>
<thead>
<tr>
<th>Field</th>
<th>Remarks</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Description of the alarm group. It is displayed on the workspace. This field is used for documentation only.</td>
<td>Text (up to 80 chars)</td>
</tr>
<tr>
<td>Group Name</td>
<td>Name of the Alarm group. During runtime, the operator can filter alarms based on the Group Name by the built-in Filters dialog of the Alarm/Event control object.</td>
<td>Text (up to 32 chars)</td>
</tr>
<tr>
<td>Email Settings</td>
<td>Launches the Email Settings dialog, where you can configure the settings for emails sent automatically based on alarm conditions.</td>
<td>Button</td>
</tr>
<tr>
<td>Advanced</td>
<td>Launches the Advanced Settings dialog, where you can configure the settings for emails sent automatically based on alarm conditions.</td>
<td>Button</td>
</tr>
<tr>
<td>On Line &gt; Display in Alarm Controls</td>
<td>When checked, the alarms are available to be displayed on the Alarm/Event Control object.</td>
<td>Checkbox</td>
</tr>
<tr>
<td>On Line &gt; Ack Required</td>
<td>When checked, the alarms require acknowledgment. In this case, the alarms are displayed on the Alarm/Event Control object (Online mode) until they are acknowledged AND normalized.</td>
<td>Checkbox</td>
</tr>
<tr>
<td>On Line &gt; Beep</td>
<td>When checked, the computer keeps beeping while there are alarm(s) to be acknowledged, currently active.</td>
<td>Checkbox</td>
</tr>
<tr>
<td>On Line &gt; Send to Printer</td>
<td>When checked, the alarm messages are sent to the printer as soon as the alarm event occurs. When using this option, you must use a matrix printer (instead of DeskJet or LaserJet) in order to print the message(s) and feed just one line — otherwise, each alarm will be printed in a different sheet of paper. The alarms will be printed in the default printer. If you want to send alarms to a printer different from the default printer, you can specify the printer path/name, editing the following parameter in the project_name.APP file:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>[AlarmLog] Device=PrinterPath/PrinterName</code></td>
<td>Checkbox</td>
</tr>
<tr>
<td>History &gt; Save to Disk</td>
<td>When checked, the alarm messages are stored in the history log when they become active.</td>
<td>Checkbox</td>
</tr>
<tr>
<td>History &gt; Generate Ack Messages</td>
<td>When checked, the alarm messages are stored in the history log when they are acknowledged.</td>
<td>Checkbox</td>
</tr>
<tr>
<td>History &gt; Generate Norm Messages</td>
<td>When checked, the alarm messages are stored in the history log when they become normalized.</td>
<td>Checkbox</td>
</tr>
<tr>
<td>Colors in Alarm Controls &gt; Enable</td>
<td>When checked, the alarms configured in this group will be displayed with the colors assigned to each alarm state (Activation, Acknowledgement or Normalization), according to the colors configured in the Alarm Group.</td>
<td>Color</td>
</tr>
<tr>
<td>Colors in Alarm Controls &gt; FG and BG</td>
<td>You can configure the text foreground color (FG) and background color (BG) for the alarms displayed on the Alarms/Events Control object. Each alarm state can be displayed with a different color schema:</td>
<td>Color</td>
</tr>
</tbody>
</table>
### EMAIL SETTINGS FOR ALARM WORKSHEET

IWS has the ability to send emails automatically when alarm events occur. The emails are sent using the standard SMTP (Simple Message Transfer Protocol). Therefore, you just need a valid email account with a SMTP Server and POP3 server — it is not necessary to install any additional software, such as Microsoft Outlook.

⚠️ **Important:** Before being email to send emails, it is necessary to execute successfully the `CnfEmail` function (from the built-in language) at least once. This function sets the email account parameters used when sending emails from the project (e.g., SMTP server, user name, password, and so forth).

<table>
<thead>
<tr>
<th>Field</th>
<th>Remarks</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable send automatic email</td>
<td>Name of the tag associated with the alarm.</td>
<td>Checkbox</td>
</tr>
<tr>
<td>To, Cc, Bcc</td>
<td>Target addresses to whom the emails will be sent. You can configure multiple email addresses in each box (To, Cc and/or Bcc) by separating the addresses with the semi-colon character (;).</td>
<td>Text and/or (Tag) (up to 1024 chars)</td>
</tr>
<tr>
<td>Subject</td>
<td>When selecting &quot;Use alarm message&quot;, the alarm message itself is used as the subject of the email to be sent. When selecting &quot;Custom&quot;, you can configure a custom text to be used as Subject when sending the alarm.</td>
<td>Radio-button / Text (up to 1024 chars)</td>
</tr>
<tr>
<td>Send 1 message per email</td>
<td>When checking this option, each alarm is sent in an individual email and all emails are sent when the Send Trigger is triggered. Otherwise, all alarm messages are buffered and sent in only one email when the Send Trigger is triggered. You cannot disable (uncheck) this option when the Subject option is configured with &quot;Use alarm message&quot;.</td>
<td>Checkbox</td>
</tr>
<tr>
<td>Remove failed messages from the buffer</td>
<td>When checking this option, the emails are removed from the buffer after attempting to send them, even if there was an error (failure) and the email was not sent. Otherwise, the messages are kept in the buffer until they are sent successfully or when the buffer reaches its maximum size.</td>
<td>checkbox</td>
</tr>
</tbody>
</table>
| Send email when alarm is                   | Allow you to configure which alarm events should generate emails:  
  - **Active:** When the alarm becomes active. | checkbox                 |

### Alarms Worksheet — Email Settings
### Field Remarks Syntax

- **Ack**: When the alarm is acknowledged.
- **Norm**: When the alarm is normalized.
  Notice that each event can be enabled/disabled individually.

<table>
<thead>
<tr>
<th>Current Status</th>
<th>The tag configured in this field, if any, is updated with the current status of the current or last email that the project attempted to send:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-2: Incorrect version of the INDMail.DLL library.</td>
</tr>
<tr>
<td></td>
<td>-1: The INDMail.DLL library is corrupted.</td>
</tr>
<tr>
<td></td>
<td>0: SendEmail function is not being executed.</td>
</tr>
<tr>
<td></td>
<td>1: Sending email(s)</td>
</tr>
<tr>
<td></td>
<td>2: Last email was sent successfully.</td>
</tr>
<tr>
<td></td>
<td>3: There was an error sending the last email.</td>
</tr>
</tbody>
</table>

| Current Error | The tag configured in this field, if any, is updated with the error message describing the result of the last email that the project attempted to send. Therefore, when configuring a tag in this field, this tag must be a String type. |

<table>
<thead>
<tr>
<th>Message Format</th>
<th>This interface allows you to configure the actual format of the message sent by email, based on the alarm event(s):</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- <strong>Day, Month, Year, Hour, Minute, Second, MS</strong>: The options checked will compose the timestamp for the alarm messages. MS stands for milliseconds.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Items</strong>: The options checked will compose the email message for each alarm. You can configure the order of the items, by using the Move Up and Move Down buttons.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Separator</strong>: Allow you to choose the separator used between the items checked in this interface.</td>
</tr>
</tbody>
</table>

While you configure these settings, the **Sample** field displays an example of the format of the message according to the settings being configured.

| Send Trigger | When the alarm events are generated, they are kept in an internal buffer (memory). When the tag configured in this field changes of value, the email(s) on the internal buffer are sent to the addresses configured in the **To**, **Cc** and **Bcc** fields. After being successfully sent, the emails are removed from the internal buffer. |

| Max buffer size | Maximum number of alarm messages (events) that can be stored in the internal buffer simultaneously. When this limit is reached, the buffer follows a FIFO (First-In, First-Out) behavior, discharging the older messages as soon as the newer messages are generated, guaranteeing that the buffer does not exceed the limit configured in this field. |

| Buffer size | The tag configured in this field, if any, is updated with the number of messages (events) currently stored in the internal buffer. |

| Clear Buffer | When the tag configured in this field changes of value, all messages (events) currently stored in the buffer are deleted. These messages will never be sent. |

| Disable send | When the value of the tag configured in this field is TRUE, the Email feature is temporarily disabled. Alarm events generated while the Email feature is disabled will not be stored in the internal buffer. Also, emails will NOT be sent in this condition, even if the tag configured in the field **Send Trigger** changes of value. |
ADVANCED SETTINGS FOR ALARM WORKSHEET

The following table describes the Advanced settings on an Alarm worksheet:

<table>
<thead>
<tr>
<th>Field</th>
<th>Remarks</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable</td>
<td>When the value of the tag configured in this is TRUE, all alarms configured in this group are temporarily disabled. This option is useful to disable alarms under special conditions (e.g., during maintenance).</td>
<td>Tag</td>
</tr>
<tr>
<td>Total Alarms</td>
<td>The tag configured in this field, if any, is updated with the number of alarms from this group, which are currently active.</td>
<td>Tag</td>
</tr>
<tr>
<td>Total Unack</td>
<td>The tag configured in this field, if any, is updated with the number of alarms from this group, which are currently active AND have not been acknowledged yet.</td>
<td>Tag</td>
</tr>
<tr>
<td>Remote Ack Trigger</td>
<td>When the tag configured in this field change of value, all active alarms from this group are acknowledged. This option can be used to acknowledge alarms regardless of any action from the operator.</td>
<td>Tag</td>
</tr>
<tr>
<td>Dead Band Time &gt; Activation</td>
<td>Each alarm must remain continuously in its alarm condition for the period of time specified in this field before becoming active. This option is useful to avoid generating alarms on intermittent conditions (e.g., noise). If this field is left in blank, the alarm becomes active as soon as its condition is true.</td>
<td>Tag or Number</td>
</tr>
<tr>
<td>Dead Band Time &gt; Normalization</td>
<td>Each alarm must remain continuously out from its alarm condition for the period of time specified in this field before becoming normalized. This option is useful to avoid normalizing alarms on intermittent conditions (e.g., noise). If this field is left in blank, the alarm become normalized as soon as its condition is no longer true.</td>
<td>Tag or Number</td>
</tr>
</tbody>
</table>
| Dead Band Time > Time Stamp/Value | Each alarm maintains a time stamp of the last significant activity, along with the value of the tag at that time. You can select the type of activity that updates the time stamp:  
  • Activation/Norm (default): The time when the dead band ended — that is, when the alarm becomes activated or normalized.  
  • Last Tag Change: The time when the value of the tag last changed during the dead band.  
  • Start Condition: The time when the dead band started. | Combo   |

Alarm Worksheet Body

The following table describes the Body settings on an Alarm worksheet:

<table>
<thead>
<tr>
<th>Field</th>
<th>Remarks</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag Name</td>
<td>Name of the tag associated with the alarm.</td>
<td>Tag</td>
</tr>
</tbody>
</table>
| Type   | Type of the alarm:  
  • HIHI: Activates the alarm if the tag value is equal or higher than the limit.  
  • HI: Activates the alarm if the tag value is equal or higher than the limit. (For Boolean tags, if the value is 1.) | Combo-box |
### Field | Remarks | Syntax
--- | --- | ---
• **Lo**: Activates the alarm if the tag value is equal or lower than the limit. (For Boolean tags, if the value is 0.)
• **LoLo**: Activates the alarm if the tag value is equal or lower than the limit.
• **Rate**: Activates the alarm if the tag value varies faster than the rate specified to the alarm. (For Boolean tags, if the value changes.)
• **DevP**: Activates the alarm if the tag value is equal or higher than the Set Point tag plus the limit.
• **DevM**: Activates the alarm if the tag value is equal or lower than the Set Point tag minus the limit.

When using the types Rate, DevP and DevM, it is necessary to configure additional settings by the Tag Properties dialog (`Properties` on the Home tab of the ribbon).

<table>
<thead>
<tr>
<th>Limit</th>
<th>Limit associated with each alarm. The limits can be modified dynamically during runtime, by the tag fields HiHiLimit, HiLimit, LoLimit, LoLoLimit, Rate, DevP and DevM (e.g., <code>TagLevel-&gt;HiLimit</code>).</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>Message associated to the alarm. The message can be displayed on the Alarm/Event Control object and/or stored in the Alarm History and/or sent by Email, depending on the settings configured in the Header of the Alarm group.</td>
<td>Text and/or {Tag} (up to 256 chars)</td>
</tr>
<tr>
<td>Priority</td>
<td>Priority number associated to the alarm. When displaying alarms on the Alarm/Event Control object, the operator can filter and/or sort the alarms by priority.</td>
<td>Number (from 0 to 255)</td>
</tr>
<tr>
<td>Selection</td>
<td>Alias associated to the alarm (e.g., AreaA, AreaB, etc). When displaying alarms on the Alarm/Event Control object, the operator can filter and/or sort the alarms by their selection value.</td>
<td>Text (up to 7 characters)</td>
</tr>
<tr>
<td>Custom fields</td>
<td>Additional custom fields that will be saved in history. The number of custom fields (up to 10) can be set in <code>Project Settings: Options</code>.</td>
<td>Any</td>
</tr>
</tbody>
</table>

**Tip:** If you need more characters or a different data type, use a custom field instead.

> **Note:** You cannot configure more than one alarm of the same type on the same tag.

### Sort or Filter the Rows in a Worksheet

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item.

Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications.

Please note that you can sort or filter rows only in the following types of worksheets:
- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.
Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.
Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag3</td>
<td>Hi</td>
</tr>
<tr>
<td>Tag3</td>
<td>Lo</td>
</tr>
<tr>
<td>Tag3</td>
<td>LOLO</td>
</tr>
<tr>
<td>Tag3</td>
<td>HIII</td>
</tr>
</tbody>
</table>

*Alarm worksheet rows filtered where Tag Name is “Tag3”*

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag1</td>
<td>Lo</td>
</tr>
<tr>
<td>Tag2</td>
<td>Lo</td>
</tr>
<tr>
<td>Tag3</td>
<td>Lo</td>
</tr>
<tr>
<td>Tag3</td>
<td>HIII</td>
</tr>
</tbody>
</table>

*Alarm worksheet rows filtered where Type is “Lo”*

Tip: You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order. The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   Note: You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.

3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either Tab or Return.

   You may include * and ? as wildcard characters in your string:
   - * matches any number of characters, including none. For example, Tag* would match Tag, Tag3, Tag34567, TagA, and Tag_TEMP.
   - ? matches exactly one character. For example, Tag? matches Tag3 and TagA, while Tag?????? matches Tag34567 and Tag_TEMP.

   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.

4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either Tab or Return.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.
**Saving your alarm history / event log to an external database**

By default, your project’s alarm history and event log are saved to proprietary-format text files in your project’s Alarms folder. However, you can change your project settings to save them to an external SQL database instead.

**Note:** If your project was created with InduSoft Web Studio v7.1+SP1 or earlier and then upgraded to the current version, you should consider starting over with new database tables.

Beginning with InduSoft Web Studio v7.1+SP2, new database tables are automatically indexed by event time in order to improve runtime performance. Existing database tables cannot be indexed in this way, so if you can afford to discard that data, you should change your database configuration to create new tables.

If you do this, you must also manually edit your project file (`projectname.APP`) to add the following setting:

```
[Alarm]
AddEventTimeColumn=1
```

The setting exists for backward compatibility, and it defaults to 0 for projects that were upgraded from earlier versions.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**.
   The **Project Settings** dialog is displayed.

   ![Project Settings: Options](image)

2. In the **Alarm History and Events** area, in the **History Life Time** box, type the number of days of history that you want to save.
   As the history exceeds the specified number of days, it will be automatically deleted in a first-in, first-out manner. If no number is specified — that is, if it is left blank or set to 0 — then history will never be deleted. There is no limit to how much history you can save, but the more you save, the more disk space it will take.

3. From the **History Format** list, select **Database**.
4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), in the Default Database area, click Configure. The Default Database Configuration dialog is displayed. Use the dialog to configure the database connection. For more information, see Configuring a default database for all task history.

5. To configure a separate database for either your event log or your alarm history, click Event Database or Alarm Database, respectively. In either case, a Database Configuration dialog is displayed. Use the dialog to configure the database connection. For more information, see Database Configuration.

6. Click OK.

**Format of the alarm history**

The location and format of the alarm history depends on whether History Format is set to Proprietary or Database. This section describes both.

When the History Format is Proprietary (default), the alarm history is saved as a series of text files in your project’s Alarm folder, with one file per calendar day. The name of each file is ALyyymmdd.ALH, where:

- **yy** is the last two digits of the year in which the alarm history file was generated;
- **mm** is the month in which the alarm history file was generated; and
- **dd** is the day of the month on which the alarm history file was generated.

Therefore, the alarm history file for 07 May 2003 is located at [..]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Alarm\AL030507.ALH.

Tip: To change where your project saves these files, use the SetAppAlarmPath function.

Within a specific day’s *.ALH file, each alarm is saved as a new line, using the pipe character (|) to delimit the fields, as illustrated below:

```
  ...
```

When the History Format is Database, the alarm history is saved as a table in whichever database you configured for Alarm Database. For more information, see Saving your alarm history to an external database.

The fields/columns of the alarm history are described below:

<table>
<thead>
<tr>
<th>Proprietary</th>
<th>Database</th>
<th>Description</th>
<th>File Vers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Number</td>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>P1</td>
<td>—</td>
<td>—</td>
<td>File version (Current = 005)</td>
</tr>
<tr>
<td>P2</td>
<td>Al_Start_Time</td>
<td>TimeStamp</td>
<td>Start Date (MM/DD/YYYY)</td>
</tr>
<tr>
<td>P3</td>
<td>Al_Tag</td>
<td>String</td>
<td>Tag Name</td>
</tr>
<tr>
<td>P4</td>
<td>Al_Message</td>
<td>String</td>
<td>Alarm Message</td>
</tr>
<tr>
<td>P5</td>
<td>Al_Ack</td>
<td>Boolean</td>
<td>Ack, where:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: Alarm was acknowledged or does not require acknowledgment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Alarm was not acknowledged</td>
</tr>
<tr>
<td>P6</td>
<td>Al_Active</td>
<td>Boolean</td>
<td>Active, where:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: Alarm is not active</td>
</tr>
<tr>
<td>Field Number</td>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P8</td>
<td>Al_Tag_Value</td>
<td>Real</td>
<td>Tag Value when the alarm occurred</td>
</tr>
<tr>
<td>P9</td>
<td>Al_Group</td>
<td>Integer</td>
<td>Alarm Group Number</td>
</tr>
<tr>
<td>P10</td>
<td>Al_Priority</td>
<td>Integer</td>
<td>Priority Number</td>
</tr>
<tr>
<td>P11</td>
<td>Al_Selection</td>
<td>String</td>
<td>Selection</td>
</tr>
<tr>
<td>P12</td>
<td>Al_Type</td>
<td>Integer</td>
<td>Type, where:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: HiHi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 2: Hi(On)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 4: Lo(Off)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 8: LoLo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 16: Rate(Change)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 32: Deviation+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 64: Deviation-</td>
</tr>
<tr>
<td>P13</td>
<td>Al_Ack_Req</td>
<td>Boolean</td>
<td>Ack required, where:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: Alarm requires acknowledge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Alarm does not require acknowledge</td>
</tr>
<tr>
<td>P14</td>
<td>Al_Norm_Time</td>
<td>TimeStamp</td>
<td>Normalization Date (MM/DD/YYYY)</td>
</tr>
<tr>
<td>P15</td>
<td></td>
<td></td>
<td>Normalization Time (HH:MM:SS.mmm)</td>
</tr>
<tr>
<td>P16</td>
<td>Al_Ack_Time</td>
<td>TimeStamp</td>
<td>Ack Date (MM/DD/YYYY)</td>
</tr>
<tr>
<td>P17</td>
<td></td>
<td></td>
<td>Ack Time (HH:MM:SS.mmm)</td>
</tr>
<tr>
<td>P18</td>
<td>Al_User</td>
<td>String</td>
<td>User Name</td>
</tr>
<tr>
<td>P19</td>
<td>Al_User_Comment</td>
<td>String</td>
<td>Comment</td>
</tr>
<tr>
<td>P20</td>
<td>Al_User_Full</td>
<td>String</td>
<td>User Full Name</td>
</tr>
<tr>
<td>P21</td>
<td>Al_Station</td>
<td>String</td>
<td>Station</td>
</tr>
<tr>
<td>P22</td>
<td>Al_Prev_Tag_Value</td>
<td>Real</td>
<td>Previous Value</td>
</tr>
<tr>
<td>P23</td>
<td>Bias</td>
<td>Integer</td>
<td>Time Zone Bias</td>
</tr>
<tr>
<td></td>
<td>Al_Start_Time_ms</td>
<td>Integer</td>
<td>Number of milliseconds for the Start Time timestamp. This field is used when the database does not support ms in a TimeStamp field.</td>
</tr>
<tr>
<td></td>
<td>Al_Norm_Time_ms</td>
<td>Integer</td>
<td>Number of milliseconds for the Norm Time timestamp. This field is used when the database does not support ms in a TimeStamp field.</td>
</tr>
<tr>
<td></td>
<td>Al_Ack_Time_ms</td>
<td>Integer</td>
<td>Number of milliseconds for the Ack Time timestamp. This field is used when the database does not support ms in a TimeStamp field.</td>
</tr>
<tr>
<td></td>
<td>Al_Deleted</td>
<td>Boolean</td>
<td>Deleted, where:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: Alarm message was not deleted by the user (not visible).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Alarm message was deleted by the user (visible).</td>
</tr>
<tr>
<td></td>
<td>Last_Update</td>
<td>TimeStamp</td>
<td>Timestamp of the last update for this alarm.</td>
</tr>
</tbody>
</table>
Tip: You can customize the names of the columns in the database table by manually editing the project file (e.g., [...]\Documents\InduSoft Web Studio v7.1 Projects \projectname\projectname.APP) as follows:

```plaintext
[Alarm]
DefaultName=NewName
```

For example:

```plaintext
[Alarm]
Message=Alarm Message
Ack=Acknowledgment
```
About events and event logging

This section describes IWS’s logging and event-retrieval features. An event can be any tag change, generating reports or recipes, opening and closing screens, logging onto and logging off the security system, and so forth. IWS saves all of these events in a log file, which can then be retrieved by the Alarm/Event Control object.

**Enabling the event logger**

Event logging is disabled by default, to conserve runtime resources. To enable the saving of events to the history file, use the Event Logger in the Project Explorer.

1. In the Project Explorer, on the Global tab, double-click Event Logger. The Event Settings dialog is displayed.

![Event Settings dialog](image)

2. Select Enable event logger.

3. In the Disable box, type the name of a project tag. Whenever the value of the tag is TRUE (i.e., non-zero) during runtime, event logging will be suspended.

4. In the Settings area, select which types of events that you want to log to the history file.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Security System | Events generated by your project’s security system, including:  
- Log On / Log Off users  
- User created/removed by calling the CreateUser or RemoveUser functions  
- User blocked/unblocked by calling the BlockUser or UnblockUser functions  
- User blocked by the security system after several attempts to enter an invalid password  
- Password expired  
- Password modified  
- Invalid Log On attempt |
<p>| Display   | Open Screen and Close Screen events.                                                                                                          |
| Recipe    | Recipes loaded, saved, initialized, or deleted.                                                                                               |
| Report    | Reports saved to disk or sent to printer.                                                                                                       |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Messages</td>
<td>Events generated by calling the <code>SendEvent</code> function.</td>
</tr>
<tr>
<td>System Warning</td>
<td>Various runtime warnings and errors, including:</td>
</tr>
<tr>
<td></td>
<td>• Errors that occur when sending alarms by email</td>
</tr>
<tr>
<td></td>
<td>• Tag was blocked/unblocked</td>
</tr>
<tr>
<td></td>
<td>• Division by zero</td>
</tr>
<tr>
<td></td>
<td>• Connection/Disconnection of the remote security system</td>
</tr>
</tbody>
</table>

5. To log changes in specific project tags, select **Tags**, and then in the table, specify the tags.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag Name</td>
<td>The name of the project tag that you want to log to the history file.</td>
</tr>
</tbody>
</table>
| Dead Band| A value to filter changes against, so that only changes greater than this value are logged.  
For example, if you specify a **Dead Band** value of 5 for a tag value of 50 and the tag value changes to 52, then the system will not register this variation in the event log, because the change is less than 5. However, if the tag value change is equal to or greater than 5, then the system will log the new value to the history file. |
| Message  | A string (message) related to this tag change. You can specify tags in messages using the `{tagname}` syntax. |

The **Tags** option is useful for logging events that are not important enough to be alarm conditions (for example, Motor On, Motor Off, and so on).

6. Click **OK**.

By default, the event log is saved as a series of text files in your project’s Alarms folder. For more information, see **Format of the event log**.

Alternatively, you can save the event log to an external SQL database. For more information, see **Saving your event log to an external database**.

**Saving your alarm history / event log to an external database**

By default, your project’s alarm history and event log are saved to proprietary-format text files in your project’s Alarms folder. However, you can change your project settings to save them to an external SQL database instead.

---

**Note:** If your project was created with InduSoft Web Studio v7.1+SP1 or earlier and then upgraded to the current version, you should consider starting over with new database tables.

Beginning with InduSoft Web Studio v7.1+SP2, new database tables are automatically indexed by event time in order to improve runtime performance. Existing database tables cannot be indexed in this way, so if you can afford to discard that data, you should change your database configuration to create new tables.

If you do this, you must also manually edit your project file (`projectname.APP`) to add the following setting:

```
[Alarm]
AddEventTimeColumn=1
```
The setting exists for backward compatibility, and it defaults to 0 for projects that were upgraded from earlier versions.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**. The **Project Settings** dialog is displayed.

![Project Settings: Options](image)

2. In the **Alarm History and Events** area, in the **History Life Time** box, type the number of days of history that you want to save.

   As the history exceeds the specified number of days, it will be automatically deleted in a first-in, first-out manner. If no number is specified — that is, if it is left blank or set to 0 — then history will never be deleted. There is no limit to how much history you can save, but the more you save, the more disk space it will take.

3. From the **History Format** list, select **Database**.

4. To configure a single, default database to be used for both the alarm history and the event log (as well as all other runtime tasks), in the **Default Database** area, click **Configure**. The **Default Database Configuration** dialog is displayed. Use the dialog to configure the database connection. For more information, see Configuring a default database for all task history.

5. To configure a separate database for either your event log or your alarm history, click **Event Database** or **Alarm Database**, respectively.

   In either case, a **Database Configuration** dialog is displayed. Use the dialog to configure the database connection. For more information, see Database Configuration.

6. Click **OK**.

**Format of the event log**

The location and format of the event log depends on whether History Format is set to Proprietary or Database. This section describes both.

When the History Format is Proprietary (default), the event log is saved as a series of text files in your project’s Alarm folder, with one file per calendar day. The name of each file is `EVyyymmdd.EVT`, where:

- **yy** is the last two digits of the year in which the event log file was generated;
- **mm** is the month in which the event log file was generated; and
• \textit{dd} is the day of the month on which the event log file was generated.

Therefore, the event log file for 07 May 2003 is located at [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Alarm\EV030507.EVT.

**Tip:** To change where your project saves these files, use the \texttt{SetAppAlarmPath} function.

Within a specific day’s *.EVT file, each event is saved as a new line, using the pipe character (|) to delimit the fields, as illustrated below:

\[
\begin{align*}
P1 & | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 \\
P1 & | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 \\
\vdots & \\
P1 & | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 \\
\end{align*}
\]

When the History Format is Database, the event log is saved as a table in whichever database you configured for Event Database. For more information, see Saving your event log to an external database.

The fields/columns of the event log are described below:

<table>
<thead>
<tr>
<th>Proprietary</th>
<th>Database</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Number</td>
<td>Column Name</td>
<td>Data Type</td>
</tr>
<tr>
<td>P1</td>
<td>Version</td>
<td>Integer</td>
</tr>
<tr>
<td>P2</td>
<td>Event_Type</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Event_Time</td>
<td>TimeStamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>Event_Info</td>
<td>String</td>
</tr>
<tr>
<td>P5</td>
<td>Value</td>
<td>Real</td>
</tr>
<tr>
<td>P6</td>
<td>Source</td>
<td>String</td>
</tr>
<tr>
<td>P7</td>
<td>User</td>
<td>String</td>
</tr>
<tr>
<td>P8</td>
<td>User_Full</td>
<td>String</td>
</tr>
<tr>
<td>P9</td>
<td>Message</td>
<td>String</td>
</tr>
<tr>
<td>P10</td>
<td>Station</td>
<td>String</td>
</tr>
<tr>
<td>P11</td>
<td>Comment</td>
<td>String</td>
</tr>
<tr>
<td>P12</td>
<td>Previous_Value</td>
<td>Real</td>
</tr>
<tr>
<td>—</td>
<td>Deleted</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Page 295
<table>
<thead>
<tr>
<th>Field Number</th>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P13</td>
<td>Bias</td>
<td>Integer</td>
<td>Difference (in minutes) between Event_Time and the GMT time. This field only exists for Version &gt;= 2.</td>
</tr>
<tr>
<td>—</td>
<td>Last_Update</td>
<td>TimeStamp</td>
<td>Time stamp indicating when the register was created/modified. This field is used to synchronize the databases when using the Secondary Database in addition to the Primary Database. This field is created only when the History Format is Database.</td>
</tr>
</tbody>
</table>

**Tip:** You can customize the names of the columns in the database table by manually editing the project file (e.g., `\[EventLogger\]
DefaultName=NewName` as follows:

```plaintext
[EventLogger]
DefaultName=NewName
```

For example:

```plaintext
[EventLogger]
Event_Info=Information
Message=Event_Message
```
Alarm/Event Control object

Use the Alarm/Event Control tool to add an Alarm or Event Control object to a project screen.

To create and configure an Alarm/Event Control object:
1. On the Graphics tab of the ribbon, in the Data Objects group, click Alarm/Event Control.
2. Click in the display, and drag the mouse to create and adjust the object's shape.

**Note:** This object cannot be moved behind active objects or shapes on a project screen. In other words, it must always be in front. For more information, see Bring to front / Send to back on page 249.

3. Double-click on the object to open the following Object Properties dialog box.

![Object Properties: Alarm/Event Control](image)

You can use this dialog box to specify the following parameters:

- Select an alarm object mode in the Type group:
  - **Alarm Online**: Display only current alarm messages, as configured in your Alarm worksheets.
  - **Alarm History**: Display only alarm messages from the Alarm History database.
  - **Alarm History + Event**: Display both alarm messages from the Alarm History database and logged events from the Event History database.
  - **Event**: Display only logged events from the Event History database.

- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.

- Click (enable) the **Enable translation** checkbox to enable the external translation of messages using the Translation Table. (For more information, see Project Localization on page 496.)

- **VK**: Virtual Keyboard type used for this object. You need to select the Virtual Keyboard option in the Viewer settings (Viewer on the Project tab of the ribbon) before configuring the Virtual Keyboard for this interface.
Filters
To filter alarm messages during runtime, click the Filters button. The Filters dialog box displays so you can specify filtering parameters for the Alarm Control object.

- Use the **Group** field to filter messages by the Alarm group/worksheet number. The worksheets are organized in the Alarms folder, in the Tasks tab of the Project Explorer, starting with 1. If you specify a Group of 0, then all of the worksheets will be displayed. You can use commas and/or dashes to specify a range of groups (e.g., 1, 3, 5–6).
- Use the **Selection** field to filter messages by the Selection text configured on the Alarm worksheet.
- In the **Priority** group, use the From and To fields to filter messages by the Priority configured on the Alarm worksheet. Type numerical values into these fields to delimit the priority range.
- Use the **Type** field to filter messages by the alarm type (e.g., HiHi, Hi, Lo, LoLo, Rate, Dev+, Dev-). You can use commas to specify more than one type; for example, HiHi, LoLo.
- Use the **State** field to filter messages by the alarm status:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All alarms (default)</td>
</tr>
<tr>
<td>1</td>
<td>All active and unacknowledged alarms</td>
</tr>
<tr>
<td>2</td>
<td>All active and acknowledged alarms</td>
</tr>
<tr>
<td>3</td>
<td>All inactive and acknowledged alarms</td>
</tr>
<tr>
<td>4</td>
<td>All inactive and unacknowledged alarms</td>
</tr>
</tbody>
</table>

Leaving this field blank is effectively the same as entering a value of 0.

- In the **Search in columns** group, use the **Tagname**, **Message**, and/or **Username** text fields to specify criteria for filtering messages. Type a tagname, message, and/or user name into the text field for which you want IWS to search.
- Use the settings in the **Interval** group to filter messages by the last x messages (Latest) or based on a period of time (Period). If you do not specify any interval at all, then only the alarms for the current day will be displayed.

**Note:**
- You can specify String tags in curly brackets (e.g., \{tagname\}) in the **Group**, **Selection**, **Tagname**, **Message**, and **Username** fields, to change these values during runtime.
You must specify String tags *without* curly brackets (e.g. `tagname`) in the **Type** field and the **Period** fields of the **Interval** group. These fields cannot take values directly.

You can specify Integer tags in the **From** and **To** fields **Priority** group, the **State** field, and the **Latest** field from the **Interval** group.

You can use wildcards ( * and ?) when specifying values for the **Selection**, **Tagname**, **Message**, and **Username** fields.

Use the **Filter Expression** to configure an expression that will filter unwanted messages out of the display. Only messages that satisfy the expression will be shown.

To enter a filter expression, click **Edit**; the **Alarm Filter Expression** dialog box is displayed. The filter expression must follow the basic syntax of...

```
[columnname]operator'value'
```

...where the *columnname* is the name of a column in the Alarm/Event Control object and *operator* is any of the standard relational operators (e.g., =, <, >, <=, >=, <>). For example:

```
[Activation Time]>'08/17/2007 15:00'
```

This filter will only show alarm messages with activation times greater (later) than 15:00 on 08/17/2007.

**Tip:** It is not necessary to use square brackets when *columnname* is one word (e.g., `Value`), but doing so makes the filter expression easier to read.

You can combine several conditions simultaneously by using the logical operators AND, OR, and NOT. For example:

```
[Custom1]='MyCustomArea' AND [State]=0
```

```
[Type]='HiHi' OR [Type]='LoLo' AND [Activation Time]>'08/17/2007 15:00'
```

You can use wildcards (* and ?) in the filter expression. You can also change the filter expression during run time by specifying project tags and/or built-in functions in curly brackets. For example:

```
[Value]='{AlarmFilterValue}'
```

```
[ActivationTime]<='{DateTime2UTC(Date + " " + Time)}'
```

Please note that filtering by time works only if the Alarm/Event Control object is configured to show alarm history, and all times should be specified in Coordinated Universal Time (UTC), because that is how the alarm history is saved. If you need to convert between local time and UTC, you can use the **Date & Time functions**.

The **Alarm Filter Expression** dialog box has a limit of 1024 characters. You can configure a filter expression of up to 2048 characters, however, by using the curly brackets syntax described above. The contents of the brackets will be evaluated during run time. So, for example, you can specify two or more project tags (String type) that contain parts of the overall expression and then use AND operators to combine the parts:

```
{MyFilterExpression1} AND {MyFilterExpression2}
```
• Use the settings in the Initial Sort group to set the default sorting order. Select a sort type from the Column combo-box, and then select Asc or Desc to sort in ascending or descending order. You can configure up to three levels of sorting.

**Note:** If you configure all three levels with sort types other than Activation Time, then the project will automatically sort on a fourth level according to Activation Time, in descending order.

You cannot change the type of this fourth-level sort, but you can toggle its default order — from descending to ascending — by manually editing your project file [projectname.app] to change the following setting:

```
[Objects]
DescendingAlarmListTime=TRUE or FALSE
```

TRUE sorts in descending order, FALSE sorts in ascending order. Please note that this setting only works for projects created with or updated to InduSoft Web Studio v6.1+SP5 or later.

Click the Allow sort in runtime checkbox if you want to allow the user to change the sort order during runtime.

**Columns**

Click the Columns button to open the Columns dialog box where you can specify display properties for columns in the object.

![Columns dialog box](image)

• The Available list contains all of the column types available for this object. The Visible list contains all of the column types currently in use for the object.

Click the » and « buttons to move selections between the two lists.

**Tip:** You can configure an Alarm Control object to display recently replaced values together with their new values. To do so, move both Value and Previous to the Visible list.

Click the Move Up or Move Down buttons to rearrange the order of columns in the Visible list.

• Use the Label and Width fields in the Properties group to change the default column labels and widths at runtime.

• Use the Align combo box to specify alignment (Left, Center, or Right) for the alarm message text within a specified column.

• Click (enable) the Available during runtime checkbox to allow the user to add selected columns to the visible list during runtime.

• Use the Key box to assign a shortcut to each column. This allows you to sort the information on the Alarm Control object by any column, using keyboard keys instead of the mouse cursor.

When you are finished, click OK to close the Column dialog box.
**Tip:** You can associate text labels with priority values, so that more meaningful information is displayed in the Priority column of the Alarm Control. To do this, manually edit your project file (projectname.APP) to add the following entries:

```
[Alarm]
PriorityLabelCount=N (total number of labels)
PriorityValue1=value
PriorityLabel1=label
...
PriorityValueN=value
PriorityLabelN=label
```

Here is an example:

```
[Alarm]
PriorityLabelCount=3
PriorityValue1=490
PriorityLabel1=ALMTest
PriorityValue2=480
PriorityLabel2=Test2
PriorityValue3=470
PriorityLabel3=Test100
```

**Advanced**

Click the **Advanced** button to open the **Advanced** dialog box where you can specify advanced properties for the Alarm Control object.

![Advanced dialog box](image)
• Use the settings in the **Date & Time Format** group to control which date and time information displays in the alarm message. Click (enable) a checkbox to include that element in the display. Note: **MS** stands for milliseconds.

**Tip:** Watch the **Sample** text to preview how the information will look in the alarm message.

• Use the settings in the **Delete Message** group to control who can delete alarm messages from the Alarm History:

  • **Security**: Use this field to specify which security level can delete alarm messages. Only those users with the specified security level will be allowed to delete an alarm message.
  
  • **Confirm**: Select this option to require the user to confirm a message deletion before IWS actually deletes the selected alarm message.

• Use the settings in the **Acknowledgement** group to control how alarms are acknowledged:

  • **Ack All trigger** box: Type a tag to receive a value. When the tag value changes, it indicates that all messages in the alarm object have been acknowledged.
  
  • **Ack trigger** box: Type a tag to receive a value. When the tag value changes, it indicates that the message at the top of the alarm object has been acknowledged.
  
  • **Ack comment** list: Select **Disabled**, **Optional**, or **Mandatory** to determine whether the user can or must enter comments after acknowledging alarms.
  
  • **Disable ack on double click** box: Type either a numeric value or a tag/expression. When it evaluates as **TRUE** (non-zero), the user cannot acknowledge alarms by double-clicking (or double-tapping) them. This option may be used, for example, either to force the operator to click another button to acknowledge the alarm or to prevent alarms from being acknowledged on thin clients.
  
  • **Security** box: Type a numeric value to specify which security level can acknowledge an alarm message. Only those users with the specified level can respond.
  
  • **Require confirmation** checkbox: Select this option to display a confirmation dialog when the user tries to acknowledge a single alarm.

• Use the settings in the **Run-time returned values** group to get information about the alarms during run time:

  • **Total items** box: Type an integer tag to see how many alarms remain after IWS filters the alarm object using parameters specified on the **Filters** dialog box.
  
  • **Selected tag** box: Type a string tag to enable the end user to click on an alarm message to see the name of the tag associated with that message.
  
  • **First Row Text** box: Type the name of a project tag or array (String type). That tag or array will receive the contents of the columns of the first row of the Alarm/Event Control. If you specify a tag, the columns will be separated by tabs. If you specify an array, the array elements will each receive one column. If the array is not large enough to receive all of the columns, the remaining columns will be discarded. Whenever the first row changes — either due to a new Alarm/Event, or simply because the rows are reordered — the specified array is updated.
  
  • **Selected Row Text** box: Type the name of a project tag or array (String type). That tag or array will receive the contents of the columns of the selected row (i.e., the row that the user has clicked/tapped) of the Alarm/Event Control. If you specify a tag, the columns will be separated by tabs. If you specify an array, the array elements will each receive one column. If the array is not large enough to receive all of the columns, the remaining columns will be discarded. Whenever the selected row changes — that is, whenever the user clicks/taps another row — the specified array is updated.
  
  • **Summary Changes** box: Type the name of a project tag (Integer type). That tag will receive a running count of the number of changes in the Alarm/Event Control. For example, when a new Alarm occurs or when an Alarm is acknowledged, the value of the configured tag will be incremented. Reordering the rows is not counted as a change.

• Use the settings in the **Run-time dialog triggers** group to allow the user to customize the object during run time:

  • **Columns** box: Type a tag name. When the tag value changes, it opens a dialog box allowing the user to customize the columns visible in the object.
  
  • **Filters** box: Type a tag name. When the tag value changes, it opens a dialog box allowing the user to filter the columns visible in the object.
• **Auto Format** checkbox: When checked, decimal values in the **Display Value**, **Previous** and **Value** columns of the object will be formatted according to the virtual table created by the function **SetDecimalPoints**.

• Use the **Save / Print** group to control the printing of alarms during run time:
  
  • **Print Trigger**: When the tag configured in this field is toggled, the current state of the Alarm/Event Control object is sent to the default printer.
  
  • **PDF Trigger** box: When the tag configured in this field is toggled, the current state of the Alarm/Event Control object is saved as a PDF file at the location specified by **PDF Filename**.
  
  • **PDF Filename** box: Enter a complete file path and name where the PDF file is to be saved. You can also enter a tag name using the `{tag}` syntax.

  **Note:** **PDF Trigger** and **PDF Filename** are not supported in projects running on Windows Embedded or Thin Client.

• **Multiline** checkbox: When this option is checked, the print output or PDF will be formatted according to the available column space, and the text within each cell will be wrapped so that all of it is shown.

• Click the **Navigation Triggers** button to open the following dialog box:

  ![Navigation Triggers dialog box](image)

  You can make the on-screen Alarm Control object scroll up, scroll down, page up, page down, go to home (beginning) of page, or go to end of page by configuring tags in the corresponding fields. Whenever the values of the configured tags change, the Alarm Control object will navigate that way. This is useful for adding navigation controls to the screen; for example, if you configure the same tag to the **Up** field in this dialog box and a **PushButton object**, then the Alarm Control object will scroll up whenever the Pushbutton object is pressed.

  When you are finished, click **OK** to close the **Advanced** dialog box.

**Fonts**

Click the **Fonts** button to open a **standard Fonts interface** where you can specify display properties for the message text.

**Format**

Use the **Format** area of the object properties to configure the appearance of the Alarm/Event Control object during run time:

• Select **Show header** to show a header on the object. The header displays the column labels.

  ![Showing the header](image)
• Select **Show gridlines** to show gridlines in the object. Gridlines can make it easier to distinguish individual rows and columns in the object.

![Showing the gridlines](image)

• Use the **Background color** box to select a background color for the object. Click the box to open the color palette pop-up, then click a color to select it.

• Click **Colors** to open the **Colors** dialog box, which you can use to select background and foreground colors for specific alarms. These colors will override the default colors that you selected in your Alarm worksheet(s). This is useful for highlighting special alarms.

![Colors dialog box](image)

**Note:** This feature is not supported in projects running on Windows Embedded target systems.

In each row of this table, you can configure a subset of alarms using similar criteria as in the **Filters** dialog box, and then for that subset you can select custom background and foreground colors:

1. In the **Group** box, type the number of the Alarm group/worksheet. You may use commas and/or dashes to specify a range of groups.

2. In the **Active** box, select **All** (both active and normalized alarms), **Active** (active alarms only), or **Norm** (normalized alarms only).

3. In the **Ack** box, select **All** (both acknowledged and unacknowledged alarms), **Acked** (acknowledged alarms only), or **Unacked** (unacknowledged alarms only).

4. In the **Selection** box, type the selection text that you configured in the Alarm worksheet. You may leave this box empty.

5. In the **Priority** box, type the priority number that you configured in the Alarm worksheet. You may leave this box empty.

6. Click the **BG Color** box to open a color picker, and then select the color that you want to be the background color of the alarm message.

7. Click the **FG Color** box to open a color picker, and then select the color that you want to be the foreground color of the alarm message.

8. Select **Blink** if you want the alarm message to blink. This will make it more noticeable.

Please note that the subsets you configure here must pass any filters that you previously configured in the **Filters** dialog box. For example, if you configured the filters to show groups 1–6 in the Alarm/Event Control object, then configuring a subset of group 7 here will have no effect.

Also, the colors that you select will be used to indicate all possible alarm states (i.e., activation, acknowledgement, and normalization), if you configure the subset to include all of those states. Therefore, if you want different colors for different alarm states, then you must configure additional subsets.
The Trend folder enables you to configure history groups that store trend curves. You can use the Trend worksheet to declare which tags must have their values stored on disk, and to create history files for trend graphs. The project stores the samples in a binary history file (*.hst), and shows both history and on-line samples in a screen trend graph.

The Trend worksheet is executed by the Background Task module (see Execution Tasks). It handles the saving of trend data to the history, but it does not display that data to the operator; the Trend Control screen object, available on the Graphics tab of the ribbon, must be created and configured in a screen in order to display trend data.

To create a new Trend worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click Trend;
- Right-click the Trends folder in the Project Explorer, and then click Insert on the shortcut menu; or
- Click New on the Application menu, click the File tab, and then select Trend Worksheet.

To edit an existing Trend worksheet, double-click it in the Project Explorer.

**Trend worksheet**

The Trend worksheet is divided into two areas:

- Header area (top section), which contains information for the entire group
- Body area (bottom section), where you define each project tag in the group. This section contains several columns (only two are shown in the preceding figure).

**Header**

Configure the following settings in the worksheet header:

- **Description**
  Type a description of the worksheet for documentation purposes.

- **History Format**
  Click the arrow button to select a trend history format from the list. The available options are:
  - **Proprietary**
    Save trend history in a proprietary, binary file. The file is saved in your project folder (on the project runtime server) at: [%projectname%\Hst\GGYYMMDD.hst
  
  - **GG** = Trend worksheet number
  - **YY** = Last two digits of the year
A new history file is created for each calendar day that the project runs.

The utility programs HST2TXT.EXE and TXT2HST.EXE are provided in order to convert history files from binary (*.hst) to plain text (*.txt) and vice versa. For more information, see Converting Trend History Files from Binary to Text on page 311 and Converting Trend History Files from Text to Binary on page 312.

**Database**

Save trend history in an external SQL database of your choice. After you select this format, click Database Configuration to open the Database Configuration dialog box, where you can configure the connection to the database. For more information, see Database Configuration on page 98 and Database Interface on page 605.

By default, the history is saved in the table TRENDGGG (GGG = Trend Worksheet Number; e.g., TREND001 for the Trend Worksheet 001).

**Wonderware Historian**

Save trend history in a Wonderware Historian database of your choice. After you select this format, click Historian Configuration to open the Wonderware Historian dialog box, where you can configure the connection to the database. For more information, see Connect to a Wonderware Historian database on page 310.

The trend history for each project tag is saved separately in the Historian database, but you can use Prefix in the connection settings in order to keep the tags grouped together.

![Note:](image)

**Save On Trigger**

Click (enable) and type a tag name to save trend samples when someone changes the specified tag. (Tag change can be an event from the Scheduler.)

**Save On Tag Change**

Click (enable) to always save the trend sample when a value change occurs in any of the tags from that group.

**Advanced**

Click to display the Trend Advanced Settings dialog. For information about completing the fields in this window, see Batch History Configuration.

**Body**

For each project tag, configure the following settings in the worksheet body:

**Tag Name**

The name of the project tag for which trend history will be saved.

**Dead Band**

Type a value to filter acceptable changes when Save on Tag Change is used. For example, Dead Band has value = 5. If the tag value is 50 and changes to 52, the system will not register this variation in the database, because it is less than 5. If the change is equal to or greater than 5, the new value will be saved to the history file.

**Field**

When History Format is Database, this is the name of the field (in the SQL database table) where the trend history will be saved. If this field is left blank, the project tag name will be used.
For array tags and classes, special characters ([ ] .) will be replaced by underscores (_), as shown in the examples below:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyArray[1]</td>
<td>MyArray_1</td>
</tr>
<tr>
<td>MyClass.Member1</td>
<td>MyClass_Member1</td>
</tr>
<tr>
<td>MyClass[3].Member2</td>
<td>MyClass_3_Member2</td>
</tr>
</tbody>
</table>

**Historian Tag**

When **History Format** is **Wonderware Historian**, this is the name of the tag (in the Historian database) where the trend history will be saved. If this field is left blank, the project tag name will be used.

**Note:** The Trend task can accept only up to 240 tags in a single worksheet. If you manually configure more than 240 tags in the same worksheet, the Trend task will generate an error when you run the finished project.

**Sort or filter the rows in a worksheet**

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item.

Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications.

Please note that you can sort or filter rows only in the following types of worksheets:

- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.
Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.
Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.

**Tip:** You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order. The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   **Note:** You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.

3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either Tab or Return.

   You may include * and ? as wildcard characters in your string:
   - * matches any number of characters, including none. For example, Tag* would match Tag, Tag3, Tag34567, TagA, and Tag_TEMP.
   - ? matches exactly one character. For example, Tag? matches Tag3 and TagA, while Tag?????? matches Tag34567 and Tag_TEMP.

   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.

4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either Tab or Return.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.
Connect to a Wonderware Historian database

Configure a connection to a Wonderware Historian database in order to save historical data.

Before you begin this task, you should have the address or host name of the Wonderware Historian database server, as well as appropriate credentials (i.e., user name and password) to access that database. You must also have a task worksheet (e.g., a Trend worksheet) open for editing.

To configure the connection to a Wonderware Historian database:

1. In the header of the task worksheet, make sure Wonderware Historian is selected as the history format, and then click Historian Configuration. The Wonderware Historian dialog box is displayed.

2. In the Server box, type the host name or IP address and port number of the Historian database server. For example, HistorianDBServer:123. The port number is optional. If you did not change the port number in the Historian database settings, you can omit it here and the default (port 32568) will be used. Otherwise, the port number must match the one that is specified in the Historian database settings.

   Also, you can configure a tag name in curly brackets (e.g., {MyServer}) in order to change this setting during run time. See "Reload" below.

3. In the User and Password boxes, type your credentials to access the Historian database. You can configure a tag name in curly brackets for User (e.g., {MyUser}) in order to change this setting during run time. See "Reload" below.

4. In the Prefix box, type a prefix that will be added to the tags saved in the Historian database in order to keep them grouped together. For example, if you are configuring the connection for Trend worksheet TREND001, you could make that the prefix as well.

   This setting is optional, but if you do not specify a prefix, the tags will be not be grouped together in the database and therefore might be difficult to find. Also, you can configure a tag name in curly brackets (e.g., {MyPrefix}) in order to change this setting during run time. See "Reload" below.

5. Select the Enable store and forward check box if you want to store historical data when the connection is unavailable and then forward the stored data when the connection is restored.

   The historical data are saved in your project folder (on the project runtime server). Please note that this can consume a large amount of drive space if the connection is unavailable for a long time.

6. In the Run-time group, in the Status box, type the name of a project tag (Integer type) that will receive codes describing the status of the connection during run time.

   This setting is optional.

7. In the Run-time group, in the Reload box, type the name of a project tag that will trigger reloads of the connection settings.
When the value of this tag changes, the connection settings above will be reloaded. For those settings, you can configure tag names in curly brackets (e.g., {MyServer}) in order to change the settings during runtime. The connection to the Historian database is updated only when the settings are reloaded, and the settings are reloaded only when the value of the specified tag changes. This is to make sure that all of the settings are changed together rather than one at a time.

8. In the Gateway group, in the Host and Port boxes, type the host name or IP address and port number of the ADO.NET database gateway (StADOSvr).

   The default value for Host is 127.0.0.1 (a.k.a. localhost), because the project runtime server and the database gateway are typically hosted on the same computer.

9. Click OK to save the settings and close the dialog box.

   As mentioned above, the ADO.NET database gateway is typically hosted on the same computer that hosts the IWS project runtime server and does not require additional configuration. However, there are some things to consider when you develop your IWS project to connect to a Wonderware Historian database. The Historian database interface requires Microsoft .NET Framework 4, which is not included in older versions of Microsoft Windows. Therefore, when you run your project, it might not be able to connect to the Historian database. If that happens, check whether the framework is installed and turned on (Control Panel > Programs > Programs and Features > Turn Windows features on or off), and if it is not, do one of the following:

   • Download and install the framework (www.microsoft.com/download/details.aspx?id=17718); or
   • Install and run the database gateway on another computer — perhaps the computer that hosts the Historian database server itself — and then change the Gateway settings above to specify that computer.

The first option might not be feasible if the computer is a Windows Embedded device; depending on how the device is configured and what version of Windows Embedded it runs, it might not be possible to install the framework separately. In many cases, the device’s OS image must be configured to include the framework, and that is typically done by the manufacturer of the device. Please consult the manufacturer’s documentation.

If you choose the second option, you must include the Historian client library (aahClientManaged.dll) in the same folder with the other database gateway files. For more information about how to install and run the database gateway on another computer, see Studio Database Gateway on page 609.

**Converting Trend History Files from Binary to Text**

By default, IWS saves trend history files in a binary format (.hst). Because you may want to have these files in .txt format, IWS provides the HST2TXT.EXE program to convert trend history files from binary into text format.

To convert a file, use the following procedure:

1. From a DOS window, change directory (cd) to the IWS \Bin directory:

   cd C:\Program Files\InduSoft Web Studio v7.1\Bin

2. At the command prompt, copy the Hst2txt.exe into the same directory where the .hst file is located.

3. At the command prompt, type Hst2txt.exe and specify the following parameters:

   • *filename*: Name of the trend history file to convert
   • *separator*: Data separator character (default is <TAB>)
   • */e*: Extended functionality (convert data with more than 10 characters)
   • */i:HH:MM:SS*: Start time in hours (HH), minutes (MM), and seconds (SS)
   • */f:HH:MM:SS*: Finish time in hours (HH), minutes (MM), and seconds (SS)
   • */m*: Include milliseconds in the Time column (Type 1 to print the milliseconds value in the text file created from the .hst file.)

   For example:

   Hst2txt.exe 01952010.hst
The program creates a .hdr (header) file and the .txt file, which are both plain text files that can be viewed using any text editor (for example, Notepad).

- The .hdr file contains the name of the tags configured in the Trend Worksheet.
- The .txt file contains the tag values saved in the history file.

4. After the program converts the file, type **Exit** to close the DOS window.

**Note:** Alternatively, you can use the HST2TXT math script in a *Math* worksheet to convert binary files into text format automatically without having to use a DOS window.

### See also:
- Converting Trend History Files From Text to Binary
- Creating Batch History
- Configuring a Default Database for All Task History

#### Converting Trend History Files from Text to Binary

IWS provides the **TXT2HST.EXE** program to convert text files back into binary format.

To convert a file, use the following procedure:

1. From a DOS window, change directory (cd) to the IWS \Bin directory:

```
   cd C:\Program Files\InduSoft Web Studio v7.1\Bin
```

2. At the command prompt, copy the **Txt2hst.exe** into the same directory where the .txt file is located.

3. At the command prompt, type **Txt2hst.exe** and specify the following parameters:

   - **filename**: Name of the ASCII file with history data to convert
   - **[separator]**: Data separator character (default is <TAB>)
   - **[/e]**: Extended functionality (data value with more than 10 characters)
   - **[/i:HH:MM:SS]**: Start time of data value in hours (HH), minutes (MM), and seconds (SS)
   - **[/f:HH:MM:SS]**: Finish time of data value in hours (HH), minutes (MM), and seconds (SS)

   For example:

   ```
   Txtt2hst.exe 02950201.txt
   ```

   The program creates a .hdr (header) file and converts the .txt file into a .hst binary file.

4. After the program converts the file, type **Exit** to close the DOS window.

**Note:** You cannot create a math script for the **TXT2HST.EXE** program and use it in a *Math* worksheet to convert text files into binary format as you can for **HST2TXT.EXE**. The math script shortcut is available for binary files only.

### See also:
- Converting Trend History Files from Binary to Text
- Creating Batch History
- Configuring a Default Database for All Task History

#### Creating Batch History

IWS provides powerful tools that enable the user to create and manage batch historical information. The user is able to create batches by using the following formats:
• **Proprietary**: When using the proprietary format, each batch will be stored on a different historical file. The user can save historical data in both the normal historical file and batch files at the same time (see Trend Folder for more information about these files).

• **Database**: The historical data used for the batch is saved in the same table as the normal historical data; an additional table called BatchHistory keeps registers with the information about the batches. The list below describes the fields on the BatchHistory table:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group_Number</td>
<td>Integer</td>
<td>Trend group number. This is the number of the worksheet that you are creating to specify the tags that will be stored on your batch history.</td>
</tr>
<tr>
<td>Batch_Name</td>
<td>String</td>
<td>Name of the batch</td>
</tr>
<tr>
<td>Start_Time</td>
<td>TimeStamp</td>
<td>Date and Time that the batch was started.</td>
</tr>
<tr>
<td>End_Time</td>
<td>TimeStamp</td>
<td>Date and Time that the batch was finished</td>
</tr>
<tr>
<td>Pri_Table</td>
<td>String</td>
<td>Reserved</td>
</tr>
<tr>
<td>Sec_Table</td>
<td>String</td>
<td>Reserved</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>Batch description</td>
</tr>
<tr>
<td>Deleted</td>
<td>Boolean</td>
<td>0: Batch has not been deleted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Batch has been deleted</td>
</tr>
</tbody>
</table>

**Tip**: You can customize the name of the table and the name of the columns created in the database by editing the `project_name.APP` file, as follows:

```plaintext
[Trend]
DefaultName=NewName

[TrendGroupPRI|SEC]
BatchHistory=TableName

For example:

[TREND001PRI]
BatchHistory=MyTableForPrimaryDB

[TREND001SEC]
BatchHistory=MyTableForSecondaryDB

[Trend]
Group_Number=Trend_Worksheet
Batch_Name=Load_Number
```
Batch History Configuration

When you add a Trend worksheet (see Trend folder) and click the Advanced button, the following window will display:

![Trend Advanced Settings](image)

In the **Batch** pane, you can configure the saving of the batch history:

- **Start/Stop (input):** Enter the tag that will start/stop your batches. When the tag in this field is set to TRUE (different from 0), IWS will either start saving data to your batch file (if you are using proprietary format), or add a new register to the BatchHistory table on your database, indicating that a batch has been started. Note that historical data will be saved according to the configuration in the fields Save Trigger and Save On Tag Change options on the Trend Worksheet.

- **Name (input):** This field represents the batch name; its meaning depends on the format selected on the Trend Worksheet:
  - If you selected **Proprietary** in the **Type** field, the **Name** should comply with the format [Path]<FileName>, where:
    - **Path:** An optional field. If the path is not specified, the batch history file will be stored in the same path as the **project_name**.app file.
    - **FileName:** Name of the batch history file.
  - If you selected **Database** in the **Type** field, the value in this field will be stored in the **Batch_Name** field of the BatchHistory table.

  **Tip:** You can enter tag names between curly brackets in this field (e.g., C:\MyBatches \{MyTagWithName\}{MyTagWithNumber}.hst).

- **Delete (input):** When the tag specified in this field changes its value, the batch will be deleted. With the **Proprietary** format, the batch history file will be removed. With the **Database** format, it will set the **Delete** field in the BatchHistory table to true, but the saved historical data will remain. The Trend object only sees batches that have the delete field set to 0 (zero).

- **Existant (output):** The tag entered on this field will receive the value 1 if the batch specified in the **Name** field already exists; otherwise the tag will receive the value 0.

- **Description (output):** This field is available only when using the **Database** format. When the tag in the **Start/Stop** field changes to TRUE, the register that is added to the BatchHistory table will display the string in this field.

  **Tip:** You can enter tag names between curly brackets in this field (e.g., {MyTag}).

- **Save data even if batch is not running:** If this field is unchecked, the historical data will be saved only when the tag in the **Start/Stop** field is TRUE.

  **Tip:** The Batch Historical data can be displayed to the user in either Graphical or Table format. See Trend Folder or Grid Object to display information in these formats.
In the **Disk Space Control** area, you can control disk usage:

- **History Life Time (days)** field: Specify how many days to keep the history file on the disk. After the specified period, IWS automatically erases the file. Use this option only for files based on a date.

- **Compress After (days)** field: Specify how many days to keep the trend history file (*hst*) on the disk before compressing the file. After the specified period, IWS automatically compresses the file. Use this option only for files based on a date. This option is not available for Windows Embedded target systems.

In the **Bad Quality** area, you can determine what value will actually be saved in the batch history when the tag quality is BAD:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag Value</td>
<td>The actual value of the project tag when the tag quality was BAD, plus the specified Offset (if any).</td>
</tr>
<tr>
<td>Min Value</td>
<td>The minimum historical value of the project tag, minus the specified Offset (if any).</td>
</tr>
<tr>
<td>Max Value</td>
<td>The maximum historical value of the project tag, plus the specified Offset (if any).</td>
</tr>
<tr>
<td>Value</td>
<td>The specified Value only.</td>
</tr>
</tbody>
</table>
| NaN        | Not a number.                  Please note that when History Format is Database and Bad Quality is NaN, all of the database fields will be saved as Float type.  
  Also, if a Trend Control screen object is configured to use the history generated by this Trend worksheet, then NaN entries are counted as 0 for the purpose of calculating a trend's statistical average and deviation. |

**Note:** The **Bad Quality** feature cannot be used in projects running on Windows Embedded target systems.

Finally, in the **Disable All Data Saving** box, type the name of a project tag. When the value of the tag is TRUE (non-zero) during runtime, all data saving is disabled for this worksheet. Other Trend worksheets are not affected.
Trend Control object

The Trend Control object displays data points (values) from different data sources in a graphic format. The main features provided by the Trend Control object are:

- Display of multiple pens simultaneously
- Support for different Data Sources, such as Tag, Batch, Database and Text File
- Capability to generate X/Y graphs from the configured data sources (please refer to Appendix A for an example of an X/Y chart).
- Simultaneous display of an unlimited number of data points. This feature might be limited by the hardware used since available memory and performance will vary.
- Built-in toolbar, which provides interfaces for the user to interact with the Trend Control object during runtime
- Built-in legend, which displays the main information associated to each pen linked to the object
- Zooming and auto-scaling tools
- Horizontal and vertical orientation

**Note:** This object cannot be moved behind active objects or shapes on a project screen. In other words, it must always be in front. For more information, see #unique_370.

About the trend control runtime interface

During project runtime, a trend control has its own built-in interface that the operator can use to change how trends are displayed. This section describes the major parts of the interface and how they are used.

![Trend control runtime interface](image)

**Toolbar**

<table>
<thead>
<tr>
<th>Command/Tool</th>
<th>Icon</th>
<th>Description</th>
<th>Activation Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td><img src="image" alt="Run Icon" /></td>
<td>Sets the trend control to Run Mode (a.k.a. Online Mode). In this mode, the X-axis continues to scroll with the passage of time and the trends are updated with current tag values.</td>
<td>0 = stop trend&lt;br&gt;1 = run trend&lt;br&gt;Changing the value of this tag also changes the value of the activation tag for Stop.</td>
</tr>
<tr>
<td>Stop</td>
<td><img src="image" alt="Stop Icon" /></td>
<td>Sets the trend control to Stop Mode (a.k.a. Historical Mode). In this mode, the X-axis is stopped and the trends display only historical data. If decimation is enabled for one or more trends, the calculation and redrawing is done only in this mode.</td>
<td>0 = run trend&lt;br&gt;1 = stop trend</td>
</tr>
</tbody>
</table>
## Alarms, Events, and Trends

<table>
<thead>
<tr>
<th>Command/Tool</th>
<th>Icon</th>
<th>Description</th>
<th>Activation Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td><img src="image" alt="Period Icon" /></td>
<td>Opens a dialog which can be used to modify the X-axis scale main settings.</td>
<td>Changing the value of this tag also changes the value of the activation tag for Run.</td>
</tr>
</tbody>
</table>
| Window Zoom     | ![Window Zoom Icon](image) | Allows the user to click on the trend graph and drag the cursor to select the area that must be visible when the cursor is released. This option is disabled when the Multiple Section option (for the Y scale) is active. | 0 = disable zoom  
1 = enable zoom  
Restets to 0 after user input. |
| Horizontal Zoom | ![Horizontal Zoom Icon](image) | Allows the user to click on two points in the trend graph, defining the horizontal scale that must be available. |  |
| Vertical Zoom   | ![Vertical Zoom Icon](image) | Allows the user to click on two points in the trend graph, defining the vertical scale that must be available. This option is disabled when the Multiple Section option (for the Y scale) is active. |  |
| Zoom In         | ![Zoom In Icon](image) | Zooms in (i.e., halves the current X and Y scales) each time the user clicks the tool. | 1 = execute command  
Restets to 0 after execution. |
| Zoom Out        | ![Zoom Out Icon](image) | Zooms in (i.e., doubles the current X and Y scales) each time the user clicks the tool. |  |
| Cancel Zoom     | ![Cancel Zoom Icon](image) | Cansels the current zoom and returns the trend graph to its original scale. |  |
| Legend Properties | ![Legend Properties Icon](image) | Opens a dialog which can be used to modify the Legend main settings. |  |
| Pen Style       | ![Pen Style Icon](image) | Opens a dialog which can be used to modify the pen style of the selected trend. |  |
| Add Pen         | ![Add Pen Icon](image) | Opens a dialog which can be used to add a new trend to the trend control. |  |
| Remove Pen      | ![Remove Pen Icon](image) | Removes the selected trend from the trend control. |  |
| Multiple Sections | ![Multiple Sections Icon](image) | Switches the Y scale between Multiple Sections (a section for each trend) and Single Section (all trend share the same Y scale section). | 0 = Single Section  
1 = Multiple Sections |
| Cursor          | ![Cursor Icon](image) | Switches the cursor (ruler) between visible and hidden. | 0 = Cursor hidden  
1 = Cursor visible |
| Auto Scale      | ![Auto Scale Icon](image) | Changes the Y axis scale to fit all values from the trends that are currently being monitored. | 1 = execute command  
Restets to 0 after execution. |
| Print           | ![Print Icon](image) | Prints the current state of the trend control. (Historical data are not printed.) |  |
| SPC             | ![SPC Icon](image) | Opens a dialog which can be used to show the statistical process control (SPC) information for the selected trend. | The tag's Bit properties (B0–B4) can be used to open the dialog and pre-select options:  
• **tagname**→B0  
  • 1 = open dialog  
  Restets to 0 after open.  
• **tagname**→B1  
  • 0 = Draw Mode: Line selected  
  • 1 = Draw Mode: Shade selected  
• **tagname**→B2  
  • 0 = Show average cleared  
  • 1 = Show average selected  
  |
### Command/Tool

<table>
<thead>
<tr>
<th>Command/Tool</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td><img src="image" alt="Icon" /></td>
<td>Draws the average value and standard deviation as dashed lines, and draws the min/max values as solid lines. <strong>Note:</strong> When a value is not a number (NaN) — for example, when a tag is flagged as BAD quality — it is counted as 0 for the purpose of calculating the average.</td>
</tr>
<tr>
<td>Show average</td>
<td><img src="image" alt="Icon" /></td>
<td>Show the calculated average of all of the trend's historical values.</td>
</tr>
<tr>
<td>Show min/max</td>
<td><img src="image" alt="Icon" /></td>
<td>Show the minimum and maximum historical values of the trend.</td>
</tr>
<tr>
<td>Show standard deviation</td>
<td><img src="image" alt="Icon" /></td>
<td>Show the standard deviation of the trend. A low standard deviation indicates that the actual value tends to stay close to the average; a high standard deviation indicates that the actual value tends to vary greatly from the average.</td>
</tr>
</tbody>
</table>

### Activation Tag

<table>
<thead>
<tr>
<th>Activation Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagname-&gt;B3</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>0 = Show standard deviation cleared</td>
<td></td>
</tr>
<tr>
<td>1 = Show standard deviation selected</td>
<td></td>
</tr>
<tr>
<td>tagname-&gt;B4</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td>0 = Show min/max cleared</td>
<td></td>
</tr>
<tr>
<td>1 = Show min/max selected</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Activation tags are configured in the trend control’s object properties.

For more information, see Toolbar dialog on page 330.

### Time bar

The time bar displays the start date/time and end date/time of the graph, or in other words, the period of the X axis. For more information, see Axes dialog on page 327.

If you have configured the trend control to display historical data from a batch file, you can change these dates and times during project run time and therefore increase or decrease the period of the graph. The longer the period, the more data the graph will display but at lower resolution. The shorter the period, the less data the graph will display but at higher resolution. This is similar to using the Horizontal Zoom tool.

**Note:** If you enter a date (start or end) before 01/01/2000, it will be automatically adjusted to 01/01/2000. And if you enter a date (start or end) after the current date, it will be automatically adjusted to the current date.

### Legend

<table>
<thead>
<tr>
<th>Command</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td><img src="image" alt="Icon" /></td>
<td>Launches a dialog, where the user can replace the data point associated with the selected trend on the legend</td>
</tr>
<tr>
<td>Remove</td>
<td><img src="image" alt="Icon" /></td>
<td>Removes the selected trend from the trend control</td>
</tr>
<tr>
<td>Hide</td>
<td><img src="image" alt="Icon" /></td>
<td>When checked, the selected trend is visible; otherwise, it is hidden.</td>
</tr>
<tr>
<td>Pen Style</td>
<td><img src="image" alt="Icon" /></td>
<td>Launches an embedded dialog, where the user can modify the pen style of the selected trend.</td>
</tr>
<tr>
<td>Scale</td>
<td><img src="image" alt="Icon" /></td>
<td>When this box is checked, the Y axis scale is visible; otherwise, it is hidden. The scale can be hidden only when the Multiple Sections option is off.</td>
</tr>
</tbody>
</table>

For more information, see Legend dialog on page 332.

### Object Properties: Trend Control dialog

The **Object Properties: Trend Control** dialog is used to configure the basic properties of a Trend Control screen object.

**Accessing the dialog**

To access the **Object Properties** dialog for a specific screen object, do one of the following:
• Select the screen object, and then on the Graphics tab, in the Editing group, click Properties;
• Select the screen object, and then press Alt+Enter;
• Right-click the screen object, and then click Properties on the shortcut menu; or
• Double-click the screen object.

The dialog in detail

### Object Properties: Trend Control dialog

In addition to the elements that are common to all Object Properties dialogs, the Object Properties: Trend Control dialog contains the following elements:

<table>
<thead>
<tr>
<th>Area / Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Border</strong></td>
<td><strong>Type</strong> Sets the type of border around the graph area of the trend control. (There are no borders around the trend control’s legend or toolbar.)</td>
</tr>
<tr>
<td></td>
<td><strong>Color</strong> Sets the color of the border, if the border type is Solid. For more information, see Selecting colors and fill effects.</td>
</tr>
<tr>
<td><strong>Background</strong></td>
<td><strong>No Fill / Fill</strong> Enables the background fill for the graph area of the trend control. (There are no backgrounds for the trend control’s legend or toolbar.) If the fill is not enabled, then the graph is transparent to whatever other screen objects are behind the trend control.</td>
</tr>
<tr>
<td></td>
<td><strong>Color</strong> Sets the color and fill effect of the background fill, if it is enabled. For more information, see Selecting colors and fill effects.</td>
</tr>
<tr>
<td><strong>Points</strong></td>
<td>Opens the, which allows configuration of the trend control’s data points (or pens). For more information, see Trend Control: Points dialog.</td>
</tr>
<tr>
<td><strong>Axes</strong></td>
<td>Allows configuration of the trend control’s X and Y axes, as well as its horizontal or vertical orientation. For more information, see Trend Control: Axes dialog.</td>
</tr>
<tr>
<td><strong>Toolbar</strong></td>
<td>Allows configuration of the user toolbar that is displayed above the trend control. For more information, see Trend Control: Toolbar dialog.</td>
</tr>
<tr>
<td><strong>Data Sources</strong></td>
<td>Allows configuration of multiple data sources for the trend. For more information, see Trend Control: Data Sources dialog.</td>
</tr>
<tr>
<td><strong>Legend</strong></td>
<td>Allows configuration of the legend that is displayed below the trend control. For more information, see Trend Control: Legend dialog.</td>
</tr>
<tr>
<td><strong>Advanced</strong></td>
<td>Allows configuration of the trend control’s advanced properties, such as runtime options and tag triggers. For more information, see Trend Control: Advanced dialog.</td>
</tr>
</tbody>
</table>

Although the Trend Control object supports flexible configurations to meet the specific needs of your project, most of the settings are set by defaults based on the most common interfaces. Therefore, in many cases, you will only configure data points (displayed during runtime), which can be done easily by clicking the Points button from the Object Properties window.
**POINTS DIALOG**

The Points dialog is used to configure the data points for a Trend Control screen object. The value of each data point is represented as a pen in the trend display. You can dynamically change which data points are visible during runtime, regardless of how many data points are associated with the screen object.

**Accessing the dialog**

To access the Points dialog for a specific Trend Control screen object, first access the Object Properties dialog for that screen object and then click Points.

**The dialog in detail**

![Points dialog](image)

The following table summarizes the properties of each data point:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>A unique ID number for the point, which is assigned automatically when the point is created in this interface.</td>
</tr>
<tr>
<td>Label</td>
<td>The label associated with the Point can be displayed on the Legend, during runtime, providing a short reference to the user for each Point.</td>
</tr>
<tr>
<td>Color</td>
<td>The color of the pen used to draw the values of the Point on the Trend Control object.</td>
</tr>
<tr>
<td>Data Source</td>
<td>The data source for this point. Tag is available by default, but all other sources must be configured in the Data Sources dialog.</td>
</tr>
</tbody>
</table>
| Tag/Field   | The meaning of this parameter depends on the Data Source type associated with the data point:  
- **Tag**: Type the name of the tag with values to display. If the tag is configured in the Trend task, the history data is automatically retrieved; otherwise, only the online values are displayed.  
- **Batch**: Type the name of the tag with values to be retrieved from the Batch History file generated by the Trend Task, and displayed on the object.  
- **Database**: Type the name of the field (column) in the SQL Relational Database that holds the data Point values.  
- **Text File**: Type the number of the column that holds the Point values. The number 0 refers to the first column, 1 refers to the second column, and so on. |
| Min Scale / Max Scale | The scale of the Y-axis for this point. This overrides the default scale that is set in the Axes dialog. |

**Note**: The Min Scale and Max Scale properties can hold real numeric values up to six decimal places. If you need more precision than that, then you must configure the Min Scale and Max Scale properties with Real tags and then store the values in those tags.

- **Style** | The line and marker styles for this point; click the button to open the Pen Style dialog. |
- **Options** | Additional options for this point; click the button to open the Options dialog. |
- **SPC** | Calculated statistics to be used in statistical process control (SPC); click the button to open the SPC dialog. |
- **Hide** | Tag trigger — when the value is TRUE, the data point is hidden in the trend display. |

**Pen Style dialog box**

Use the Pen Style dialog box to customize the style of the pen that draws the point’s values during run time.

**Accessing the dialog box**

To access the Pen Style dialog box for a specific point, do the following:
1. In the screen editor, select the Trend Control object, and then open its object properties.
2. In the object properties, click **Points**.
3. In the **Points** dialog box, select a point, and then click the **Style** column for that point.

This dialog box can also be opened on the project client during run time, by clicking the Pen Style tool in either the toolbar or the legend of the trend control. For more information, see About the trend control runtime interface on page 316.

**The dialog box in detail**

![Pen Style dialog box](image)

The **Pen Style** dialog box includes the following elements:

**Elements in the Pen Style dialog box**

<table>
<thead>
<tr>
<th>Area / Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Line Settings</strong></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>You have the option of defining a Hi Limit and a Lo Limit for each data Point, with the Options dialog. The Pen Style Dialog allows you to configure different settings for the pen (e.g., color), both when its values are within the limits (Normal State) and not within the limits (Out of Limits state).</td>
</tr>
<tr>
<td>Use Normal Settings</td>
<td>Available only for the <strong>Out of Limits</strong> state. When checked, the pen will always be displayed with the settings for the Normal state, even if the data point values are not within the limits configured for it.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of line (e.g., solid, dashed, dotted) that connects the data points.</td>
</tr>
<tr>
<td>Weight</td>
<td>The weight of the line that connects the data points.</td>
</tr>
</tbody>
</table>
| Expansion           | The algorithm used to connect the points, as follows:  
  * ▲ ▼ : Consecutive points are directly connected to each other by an analog line. This option is suitable for numerical values.  
  * △ △ : Consecutive points are connected only through horizontal or vertical steps (depending on the orientation of the trend display). This option is suitable for Boolean values. |
| Marker              | The shape used to mark each data point. If no shape is selected, then only the connecting line between points is displayed. |
| Marker size         | The size of the data point marker. |
| Color               | The color of the trend line and data point markers. |
| **Fill**            |             |
| Fill Type           | The type of fill between the trend line and the number line. |
| Pattern File        | The graphic file used to fill the trend area. Available only Fill Type is set to Custom Pattern.  
  Click the browse button to open a Windows file browser and then select the desired graphics file. The file should be located in your project folder.  
  See below for an example of trends with custom fill patterns. |
### Area / Element Name
<table>
<thead>
<tr>
<th>Color</th>
<th>The color used to fill the trend area. Available only when Fill Type is set to <strong>Solid Color</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill Transparency (%)</td>
<td>The transparency level of the fill. (If the fill is transparent, then other trends behind it can be seen through it, making the entire graph easier to read.) Available for both <strong>Custom Pattern</strong> and <strong>Solid Color</strong>.</td>
</tr>
</tbody>
</table>

![Trends with custom fill patterns](image)

**Note:** When your project is viewed on either a Windows Embedded device or a Thin Client (any OS), the *Pen Style* dialog box — which can be opened during run time; see "Accessing the dialog box" above — allows the user to change the pen color only.

**Tip:** To programmatically modify these pen style settings during run time, configure *Style Modifier* in the *Options* dialog box. For more information, see Modify the pen style of a point during run time on page 324.

### Options
Use the *Options* dialog box to configure additional options for a specific point in a Trend Control object.

### Accessing the dialog box
To access the *Options* dialog box for a specific point: first access the *Points* dialog box, where all of the points in a Trend Control object are configured, and then click the *Options* column for that point. For more information, see Trend Control object on page 316.
The dialog box in detail

The *Options* dialog box includes the following settings:

**Description**

This text can be displayed in the legend of the Trend Control object during run time, providing a brief description of the trend.

If you specify a tag name in curly brackets (e.g., `{MyTag}`), that tag's Description property is used.

**Eng. Unit**

This text can be displayed in the legend of the Trend Control object during run time, providing the engineering unit (i.e., the unit of measurement) associated with the trend.

If you specify a tag name in curly brackets (e.g., `{MyTag}`), that tag's Engineering Unit property is used.

**Lo Limit**

Type a tag name or numerical value. When the trend falls below this value during run time, it can be drawn in a different style (e.g., color). For more information, see *Pen Style dialog box* on page 320.

If you specify a tag name in curly brackets (e.g., `{MyTag}`), that tag's LoLimit property is used.

**Hi Limit**

Type a tag name or numerical value. When the trend rises above this value during run time, it can be drawn in a different style (e.g., color). For more information, see *Pen Style dialog box* on page 320.

If you specify a tag name in curly brackets (e.g., `{MyTag}`), that tag's HiLimit property is used.

**Hide Scale**

You can configure a tag in this field to control the visibility of the scale (Y axis) associated with this pen during runtime by changing the value of this tag (0=Show ; 1=Hide).

**Break Interval**

Type a numerical value (default is 7200). This is the maximum interval allowed between two consecutive points in a trend. If the interval between the two points is greater than this value, the Trend Control object assumes that no data were collected during the interval and it does not draw a line connecting the two points.

If the X axis is configured to be numeric, the value specified here is taken as a numeric scalar value. If the X axis is configured to be date/time, the value specified here is taken as seconds.
This setting accepts some special values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Do not connect any points.</td>
</tr>
<tr>
<td>-2</td>
<td>Connect only points that have ascending values.</td>
</tr>
</tbody>
</table>

**X Axis Offset**

Type a tag name or numerical value. The value is the offset from the X-axis scale configured for the Trend Control object. This setting is useful when you want to display data from two or more trends using a different scale for each trend, so that you can compare them.

If the X axis is configured to be numeric, the value specified here is taken as a numeric scalar value. If the X axis is configured to be date/time, the value specified here is taken as seconds.

**Cursor Value**

Type the name of a project tag. During run time, the tag is continuously updated with the value of the trend where it is intersected by the vertical cursor (if any) in the Trend Control object.

**Y-Axis Log Base**

Type a tag name or numerical value. If the value is 0 (or the box is left empty), the Y axis of the trend is a normal linear scale. If the value is anything other than 0, the Y axis is a logarithmic scale with a log base equal to that value. The most common log base is 1, which gives a scale of 1, 10, 100, 1000, and so on, but you can specify any log base.

**Annotation ID**

**Style Modifier**

Modify the pen style's expansion, line color, line weight, or line type. For more information, see **Modify the pen style of a point during run time** on page 324.

**Draw Mode**

Type a tag name or numerical value. If the value is 1, the historical data for this trend are decimated before the trend is drawn in the Trend Control object. That means the trend's X axis is divided into a number of intervals (as determined by **Max Points** in the Advanced settings), and then all of the data points within each interval are averaged together to be drawn as a single point.

This is similar to the **Decimation** option in the Advanced settings, except that the decimation is done only for this trend rather than for all trends in the Trend Control object.

**Note:** If decimation is enabled and the X axis is configured to be numeric rather than date/time, the data used in the X axis must be properly sorted. For more information, see the **Data Sources** settings.

**Note:** If the data source for the point is a tag from a Trend worksheet that has been configured to save to a Wonderware Historian database, decimation must be enabled.

*Modify the pen style of a point during run time*

Use **Style Modifier** to modify the pen style of a point in a trend control.

By default, the data points in a trend control are drawn with a solid, black line. You can change the style of the line — more specifically, you can change the style of the pen that draws the line — by changing the settings in the **Pen Style** dialog box. The user can also open the **Pen Style** dialog box during run time and change the settings then. For more information, see **Pen Style dialog box** on page 320.
Alternatively, you can use **Style Modifier** (in the **Options** dialog box) to programmatically modify some of the pen style settings during run time. In other words, you can specify project tags that will determine the pen style settings, and then you can use scripts or user input to change the tag values during run time.

![Options dialog box](image)

The **Style Modifier** box accepts a text string that includes one or more parameters, and each parameter modifies one element of the pen style. The text string must have this basic format:

```
<Parameter1>=<Value1>;<Parameter2>=<Value2>;…;<ParameterN>=<ValueN>
```

The following table lists the supported parameters and their accepted values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Accepted Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expansion</td>
<td>The method or algorithm used to connect the data points.</td>
<td>• 0 (smooth/analog)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 (squared/digital)</td>
</tr>
<tr>
<td>Type</td>
<td>The type of line (e.g., solid, dashed, dotted) that connects the data points.</td>
<td>• 0 (solid)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 (dashed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2 (dotted)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3 (dash-dot)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4 (dash-dot-dot)</td>
</tr>
<tr>
<td>Weight</td>
<td>The weight (i.e., thickness) of the line, in pixels.</td>
<td>from 0 to 10</td>
</tr>
<tr>
<td>Color</td>
<td>The color of the line.</td>
<td>a 24-bit color value from 0 to 16777215</td>
</tr>
<tr>
<td></td>
<td>For more information about colors in IWS, see <strong>Color Interface</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** These parameters are the same as the settings in the **Pen Style** dialog box.

You could specify literal values for any or all of the parameters, but that is effectively the same as using the **Pen Style** dialog box to configure those settings. The key to programmatically modifying the pen style is specifying tag names or expressions enclosed in curly brackets ({}), for the parameter values. Then, whenever the value of a specified tag/expression changes, the pen style is modified.
For example, in the **Style Modifier** box, type the following string:

```
Expansion={(ExpansionTag);Color={RGBColor(0,0,ColorTag)}
```

For the parameter **Expansion**, you specified the project tag `ExpansionTag` enclosed in curly brackets. There are only two accepted values (0 and 1) for **Expansion**, so `ExpansionTag` could be Boolean type. Then, whenever the value of **ExpansionTag** changes during runtime, the expansion method is modified accordingly.

For the parameter **Color**, you specified an expression that calls the function `RGBColor` to convert RGB color values to a 24-bit color value. The red and green color values remain constant at 0, but the blue color value is determined by the project tag `ColorTag`. Then, whenever the value of **ColorTag** changes during run time, the value returned by the function also changes and the line color is modified accordingly.

Following this example, you can specify any project tags or expressions for the parameters as long as their returned values are within the accepted values for the parameters.

**SPC**

Use the **SPC** dialog box to specify project tags that will receive certain statistical values that are calculated from the entire history of a trend. These statistics are used in statistical process control (SPC), which is a method for monitoring processes and ensuring that they operate efficiently.

**Accessing the dialog box**

To access the **SPC** dialog box for a specific point: first access the **Points** dialog box, where all of the points in a Trend Control object are configured, and then click the **SPC** column for that point. For more information, see [Trend Control object on page 316](#).

**Note:** If the data source for the point is a tag in a Trend worksheet that has been configured to save to a Wonderware Historian database, **SPC** is not supported.

**The dialog box in detail**

The **SPC** dialog box includes the following settings:

**Average**
Type the name of a project tag (Real type) that will receive the calculated average of all of the data point’s historical values.

**Minimum Value**
Type the name of a project tag (Real type) that will receive the minimum historical value of the data point.

**Maximum Value**
Type the name of a project tag (Real type) that will receive the maximum historical value of the data point.

**Standard Deviation**
Type the name of a project tag (Real type) that will receive the standard deviation of the data point. A low standard deviation indicates that the value of the data point tends to stay close to the average; a high standard deviation indicates that the value tends to vary greatly from the average.

**Count**

Type the name of a project tag (Integer or Real type) that will receive the total number of historical values, or samples, for the data point. The count will increase as the project runs and the historical database grows.

**AXES DIALOG**

**Accessing the dialog**

To access the Axes dialog for a specific Trend Control screen object, first access the Object Properties dialog for that screen object and then click Axes.

**The dialog in detail**

The Axes dialog contains the following elements:

<table>
<thead>
<tr>
<th>Group / Setting</th>
<th>Data Type</th>
<th>Date/Time</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X Axis</strong></td>
<td>XAxis</td>
<td>Date/Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Type</td>
<td>Numeric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scale Format</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Period (when Data Type is Date/Time)</strong></td>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Auto</td>
<td>• <em>Auto</em>: When this option is selected, the Trend Control object works with Start Date/Time when it is triggered to Pause Mode, and it works with Time Before Now when it is triggered to Play Mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start Date/Time</td>
<td>When this option is selected, the value of the tag configured in the Time field defines the starting Date/Time for the data displayed on the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time Before Now</td>
<td>When this option is selected, the value of the tag configured in the Time field defines the amount of time before the current Date/Time, which will be used as the starting Date/Time for the data displayed on the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Duration</td>
<td>Defines the Period of data displayed on the object. You can configure a string tag in this field, so you can change the duration dynamically during runtime by changing the value of this tag. The format of the value supported by this property is HH:MM:SS. For example, 36:00:00 (thirty six hours).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>This field is optional. The value of the tag configured in this field represents a period of time, rather than a specific date or time. The meaning of this value depends on the option set for the Type property.</td>
</tr>
<tr>
<td>Group / Setting</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When the Type is set as Start Date/Time, the value of the tag configured in this field must comply with the format Date Time. For example, 02/10/2005 18:30:00.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When the Type is set as Time Before Now, the value of the tag configured in this field must comply with one of the following formats:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Time (string value). For example, 48:00:00 (forty eight hours).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Number of hours (real value). For example, 2.5 (two hours and thirty minutes).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the Time field is left blank (or if the tag configured in this field has the value 0), the object displays data up to the current Date/Time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period (when Data Type is Numeric)</td>
<td>Min / Max</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Minimum and maximum values displayed on the X axis.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Min and Max properties can hold real numeric values up to six decimal places. If you need more precision than that, then you must configure the Min and Max properties with Real tags and then store the values in those tags.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eng. Units</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Engineering Unit (e.g., Kg, BTU, psi) that is associated with the X axis during runtime.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid</td>
<td>Divisions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can configure the number of divisions (vertical or horizontal lines) drawn on the object for the X and/or Y axis respectively, as well as the color of these lines.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid</td>
<td>Color</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The color of the vertical grid lines.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time Bar</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When checked, the Time bar is displayed below the X axis during runtime; otherwise, it is hidden. The time bar is a standard interface that can be used by the operator to change the X axis scale during runtime.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scroll Bar</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When checked, the Scroll bar is displayed below the X axis during runtime; otherwise, it is hidden. The time bar is a standard interface that can be used by the operator to navigate through the X axis scale during runtime. Optional, you can configure a tag in the Scroll bar field, which defines the period for the scroll bar. If this field is left empty, the period is equal to the current value for Duration of the X axis.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cursor</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The cursor is an optional ruler orthogonal to the X axis, which can be used during runtime to obtain the value of any pen at a specific point (intersection of the pen with the cursor). When you click this button, the Cursor dialog launches, where you can configure the settings for the optional vertical cursor as follows:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Positon</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defines the position of the X axis, as well as its direction and orientation, as follows:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y Axis</td>
<td>Grid</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Divisions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can configure the number of divisions (vertical or horizontal lines) drawn on the object for the X and/or Y axis respectively, as well as the color of these lines.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Color</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The color of the horizontal grid lines.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scale</td>
<td>Min / Max</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default minimum and maximum values displayed in the Y axis. Used when more than one pen shares the same scale (Multiple Sections disabled), and/or for the points whose Min and Max fields are not configured (left blank).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Please note that if you configure a trend point to have a logarithmic scale (see Points - Options), then the value configured here for Min should be greater than 0. Even if you configure a minimum value less than or equal to 0 (which is impossible for a logarithm), a minimum value of 0.00000000000000000000001 will be used automatically during run time. This will not change any of the object properties that you have configured.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Multiple Selections</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When checked, the Y scale is divided automatically into one section for each pen; otherwise, all pens share the same Y scale.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Launches a dialog for configuring the format of the labels displayed by the Y axis.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The tags configured in the Period/Range fields are automatically updated when the user changes the X scale dynamically during runtime, using the Time bar embedded in the object.
• **Data Type:** The X axis can display either Date/Time values or numeric values, according to this setting.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Scale Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date/Time</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>Numeric</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

**Note:** The number of decimal points for the X or Y scale (Decimals) can be configured with a tag. Therefore, this setting can be modified dynamically during runtime.

• **Cursor:** The cursor is an optional ruler orthogonal to the X axis, which can be used during runtime to obtain the value of any pen at a specific point (intersection of the pen with the cursor). When you click this button, the Cursor dialog launches, where you can configure the settings for the optional vertical cursor as follows:

![Cursor Dialog](image3.png)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>When checked, the vertical cursor is visible during runtime.</td>
</tr>
<tr>
<td>Color</td>
<td>Color of the line drawn for the cursor.</td>
</tr>
<tr>
<td>Position (0%–100)</td>
<td>You can configure a numeric tag in this field, which is proportional to the position of the cursor on the X axis, from 0 to 100%. When this value is changed, the position of the cursor is automatically modified.</td>
</tr>
<tr>
<td>Value Output</td>
<td>You can configure a string tag in this field that returns the value of the X axis in which the cursor is currently positioned.</td>
</tr>
</tbody>
</table>

• **Position:** Defines the position of the X axis, as well as its direction and orientation, as follows:

![Position Dialog](image4.png)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placement</td>
<td>Side of the trend control on which the X axis will be placed.</td>
</tr>
<tr>
<td>Direction</td>
<td>Direction of the X axis.</td>
</tr>
</tbody>
</table>
### TOOLBAR DIALOG

The *Toolbar* dialog is used to customize the toolbar on the Trend Control screen object.

#### Accessing the dialog

To access the *Toolbar* dialog for a specific Trend Control screen object, first access the *Object Properties* dialog for that screen object and then click *Toolbar*.

#### The dialog in detail

![Toolbar dialog](image)

The **Show toolbar** option controls whether the entire toolbar is shown during runtime. You may hide the toolbar to save space or to prevent users from changing the trend display.

Also, each command/tool in the toolbar has the following properties:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>The name of the command/tool. For more information about each tool, see</td>
</tr>
<tr>
<td>Show</td>
<td>The option to show the tool on the toolbar.</td>
</tr>
<tr>
<td>Activation Tag</td>
<td>An optional <em>tag trigger</em> — when the value of the tag changes from FALSE (0) to TRUE (any non-zero value), the command is activated as if the operator clicked the tool. This can be used to script changes in the trend display during runtime.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>The tooltip that is displayed when the mouse cursor hovers over the tool.</td>
</tr>
</tbody>
</table>

For more information, see *About the trend control runtime interface* on page 316.

### DATA SOURCES DIALOG

#### Accessing the dialog

To access the *Data Sources* dialog for a specific Trend Control screen object, first access the *Object Properties* dialog for that screen object and then click *Data Sources*.
The data source defines the location of the values from the data point(s) associated with it. Many points can share the same data source — you do not need to create one data source for each data point.

The data source tag is available by default to the Trend Control object. You can add additional data sources with the New button. The name you enter will be used as an alias to link the data points to this new data source.

The other fields in this dialog allow you to edit the data source settings:

- **Source Type**: Select the source type of the location of the data point values.
- **X Axis field**: If the X axis of the trend graph is set to be numeric instead of date/time (in the Axes settings), then enter the name of the field (column) of the data source that holds the data for the X axis.

  **Note**: If you have enabled decimation (either in the Advanced settings for all trends or in the Options settings for a single trend), then the field that you have specified here must be sorted in ascending order. The procedure to do this varies by source type (e.g., text editor, spreadsheet application, external database, etc.), so for more information, see the documentation for your specific type.

- **Max. Buffer**: The maximum amount of data (in bytes) that will be held in runtime memory.
- **Load Progress**: The tag in this field will receive a real value (0-100) that represents the percentage of the Data Source load progress.
- **Sort**: This option is useful for plotting data from a Text file. When enabled (checked), it sorts the data and shows the Cursor column value until the Max. Buffer is filled. When disabled (unchecked), the data are not sorted and the Cursor column value is not shown.
- **Keep Open**: This option keeps the data source open as long as the screen that contains the Trend Control object is open. This improves the performance of the runtime project, but keeping the data source open may cause other problems like database connection errors (when Source Type is Database) and file write conflicts (when Source Type is Batch or Text File). To close the data source after the data has been loaded, clear (uncheck) this option.

- **Data Source Settings**: Click to define the settings for the selected Source Type.

The following table summarizes the settings for each Source Type:

<table>
<thead>
<tr>
<th>Data Source Type</th>
<th>Description</th>
<th>X-Axis field</th>
<th>Data Source Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>Batch generated by the Trend task of IWS</td>
<td>Disabled. The X-Axis data will be retrieved automatically on the correct position from the proprietary Batch file generated by IWS.</td>
<td>Enter the data point values in Batch Name for their retrieval. You can configure a tag between curly brackets in this field to change this setting dynamically during runtime.</td>
</tr>
</tbody>
</table>
## Data Source Type | Description | X-Axis field | Data Source Settings
---|---|---|---
**Database** | SQL Relational Database | Field name that contains the X-axis data | ![Database Configuration Dialog Window](image)

Configure the settings to link this Data Source to the SQL Relational Database that holds the data point values. See Database Configuration Dialog Window for further information about this dialog interface. Please refer to Appendix B for an example of configuring databases.

**Text File** | Text file (e.g., CSV file) with data point values separated by a specific delimiter | Number of the column that holds the X-Axis data. The number 0 refers to the first column, 1 refers to the second column, and so on. | ![Text File](image)

Enter the name of the text file that holds the data points. The default path is the current project folder. You can configure a tag between curly brackets in this field to change this setting dynamically during runtime.

You can also choose one or more delimiters for the data stored in the text file. The value of each row is written in the text file between two delimiters. When using a comma as a delimiter, the grid object is able to read data from CSV files. You can even choose a custom delimiter by checking the Other option. Please refer to Appendix A for an example of configuring text files.

---

**Note**: There is a default query timeout of 120 seconds, to prevent the project client from hanging on an unusually long data source query. To adjust the timeout period, manually edit your project file (`projectname.APP`) and change the following setting:

```plaintext
[Trend]
QueryTimeout=120
```

If you change the value to 0, then there will be no timeout at all; data source queries will always continue until they are completed.

### LEGEND DIALOG

#### Accessing the dialog

To access the Legend dialog for a specific Trend Control screen object, first access the Object Properties dialog for that screen object and then click Legend.
The dialog in detail

The *Legend* dialog contains the following elements:

- **Show**: When checked, the embedded legend is displayed during runtime. This interface provides useful information associated with the pens currently linked to the object.

- **Available / Visible**: The items in the **Visible** box are displayed in the legend during runtime. You can add items to and remove them from the **Visible** box using the » and « buttons respectively. Moreover, you can use the **Move Up** and **Move Down** buttons to change the order in which the items are displayed in the legend during runtime.

The following table lists the available legend items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Legend Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eng Units</td>
<td>![Eng Units Icon]</td>
<td>The tag/pen's Engineering Units.</td>
</tr>
<tr>
<td>Min</td>
<td>![Min Icon]</td>
<td>The tag/pen's minimum possible value.</td>
</tr>
<tr>
<td>Max</td>
<td>![Max Icon]</td>
<td>The tag/pen's maximum possible value.</td>
</tr>
<tr>
<td>Selection</td>
<td>![Selection Icon]</td>
<td>Press button to select another tag for this pen.</td>
</tr>
<tr>
<td>Remove</td>
<td>![Remove Icon]</td>
<td>Press button to completely remove this pen from the legend and the Trend chart.</td>
</tr>
<tr>
<td>Hide</td>
<td>![Hide Icon]</td>
<td>Select (check) option to hide this pen in the Trend chart.</td>
</tr>
<tr>
<td>Pen Style</td>
<td>![Pen Style Icon]</td>
<td>Press button to change the pen's line style, weight, color, markers, and so on.</td>
</tr>
<tr>
<td>Scale</td>
<td>![Scale Icon]</td>
<td>Select (check) option to show the pen's scale on the Trend chart.</td>
</tr>
<tr>
<td>Description</td>
<td>![Description Icon]</td>
<td>Description of the tag/pen.</td>
</tr>
<tr>
<td>Current</td>
<td>![Current Icon]</td>
<td>The current value of the tag configured to the pen.</td>
</tr>
<tr>
<td>Cursor</td>
<td>![Cursor Icon]</td>
<td>The value of the pen where it intersects the cursor line.</td>
</tr>
</tbody>
</table>

- **Properties**: Allows you to configure the properties for the field highlighted in the **Available** or **Visible** box:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Label for the field displayed during runtime.</td>
</tr>
<tr>
<td>Width</td>
<td>Width for the field (in pixels) during runtime.</td>
</tr>
<tr>
<td>Align</td>
<td>Alignment of the data displayed in the field.</td>
</tr>
</tbody>
</table>
Alarms, Events, and Trends

### Property Description

| Available during runtime | When this option is checked, the user can show or hide the field during runtime. |

- **Maximum size**: Defines the size of the legend in terms of number of rows. For instance, the user might have 8 points being displayed in the trend object, if the maximum size is set to two, the legend will have a scrollbar to allow the user to scroll to the other points.

- **Number of items**: Number of points (default) displayed on the legend. You can allow the user to add/remove points during runtime regardless of the value in this field.

- **Selected Item**: You can configure a numeric tag in this field. The object writes in this tag the number of the selected row. In addition, you can select different rows by writing their values in this tag.

- **Fonts**: Sets the font for the text displayed in the legend.

For more information, see About the trend control runtime interface on page 316.

### ADVANCED

Use the *Advanced* dialog box to configure advanced settings for a Trend Control object.

#### Accessing the dialog box

To access the *Advanced* dialog box for a Trend Control object, first access the object properties for that object and then click **Advanced**. For more information, see Trend Control object on page 316.

#### The dialog box in detail

**Advanced dialog box**

The *Advanced* dialog box includes the following settings:

<table>
<thead>
<tr>
<th>Group</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Mode Options</td>
<td>Update trigger</td>
<td>When the tag configured in this field changes value, the trend object is updated (refreshed).</td>
</tr>
<tr>
<td></td>
<td>Update interval</td>
<td>When the update trigger is issued and the X Axis if of type numeric, the value on this field will be added to the minimum and maximum values of the X Axis.</td>
</tr>
<tr>
<td></td>
<td>Load indicator</td>
<td>Type the name of a project tag. While the trend control is loading external data, the tag receives a value of 1, and when the trend control has finished loading the data, the tag receives a value of 0.</td>
</tr>
<tr>
<td></td>
<td>Move to current time on run</td>
<td>When this box is checked, X axis shifts to the current time automatically when the object is triggered to Play mode, during runtime.</td>
</tr>
<tr>
<td>Group</td>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Retrieve bounding samples</td>
<td>Setting</td>
<td>When this box is checked, the object retrieve the data outbound the object (first points only). Uncheck this option can improve the performance, since the points outbound the object will not be retrieved from the history. On the other hand, the object will not draw lines linking the first and last samples to the extremities of the object.</td>
</tr>
<tr>
<td>Run-Time Config</td>
<td>Save trigger</td>
<td>The settings of the Trend object modified during runtime can be saved in temporary files. This option can be used to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Keep the settings consistent, so when the user closes the screen and opens it again, or re-starts your project, the settings configured during runtime are not lost.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Create standard settings for different scenarios and load the appropriate configuration during the runtime, based on a pre-defined condition or based on the user-selection.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the tag configured in this field changes value (e.g., toggles), the current settings of the Trend object are saved in the temporary file. This command is not available for the Thin Client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following settings are saved:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Points: Color, Tag/Field (in case you are using indirect tags), Min Scale, Max Scale, Hide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Axes: Period (start time, end time, etc.), Grid Divisions, Number of Labels, Enable Cursor, and the current zoom</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Toolbar: Show/hide state of all commands that do not have activation tags configured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Legend: Visible columns with widths, Maximum Size, Number of Items, Selected Item</td>
</tr>
<tr>
<td></td>
<td>Load trigger</td>
<td>When the tag configured in this field changes value (e.g., toggles), the settings from the temporary file are loaded and applied to the Trend object during runtime.</td>
</tr>
</tbody>
</table>
|                               |              | ![Note: After the screen where the Trend object is configured is saved, the settings are not automatically loaded from the temporary file when the screen is opened again, unless the Load trigger command is executed before the screen is closed.](image)
|                               | File Name    | If this field is left blank, the temporary file is saved in your project's Web sub-folder with the syntax ScreenNameObjectIDTrendControl.stmp (e.g., MyScreen10TrendControl.stmp). The Thin Client station saves/loads the temporary file in the standard Temp directory of the operating system (e.g., Documents and Settings \CurrentUser\Local Settings\Temp). You can configure a customized file name for the temporary file in this field or even configure a string tag between curly brackets, so the user can change the name of the configuration file dynamically during runtime by changing the value of this tag. If you do not specify any path, the file is saved in your project's Web sub-folder by default.
|                               | Auto Save    | When this box is checked, the current settings of the Trend are automatically saved in the temporary file when the screen where the Trend is configured is closed during runtime. If the box is not checked, the settings are saved only when the Save trigger command is executed.                                                                                                                  |
| Custom Point Selection        | Screen       | This interface allows you to create your custom dialog to modify or insert pens to the object. Name of the screen which must be launched when the user triggers a command to modify or insert a new pen to the object during runtime.                                                                                                              |
|                               | Point Number | Point number (from the Points dialog), indicating the point associated to the pen that will be inserted or modified during runtime.                                                                                                                                                                                                                     |
|                               | Add Indicator| Flag that indicates that the user triggered an action to insert a new pen (value 1) instead of modifying a pen that is already been visualized (value 0).                                                                                                                                                                                                  |
| Export to File                | Trigger      | When the tag configured in this field changes value (e.g., toggles), the current state of the trend control is exported to an image file. The toolbar and scroll bar are not included. The legend and time display are included only if the trend control is configured to show them.                                                                                       |
|                               | File Name    | The file path and name of the exported file. If no path is specified, then the file is saved in the project directory. If no extension is specified, then it is determined by Format.                                                                                                                                                                                                 |

Tip: You can specify a project tag in curly brackets (e.g., {tagname}), to programatically change the file name during runtime.
<table>
<thead>
<tr>
<th>Group</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>The tag configured in this field receives status codes that indicate the success or failure of the export.</td>
<td></td>
</tr>
<tr>
<td>Format</td>
<td>The graphic format of the exported file. If <strong>Auto</strong> is selected, then the format is determined by the extension specified in <strong>File Name</strong>. If <strong>Auto</strong> is selected but no extension is specified, then the default format is BMP.</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>The image file is exported at full size by default. However, you can specify the width and height (in pixels).</td>
<td></td>
</tr>
<tr>
<td>Decimation</td>
<td><strong>Enable</strong></td>
<td>When this option is selected, the trends in the Trend Control object that are configured to show historical data will have their data decimated before the trends are drawn. This means that for each trend, the X axis is divided into a number of intervals (determined by <strong>Max Points</strong>) and all of the data points within each interval are averaged together to be drawn as a single point. This can improve runtime performance when there is a large amount of historical data to display, and it can make the trends easier to read. Decimation only works when the trend control is in Stop Mode (a.k.a. Historical Mode). Please note that when this option selected, decimation is done for all trends that are configured to show historical data. To do it for only for a single trend, use <strong>Draw Mode</strong> in the Points – Options settings.</td>
</tr>
<tr>
<td>Max Points</td>
<td>The maximum number of data points used to draw each trend. Default is 2048.</td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td><strong>VK (Virtual Keyboard)</strong></td>
<td>Virtual Keyboard type used for this object.</td>
</tr>
<tr>
<td></td>
<td><strong>Ignore X Filter</strong></td>
<td>When this box is checked, the X Filter is ignored to avoid adding the WHERE or querying clause to the Data Sources.</td>
</tr>
<tr>
<td></td>
<td><strong>Enable translation</strong></td>
<td>Enable the external translation for the text displayed by this object.</td>
</tr>
<tr>
<td></td>
<td><strong>Auto Format</strong></td>
<td>When checked, decimal values in the Current, Cursor, Max, Min and Scale columns will be formatted according to the virtual table created by the function <strong>SetDecimalPoints</strong>.</td>
</tr>
</tbody>
</table>

**Using the Data Source Text File**

The Trend Control can generate trend charts from any Text File that has the values organized in columns and rows. The columns should be separated from each other by special characters (usually the comma). Each sample (pair of values representing a point in the graph) is represented by a row (a line in the file). Suppose that the user wants to display a chart with the information in the following table:

<table>
<thead>
<tr>
<th>X Value</th>
<th>Y1 Value</th>
<th>Y2 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>40</td>
</tr>
</tbody>
</table>
We have one variable that represents the X Axis and two variables (Y1 and Y2) that will represent different lines in the chart. The first step is to convert the data into a text file. If we adopt the comma as our separator the file will be as shown below:

![TrendData.txt](image)

We strongly recommend that you save the file in the same folder where the project is. By doing so, you do not have to specify the entire path and your project will still work, even if it is copied to a different computer.

Once you have added the Trend Control to your screen, double-click on the object to open the **Object Properties** and click on **Axis**. Change the **Data Type** of the **X Axis** to **Numeric**, and set the ranges as shown in the picture below:

![Axis Properties](image)

Click **OK** on this window and then, in the **Object Properties** window, click on the **Data Sources** button. The following window will display:

![Data Sources](image)

*Trend Control – Data Sources dialog*
We need to create a data source in order to access to the text file. Click on the **New...** button, specify the **Data Source** name **MyTextFile** and then click **Create**. You should see the following information now:

![Setting X Axis field to 0](image)

On the **X Axis field** we need to indicate which column in our text file represents the X Axis. In our example we are using column zero, so enter **0** for this field, then click **Data Source Settings...**. The following window will display:

![Selecting the text file](image)

If you have copied the text file to the project folder, you only have to specify the file name, otherwise, enter with the complete path where the file is located (use the browse button as needed). Click **OK** on this window and **OK** again to finish the data source configuration and close the **Data Source** configuration Window.

Now we need to define our Y1 and our Y2. They will be represented by points on our Trend Control. Double-click on the Trend Control again to access the **Object Properties** window and then click on **Points...**. Your next step is to define the points according to the following figure:
After following these steps, run your project and you should see something similar to the figure below:

**Using the Data Source Database**

The Trend Control can generate trend charts from any Relational Database that can be accessed through the ADO.Net technology. This Appendix illustrates how to access a Microsoft Access Database; if you are using another type of database, almost all the definitions will apply, however you will need to configure your connection on a different way. For information on how to configure other databases, please refer to the Appendixes in the Database Interface section of this manual.
Suppose that you have an access database at your C drive named `mydata.mdb` and that you want to generate a chart based on the information in the following table:

```
<table>
<thead>
<tr>
<th>Time_Stamp</th>
<th>Temperature</th>
<th>Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:30:00 AM</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>9:30:01 AM</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>9:30:02 AM</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>9:30:03 AM</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>9:30:04 AM</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>9:30:05 AM</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>9:30:06 AM</td>
<td>90</td>
<td>40</td>
</tr>
<tr>
<td>9:30:07 AM</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>9:30:08 AM</td>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td>9:30:09 AM</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>9:30:10 AM</td>
<td>70</td>
<td>40</td>
</tr>
<tr>
<td>9:30:11 AM</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>9:30:12 AM</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>9:30:13 AM</td>
<td>40</td>
<td>70</td>
</tr>
<tr>
<td>9:30:14 AM</td>
<td>30</td>
<td>80</td>
</tr>
<tr>
<td>9:30:15 AM</td>
<td>20</td>
<td>90</td>
</tr>
<tr>
<td>9:30:16 AM</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>9:30:17 AM</td>
<td>20</td>
<td>110</td>
</tr>
<tr>
<td>9:30:18 AM</td>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>9:30:19 AM</td>
<td>40</td>
<td>130</td>
</tr>
<tr>
<td>9:30:20 AM</td>
<td>50</td>
<td>140</td>
</tr>
<tr>
<td>9:30:21 AM</td>
<td>60</td>
<td>150</td>
</tr>
</tbody>
</table>
```

The first step is to add the Trend Control to your screen. Now double-click on the object to open then Object Properties and click on `Data Sources...`. The following window will display:

**Trend Control – Data Sources dialog**

We need to create a data source in order to access to the database. Click the `New...` button, specify the `Data Source` name `MyDB` and then click `Create`. You should see the following information now:

**Setting X Axis field to Time_Stamp**
Change the **Source Type** to **Database** and specify **Time Stamp** in the **X Axis field**. Then click on the **Data Source Settings...** button, the following window will display:

![Database Configuration](image)

**Clearing the Use project default option**

Uncheck the checkbox **Use project default** and click on the browse button ... in order to configure the connection string. The following window will display:

![Data Link Properties](image)

**Selecting the OLE DB Provider**
Select the Microsoft Jet 4.0 OLE DB Provider and click Next. In the following window, you should specify the database path:

![Selecting the database file](image)

Click OK to finish the Connection String configuration. Now uncheck the option Use default name and select the table from your database as shown below:

![Selecting the table in the database](image)

Click OK on this window and OK again to finish the data source configuration and close the Data Source configuration window.
Now we need to define Temperature and Pressure. They will be represented by points on our Trend Control. Double-click on the Trend Control again to access the Object Properties window and then click **Points**... Your next step is to define the points according to the following figure:

If you run the trend, it will start with the current date/time. In order to see the data in the chart you will have to properly configure the start date/time as shown below:
The Grid object allows you to read/write data in a tabular format from the data source configured in the object.

To draw one, do the following:

1. On the Graphics tab of the ribbon, in the Data Objects group, click Grid.
2. Click on the screen worksheet, and then draw a box of the desired size (while holding down the mouse button).
3. Release the mouse button, and the Grid Object will display.

![Sample Grid Object]

**Note:** This object cannot be moved behind active objects or shapes on a project screen. In other words, it must always be in front. For more information, see Bring to front / Send to back on page 249.

Right-click on the Grid Object, and select Properties from the menu. The Object Properties dialog will open. Use this dialog to configure the Grid Object’s parameters:

![Object Properties: Grid]

- **Data Source:** Select the data source type. The object supports three data sources:

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text File</td>
<td>Displays data from a text file in the ASCII or Unicode format (e.g., CSV text files).</td>
</tr>
<tr>
<td>Class Tag</td>
<td>Displays values from a Class Tag, where the members of the tag are fields (columns) of the grid object, and each array position is one row of the grid object.</td>
</tr>
<tr>
<td>Database</td>
<td>Displays data from an SQL Relational Database, using ADO (ActiveX Database Object) to exchange data with the database.</td>
</tr>
</tbody>
</table>

- **Data source settings:** Click to launch the Data dialog, where you can specify a data source for the Grid object.
- **Columns:** Click to launch the Columns dialog, where you can configure the settings (such as label, column, width, etc.) for the columns of the Grid object.
- **Advanced:** Click to launch the Advanced dialog, where you can configure several settings for the Grid object.
- **Fonts:** Click to launch the Fonts dialog, where you can configure the font settings for the text displayed in the Grid object.
Tip: By default, the same text color is used for both the header and the body of the grid. If you want to set a different text color for the header, then manually edit the project file (projectname.APP) to add the following setting:

```plaintext
[Objects]
GridHeaderTextColor=value
```

`value` must be a hexadecimal RGB color value, such as `FF0000` for red. Please note that this setting will apply to all grid headers in your project.

- **Highlight**: Select a background color for the selected row, during runtime.
- **Text**: Select a text color for the selected row, during runtime.
- **Odd lines**: Select a background color for the odd rows.
- **Even lines**: Select a background color for the even rows.
- **Disable**: You can enter an expression in this field to disable data input or action by the user.
- **E-Sign**: When you check this option, the user will be prompted to enter an electronic signature before entering or modifying data on the object.
- **Security**: Enter the security system access level required for the object/animation.
- **Virtual keyboard**: Select a Virtual Keyboard type used for this object. The option `<Use Default>` selects the default Virtual Keyboard configured in the `Viewer` settings (`Viewer` on the Project tab of the ribbon). You can also specify a different Virtual Keyboard for this Grid object.

**Data dialog**

This dialog allows you to configure the data source for a Grid object.

**Grid Data – Text File**

When the **Data Source** type is set to Text File, you can configure the following settings:

- **File**: Enter the name of the text file source. You can either type the file name and its path or click the ... button to browse for it. (If the file is stored in your project folder, you can omit the path in the name.)

  **Tip**: You can configure tag names between curly brackets `{TagName}` in the **File** field.

  - **Delimiters**: Set the delimiter(s) used in the data source file. For instance, if the data will be read from a CSV (comma separated values) file, you would select the **Comma** option. You can even choose a custom delimiter by checking the **Other** option and typing the custom delimiter in the field beside it.
  - **Read only** checkbox: When this option is checked, the Grid object will only read data from the specified file. The object will not write anything to the file.
Grid Data – Class Tag
When the Data Source type is set to Class Tag, you can configure the following interface:

- **Class Tag**: Enter the name of the main class tag source. (Do not specify a specific member of the class tag.) You can specify the initial array position in this field (e.g., Mytag[10]); otherwise, 0 (zero) will be used as the initial position by default.
- **Number of Items**: Enter the number of array positions from the Class Tag that should be displayed.
- **View**: When the tag configured in the optional field changes value (e.g., toggles) during runtime, the grid object launches a dialog, allowing the user to show/hide each column or modify their positions.

Grid Data – Database Configuration
When the Data Source type is set to Database, you can configure the following settings:

Please refer to the Database Configuration dialog for further information about this dialog.
**Columns dialog**

You can configure the settings for each column displayed by the Grid object, as follows:

![Columns dialog box]

**Column**

The ID Number defines the position of the column in the table.

**Label**

Type a label for the column, which will be displayed in the heading row of the grid.

**Tip:** To dynamically change the label during run time, configure a project tag in curly brackets (e.g., `{MyTag}`). Also, when the label is blank, the width of the entire column is set to 0. This is useful to hide columns during run time.

**Member**

Enter the name of the class member to which this column will be linked. If this field is left in blank, then the text configured in the Label field will be used as a default member name.

This setting is available only when the Data source type is set to Class Tag.

**Field**

Enter the name of the field (column) in the database table to which this column will be linked. If this field is left in blank, then the text configured in the Label field will be used as a default field name.

This setting is available only when the Data source type is set to Database.

**Type**

Select the type of interface that will be used in the column. The options are:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Displays alphanumeric values.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Displays numeric values.</td>
</tr>
<tr>
<td>Picture</td>
<td>Displays the picture (*.bmp or *.ico format) from the data source. For instance, if the value from the data source is MyFile.bmp, the grid object will display the picture from the file MyFile.bmp stored in your project folder. The picture will be automatically resized to fit the cell of the grid object. The picture file(s) must be stored in the</td>
</tr>
</tbody>
</table>
Web sub-folder of your project folder to support this feature on the Thin Client stations. Projects running on
Windows Embedded support pictures in bitmap format (*.bmp), but not in icon format (*.ico).

Displays checkbox interfaces. The checkbox will be unchecked if the value read from the file is 0, <NULL> or
"FALSE"; otherwise, the checkbox will be checked. By default, IWS will use the value 0 for unchecked and the
value 1 for checked.

Displays the value in the Time format (e.g., HH : MM : SS).
This setting is available only when the Data Source type is set to Database.

Displays the value in the Date format (e.g., MM/DD/YYYY).
This setting is available only when the Data Source type is set to Database.

Displays the value in the Date/Time format (e.g., MM/DD/YYYY HH : MM : SS).
This setting is available only when the Data Source type is set to Database.

Displays the value in the Time format (e.g., HH : MM : SS). Also, the time is assumed to be in Coordinated
Universal Time (UTC), and it is automatically converted to the current time zone in the Viewer module / thin client.
This setting is available only when the Data Source type is set to Database.

Displays the value in the Date format (e.g., MM/DD/YYYY). Also, the date is assumed to be in Coordinated
Universal Time (UTC), and it is automatically converted to the current time zone in the Viewer module / thin client.
This setting is available only when the Data Source type is set to Database.

Displays the value in the Date/Time format (e.g., MM/DD/YYYY HH : MM : SS). Also, the date and time are
assumed to be in Coordinated Universal Time (UTC), and they are automatically converted to the current time
zone in the Viewer module / thin client.
This setting is available only when the Data Source type is set to Database.

- When the Data Source type is set to Database, it is important to make sure that
  the Type for each column configured in the object matches the Type of the
  respective field in the database.
- When the Data Source type is set to Database, you can configure valid SQL
  statements directly in the field (e.g., List(DISTINCT [Cell_Name]) AS [Cell Name]). You can also configure tag names between curly brackets to
  modify this setting during run time (e.g., {MyFieldName}).

- Tip: If Picture is the column type, the Grid object displays a default icon if the
  picture file is not found during run time. You can configure a different picture to
  be displayed when the file is not found by copying the picture file to the Web sub-
  folder of your project folder and configuring its name on the project_name.app
  file, as follows:

        [Objects]
        GridDefaultPicture=PictureFileName

Width

Enter a width of the column, in pixels.
If you want to dynamically adjust the width during project run time, type the name of an Integer
tag enclosed in curly brackets (e.g., {Column1Width}). When the value of the tag changes, the
width is changed to match.

Align

Select an Alignment for the data shown in the column. There are three options: Left, Right or
Center.
**Input**
Select to allow the user to enter data in this column during run time.

**Key**
Use this field to designate a shortcut for sorting the values. A shortcut is a combination of keys pressed at the same time (e.g., CTRL+C, CTRL+V, etc.). This option is especially useful when creating projects for target systems that do not provide a mouse or touchscreen interface and have only a keyboard for interacting with the project during run time.

**Unit**
Enter the name of the engineering unit (i.e., the unit of measurement), if any, that applies to the data displayed.

If you want to dynamically change the unit during project run time, type the name of a String tag enclosed in curly brackets (e.g., {Column1Unit}). When the value of the tag changes, the unit is changed to match.

**Decimal Points**
Enter the number of decimal places to be displayed.

If you want to dynamically change the number of decimal points during project run time, type the name of an Integer tag enclosed in curly brackets (e.g., {Column1Decimals}). When the value of the tag changes, the number of decimal points is changed to match.

**Show ID Column**
Select to display the row numbers.

**Allow sorting columns**
Check to enable the user to sort the values in the columns during run time, either by clicking on the label or by using the shortcut configured for each column.

This option is unavailable if the option Show header in the Advanced dialog box is not selected.

---

**Note:** When the Data Source type is set to Class Tag, and the Columns dialog box is left blank, the object displays the values from all members of the Class tag with the following default settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>The name of the member from the Class tag.</td>
</tr>
<tr>
<td>Type</td>
<td>Text</td>
</tr>
<tr>
<td>Width</td>
<td>The minimum size to display the name of the member from the Class tag on the header row of the Grid object.</td>
</tr>
<tr>
<td>Align</td>
<td>Center</td>
</tr>
<tr>
<td>Input</td>
<td>Enabled (selected).</td>
</tr>
<tr>
<td>Key</td>
<td>None.</td>
</tr>
<tr>
<td>Unit</td>
<td>The unit of the member from the Class tag. (The unit of a tag or member is set using the Tag Properties tool.)</td>
</tr>
</tbody>
</table>

**Tip:** Use the Move Up and Move Down buttons to reorder the columns.
**Advanced dialog**

This dialog allows you to configure the advanced settings for a Grid object.

- **User Enable**: If the value of this tag is TRUE (different from 0), the user can select different rows of the object by clicking on them during runtime. This field can be configured with a tag or with a numeric value.

- **Selected Values**: The values from each column of the selected row are written to each position of the array tag configured in this field. Moreover, you can modify the value of the cells currently selected in the Grid object by changing the value of array tag configured in this field. The initial array position (offset) can be configured in this field.

- **Number of Rows**: The grid object writes the number of rows currently available in the grid object to the tag configured in this field.

- **Row Number**: The Grid object writes the number of the row currently selected during runtime. In addition, you can select different rows by writing their values in this tag.

- **Condition**: Enter an expression to filter the grid data; only rows that match the expression will be displayed. The expression must use the following syntax:

  **Column Operator Value**

  For example...

  \[ \text{ColumnX} > 200 \]

  When **Data Source** (in the Grid Object Properties dialog) is set to Text File or Class Tag, the **Column** is the value specified in the Label. When **Data Source** is set to Database, the column is the value specified in the Field. (In this case, if the Field is left blank, then the column value specified is the Label.)

  Also, expressions for Database must be formatted like a SQL Where statement. The following table shows which operators should be used:

  **Condition Expression Operators**

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Data Source is Text File...</th>
<th>Data Source is Database...</th>
</tr>
</thead>
<tbody>
<tr>
<td>equal to</td>
<td>=</td>
<td>LIKE</td>
</tr>
<tr>
<td>not equal to</td>
<td>&lt;&gt;</td>
<td>NOT LIKE</td>
</tr>
<tr>
<td>wildcard, single character</td>
<td>?</td>
<td>_</td>
</tr>
</tbody>
</table>
Comparison | Data Source is Text File… | Data Source is Database…
--- | --- | ---
wildcard, unlimited characters | * | %

As such, the following expression for Text File…

C1 = 'ab?d'

...means the same as the following expression for Database...

C1 LIKE 'ab_d'

Finally, you can combine several expressions simultaneously in the Condition field, using the logic operators AND, OR, and NOT. For example:

ColumnAge > '10' OR ColumnName = 'John' AND ColumnDate > '05/20/2003'

Tip: You can configure tags between curly brackets {TagName} in the Condition field to change the filtering condition during runtime.

• **Print Trigger:** When the tag configured in this field is toggled, the current state of the Grid object is sent to the default printer.

• **PDF Trigger** field: When the tag configured in this field is toggled, the current state of the Grid object is saved as a PDF file at the location specified by PDF Filename.

• **PDF Filename** field: Enter a complete file path and name where the PDF file is to be saved. You can also enter a tag name using the {tag} syntax.

Note: PDF Trigger and PDF Filename are not supported in projects running on Windows Embedded or Thin Client.

• **Multiline** checkbox: When this option is checked, the print output or PDF will be formatted according to the available column space, and the text within each cell will be wrapped so that all of it is shown.

• **Reload:** When the tag configured in this field is toggled, the object reloads the data from the data source and displays it.

• **Save Trigger:** When the tag configured in this field is toggled, the data source (Text File or Database) is updated with the current values of the grid object. (This field is not available when the Data Source type is Class Tag, because the values are automatically updated in the tags as you change a cell in the grid.)

• **Insert Trigger:** When **Auto refresh after insert trigger** is enabled (checked), the tag configured in this field is used as a trigger to refresh the database table. Whenever the value of the tag changes, a new row is added to the table and the values of the array configured in the Inserted Values field are automatically inserted.

• **Inserted Values**: If the Insert Trigger is being used, then the array tag configured in this field provides the values that will be inserted. This field must only contain an array tag, although it can be of any size.

• **Save on data change**: When this option is checked, the values are updated on the data source (Text File or Database) as soon as the user enters a new value on the grid, during runtime. (This option is disabled when the Data Source type is Class Tag, because the values are automatically updated in the tags as the user changes the value of the cells in the grid.)

• **Enable Slider/Resize**: If this box is not checked, the user is unable to scroll the list by dragging the slider button, or to change the cell’s size during runtime.

• **Conditional checkbox**: When this option is checked, the user cannot check a checkbox on the Grid during runtime, unless all preceding checkboxes in the same column are also checked. This option is especially useful when you want to oblige the user to follow a pre-defined sequence. This field is not available when the Data Source type is Class Tag.

• **Show Header**: When this option is checked, the header of the Grid object is visible during runtime, displaying the label of each column.
• **Show gridlines**: When this option is checked, the gridlines of the Grid object are visible during runtime.

• **Ext. translation**: When this option is checked, the text displayed by the Grid object will be susceptible to the Translation Tool during runtime.

• **Disable TAB to navigate through cells**: When this option is checked, the user can only navigate through the cells of the Grid Object with the arrow keys, rather than the Tab key. You should disable the Tab key for navigation if you want it to be used for switching to the next object that supports focus on the screen.

• **Concatenate Label for picture**: When this option is checked, the reference name for the picture is the result of the concatenation of the name in the Field column with the value of the Label column. The result will be `<Label name>_<Field value>`.

• **Export**: This interface allows you to export the data from the grid object to a class-array tag, regardless of the Data Source selected for the object. The following fields must be configured to support this feature:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class tag</td>
<td>Type the main tag name of the class-array tag that will receive the exported values. Each row from the grid object will be exported to one array position of the array tag, by matching column labels. The initial array position can be configured in this field; 0 is the default.</td>
</tr>
<tr>
<td>Trigger</td>
<td>When the tag configured in this field changes value (e.g., toggles), the data is exported from the Grid object to the class-array tag configured in the Class tag field.</td>
</tr>
</tbody>
</table>

**Tip:** The Export feature is an easy and powerful tool to transfer data from different data sources to tags. After exporting the data to tags, you can use different tasks to manipulate the data, such as the FileWrite() function, or the Recipe or Report tasks to save the data in text files (e.g., CSV files).

• **Auto Format**: When this option is checked, decimal values in columns of Numeric type will be formatted according to the virtual table created by the SetDecimalPoints() function. This option will work only in columns for which Decimal Points are not already configured. For more information, please see Grid Object: Columns dialog.
Background Tasks

Background tasks are, as the name implies, project features that run in the background, as opposed to the graphical screens with which the user interacts.

The background tasks are executed by the Background Tasks module (see Execution Tasks), and they are defined by task worksheets in the Project Explorer.
Background Tasks

Alarm worksheet

The Alarms folder enables you to configure alarm groups and tags related to each group. The Alarm worksheet defines the alarm messages generated by the project. The primary purpose of an alarm is to inform the operator of any problems or abnormal condition during the process so he can take corrective action(s).

The Alarm worksheet is executed by the Background Task module (see Execution Tasks). It handles the status of all alarms and save the alarm messages to the history, if configured to do so, but it does not display the alarm messages to the operator; the Alarm/Event Control screen object, available on the Graphics tab of the ribbon, must be created and configured in a screen in order to display alarms.

To create a new Alarm worksheet, do one of the following:

• On the Insert tab of the ribbon, in the Task Worksheets group, click Alarm;
• Right-click the Alarms folder in the Project Explorer, and then click Insert on the shortcut menu; or
• Click New on the Application menu, click the File tab, and then select Alarm Worksheet.

To edit an existing Alarm worksheet, double-click it in the Project Explorer.

You can create multiple Alarm groups (worksheets) and each group can be configured with independent settings, such as message colors, history log enabled/disabled, and so forth.

Each Alarm worksheet is composed of two areas:

• **Header:** Settings applied to all tags and alarms configured in the same alarm group. These settings allow you to configure the formatting of the message and the actions that must be triggered based on alarm events (e.g., print alarms, send alarms by email, and so forth). For more information, see Header Settings.

• **Body:** Configure alarm messages and associate them to conditions linked to tags. For more information, see Body Settings.

**Note:**

• You can configure the Alarm Group to send notifications by Email automatically, based on alarm events. For more information, see Email Settings.

• The alarm properties associated to each tag (configured in the body of the alarm group) can also be edited by the Tag Properties dialog (Properties on the Home tab of the ribbon). However, before associating a tag to an alarm group, it is necessary to create the alarm group and configure the settings on its header, which will be applied to all tags associated to the group.

• As of IWS v6.1+SP2, the Alarm task has been modified to avoid automatically acknowledging alarms by another alarm. For example, the Hi (Lo) alarm should not be automatically...
acknowledged when the HiHi (LoLo) alarm becomes active. To enable the previous behavior, set
the following key in your project (.APP) file:

```
[Alarm]
UseLegacyPriorityAck=1
```

**Note:** The settings configured in the body of each Alarm worksheet are stored in the Tags Database
archive(s). Therefore, changes to the tags database may affect the content of the Alarm worksheets
(body). Notice that each tag/type cannot be available in more than one Alarm group simultaneously
because the Alarm Group is a property associated to each Tag/Alarm Type (e.g., Tag: Level; Alarm
Type: Hi; Alarm Group: 2).
The Trend folder enables you to configure history groups that store trend curves. You can use the Trend worksheet to declare which tags must have their values stored on disk, and to create history files for trend graphs. The project stores the samples in a binary history file (*.hst), and shows both history and on-line samples in a screen trend graph.

The Trend worksheet is executed by the Background Task module (see Execution Tasks). It handles the saving of trend data to the history, but it does not display that data to the operator; the Trend Control screen object, available on the Graphics tab of the ribbon, must be created and configured in a screen in order to display trend data.

To create a new Trend worksheet, do one of the following:

• On the Insert tab of the ribbon, in the Task Worksheets group, click Trend;
• Right-click the Trends folder in the Project Explorer, and then click Insert on the shortcut menu; or
• Click New on the Application menu, click the File tab, and then select Trend Worksheet.

To edit an existing Trend worksheet, double-click it in the Project Explorer.

The Trend worksheet is divided into two areas:

• Header area (top section), which contains information for the entire group
• Body area (bottom section), where you define each project tag in the group. This section contains several columns (only two are shown in the preceding figure).

Header

Configure the following settings in the worksheet header:

Description
Type a description of the worksheet for documentation purposes.

History Format
Click the arrow button to select a trend history format from the list. The available options are:

Proprietary
Save trend history in a proprietary, binary file. The file is saved in your project folder (on the project runtime server) at: [...]\projectname\Hst\GGYYMMDD.hst

• GG = Trend worksheet number
• YY = Last two digits of the year
• \textbf{MM} = Month
• \textbf{DD} = Day

A new history file is created for each calendar day that the project runs.

The utility programs \texttt{HST2TXT.EXE} and \texttt{TXT2HST.EXE} are provided in order to convert history files from binary (\texttt{*.hst}) to plain text (\texttt{*.txt}) and vice versa. For more information, see \textit{Converting Trend History Files from Binary to Text} on page 311 and \textit{Converting Trend History Files from Text to Binary} on page 312.

\textbf{Database}

Save trend history in an external SQL database of your choice. After you select this format, click \textbf{Database Configuration} to open the \textit{Database Configuration} dialog box, where you can configure the connection to the database. For more information, see \textit{Database Configuration} on page 98 and \textit{Database Interface} on page 605.

By default, the history is saved in the table \texttt{TRENGGGG} (\texttt{GGG} = Trend Worksheet Number; e.g., \texttt{TREND001} for the Trend Worksheet 001).

\textbf{Wonderware Historian}

Save trend history in a Wonderware Historian database of your choice. After you select this format, click \textbf{Historian Configuration} to open the \textit{Wonderware Historian} dialog box, where you can configure the connection to the database. For more information, see \textit{Connect to a Wonderware Historian database} on page 310.

The trend history for each project tag is saved separately in the Historian database, but you can use \texttt{Prefix} in the connection settings in order to keep the tags grouped together.

\begin{tabular}{|l|}
\hline
\textbf{Note:} You can specify String tags in many fields of the Trend worksheet, to change those values during run time, but doing so may affect how those values are saved in the trend history:
\begin{itemize}
\item When the history format is \textbf{Proprietary}, the value of the String tag is converted to a numerical value (if possible) and then saved in the history file. If numeric conversion is not possible, then a value of 0 is saved.
\item When the history format is \textbf{Database} or \textbf{Wonderware Historian}, the actual value of the String tag is saved in the database.
\end{itemize}
\hline
\end{tabular}

\textbf{Save On Trigger}

Click (enable) and type a tag name to save trend samples when someone changes the specified tag. (Tag change can be an event from the Scheduler.)

\textbf{Save On Tag Change}

Click (enable) to always save the trend sample when a value change occurs in any of the tags from that group.

\textbf{Advanced}

Click to display the Trend Advanced Settings dialog. For information about completing the fields in this window, see \textit{Batch History Configuration}.

\textbf{Body}

For each project tag, configure the following settings in the worksheet body:

\textbf{Tag Name}

The name of the project tag for which trend history will be saved.

\textbf{Dead Band}

Type a value to filter acceptable changes when \textbf{Save on Tag Change} is used. For example, Dead Band has value \texttt{= 5}. If the tag value is 50 and changes to 52, the system will not register this variation in the database, because it is less than 5. If the change is equal to or greater than 5, the new value will be saved to the history file.

\textbf{Field}

When \textbf{History Format} is \textbf{Database}, this is the name of the field (in the SQL database table) where the trend history will be saved. If this field is left blank, the project tag name will be used.
For array tags and classes, special characters ([ ] .) will be replaced by underscores (_), as shown in the examples below:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyArray[1]</td>
<td>MyArray_1</td>
</tr>
<tr>
<td>MyClass.Member1</td>
<td>MyClass_Member1</td>
</tr>
<tr>
<td>MyClass[3].Member2</td>
<td>MyClass_3_Member2</td>
</tr>
</tbody>
</table>

**Historian Tag**

When **History Format** is **Wonderware Historian**, this is the name of the tag (in the Historian database) where the trend history will be saved. If this field is left blank, the project tag name will be used.

**Note:** The Trend task can accept only up to 240 tags in a single worksheet. If you manually configure more than 240 tags in the same worksheet, the Trend task will generate an error when you run the finished project.
Recipe worksheet

A Recipe worksheet is used to load tag values from or save tag values to an external file during runtime. It is typically used to execute process recipes, but you can store any type of information (such as operation logs, passwords, and so on) in the external file.

To create a new Recipe worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click Recipe;
- Right-click the Recipes folder in the Project Explorer, and then click Insert on the shortcut menu; or
- Click New on the Application menu, click the File tab, and then select Recipe Worksheet.

To edit an existing Recipe worksheet, double-click it in the Project Explorer.

The worksheet is divided into two areas:

- **Header area** (top section), which contains information for the whole group
- **Body area** (bottom section), where you define each tag in the group.

Use the **Header** parameters on this worksheet as follows:

- **Description field**: Type a description of the worksheet, for documentation purposes.
- **File Name field**: Type the name of the external file, using static text ([File1] or a indirect tag ([{FileNameTag}]).
- **Register Number field**: Type a tag to define the register number to be read or written into a DBF file. (This field is for legacy purposes only and is no longer used.)
- **Save As XML**: Select (check) to save information in XML format, or deselect (uncheck) to save in DAT format.
  You can load information in a .DAT file into different tags using a second Recipe worksheet, but you must load information in an .XML file into tags with the same name as the tag from which the data originated.
  - **Note**: As with HTML pages, you must be running the Web server to view XML data from the Web. Unlike the HTML pages in the runtime system, XML pages do not need to have the project running to view the XML data. (You must be running Internet Explorer version 5.0 or higher to view XML data.)
- **Unicode checkbox**: Click (enable) to save the recipe in Unicode format (two bytes per character) or (disable) to save the recipe in ANSI format (one byte per character).
  - **Note**: When saving a worksheet, you can save it using any name you choose (you are not required to use a predefined file name). A configuration file using the default extension .RCP (or .XSL if you specify Save As XML) contains the recipe configuration and the **File Name** field contains the data file name to be read or written.
Use the *Body* parameters on this worksheet as follows:

- **Tag Name** field: Type tag names to update with file contents or with values to write to a file. If the tag is an array, you must specify the first position to use.
  
  If the tag is an Array or a Class (or both), then the development application automatically enables every array position and class member by default. To configure a specific array position and/or class member, type it in the **Tag Name** field as normal. For example, `level[3].member`.

- **Number of Elements** field: Specify how many positions of the array tag are in use.

  **Tip:** You can configure a tag name between curly brackets (`{tagname}`) in this field, allowing the user to dynamically change the Number of Elements configured in the Recipe for each array tag, during runtime.

To execute a Recipe worksheet, use the **Recipe** function anywhere an expression is allowed.
A Report worksheet is used to design a report that is dynamically generated during runtime (using the current values of the included tags) and then either sent to a printer or saved to a file.

To create a new Report worksheet, do one of the following:
- On the Insert tab of the ribbon, in the Task Worksheets group, click Report;
- Right-click the Reports folder in the Project Explorer, and then click Insert on the shortcut menu; or
- Click New on the Application menu, click the File tab, and then select Report Worksheet.

To edit an existing Report worksheet, double-click it in the Project Explorer.

The Report worksheet is divided into two areas:
- **Header** area (top section), which contains information for the whole group; and
- **Body** area (bottom section), where you define each tag in the group.

Use the Header parameters on this worksheet as follows:
- **Description** field: Type a description of the worksheet for documentation purposes.
- **Output File** field: Type a tag name for the output file (using the {tag} syntax) where data is stored when you are printing to a file. Where the tag value is part of the file name.
  
  For example: report{Day}.out. Where the generated file might be report1.out, report2.out, report3.out, and so on, according to the tag day value.

  **Note:** A report configuration file uses .RCP as the default extension. The Output File field is the file where data is stored.

- **Edit RTF file** button: Click to access the report as an RTF file, which you can edit for layout modification and so forth.
- **Disk Append** checkbox: When printing to a file
  - Check the box to add (amend) the new report to the end of an existing file
  - Uncheck the box to replace the existing report in that file with the new report
- **Unicode** checkbox: Click (enable) to save the report in Unicode format (two bytes per character) or (disable) to save the report in ASCII format (one byte per character).
- **Lock Value into the {Tag/Exp} length** checkbox: Click (enable) to automatically truncate the values of Tags/Expressions in the report to fit between the curly brackets, as they are positioned in the Body of the report (see below). This helps to preserve the layout of the report. If this option is left unchecked, then the full values of Tags/Expressions in the report will be displayed.
Use the *Body* portion of this worksheet for report formatting. You can configure a report using data in the system and indicating where to print the tag values. Each tag name will replace the `{tag_name}` tag name. For Real type tags, use the following syntax: `{tag_name n}`, where *n* is the number of decimal places you want printed.

**Note:** If you are using the standard report editor (text only: ASCII or Unicode), then the number of characters reserved for the tag value will be equal to the number of characters used to type the tag name (including the two "curly" brackets). For example, if you configure `{TagA}` in the report body, reserve six characters for the tag value in the report file. This behavior is not valid for reports in RTF format.

To execute a Report worksheet, use the *Report* function anywhere an expression is allowed.
ODBC worksheet

The ODBC interface runs in a network environment and uses the standard Windows ODBC configuration. The ODBC task is capable of data interchange between IWS and any database supporting this interface.

**Note:** In addition to configuring the ODBC worksheet, you must configure the Windows ODBC standard driver. IWS refers to the User DSN (Data Source Name), which you configure through the Control Panel. For more information, refer to your Windows documentation. Also, the ODBC interface is not available for projects running on Windows Embedded target systems.

To create a new ODBC worksheet, do one of the following:
- On the Insert tab of the ribbon, in the Task Worksheets group, click ODBC;
- Right-click the ODBC folder in the Project Explorer, and then click Insert on the shortcut menu; or
- Click New on the Application menu, click the File tab, and then select ODBC Worksheet.

To edit an existing ODBC worksheet, double-click it in the Project Explorer.

A new worksheet displays, as follows.

ODBC worksheets are executed under the ODBC Runtime task. However, creating a new worksheet does not automatically enable the task; you must use the Execution Tasks dialog (Tasks on the Home tab of the ribbon) to configure the task to start at runtime. For more information, please see Execution Tasks.

The ODBC worksheet is divided into two areas:
- **Header area** (top section), which contains information for the whole group, defines tags to start read and write events, sets return values, handles database access parameters, and so forth; and
- **Body area** (bottom section), where you define each tag in the group and relate tags to fields in the current register from the database table.

Use the **Header** parameters on this worksheet as follows:
- **Description** field: Type a description of the worksheet for documentation purposes.
- **Data Source Name** field: Type the same Data Source Name (DNS) specified in the Windows Control Panel containing information about specific database access.
- **User** field: Type a user name to access to the database.
- **Password** field: Type the user's password.
- **Table** field: Type a table name in the database.
**Background Tasks**

- **Condition** field: Type a search condition or filter.
- **Status** field: Type a return value (fill in with a tag name). The tag should report 0 for success and use another value for an error code.
- **Transaction Completed** field: Type a tag that changes value when the transaction is executed.
- **Select, Next, Insert, Delete, or Update Trigger** fields: Type a tag to work as a trigger, where each value change causes the system to execute the corresponding command. At least one trigger field is required.

Use the **Body** parameters on this worksheet as follows:

- **Tag Name** field: Type the names of tags to update with file contents or tags whose values should be written to a file.
- **Column**: Type the location in which to find data in the file (for example, R3CH corresponds to Row 3, Column H of an Excel sheet)

You must use the Windows Control Panel to set up the ODBC interface for Excel files. The procedure is as follows:

1. Click the **Start > Settings > Control Panel**.
2. When the **Control Panel** window displays, double-click on the ODBC icon to open the *ODBC Data Source Administrator* dialog.
3. In the *ODBC Data Source Administrator* dialog, click Excel Files in the User Data Sources list, and then click the Configure button.
4. When the *ODBC Microsoft Excel Setup* dialog displays, type the Windows configuration name to be used in the DSN field on *ODBC* worksheet into the Data Source Name field.
5. Click the Select Workbook button to configure the Excel file you want to use.
6. Return to the ODBC Data Source Administrator dialog and verify that your User DSN displays in the list. Click **OK to close** the dialog.
7. After configuring the ODBC Windows interface, you must configure the project's ODBC worksheets.
8. From the **Tasks** tab, insert a new ODBC worksheet.
9. Be sure you set the ODBC Runtime to start automatically using the *Execution Tasks* dialog (**Tasks** on the **Home** tab of the ribbon).

To start this configuration, you simply need to run the project. Your project will handle the **Select, Next, Insert, Delete, and Update** triggers to allow data to exchange throughout rows in Excel and tags configured in the worksheet.

Consult your Windows documentation for the meaning of specific error codes.

The following is a list of IWS error codes:

- **Select command**
  - 1 - Error in the ODBCPREPARE function.
  - 2 - Error in the ODBCBindCol function.
  - 3 - Error in the ODBCExecute function.
  - 4 - Error in the ODBCSetch function.
- **Next command**
  - 5 - Error in the ODBCSetch function.
- **Insert command**
  - 6 - Error in the ODBCPREPARE function.
  - 7 - Error in the ODBCExecute function.
  - 8 - Error in the ODBCCommite function.
- **Update command**
  - 9 - Error in the ODBCPREPARE function.
• 10 - Error in the ODBCEXECUTE function.
• 11 - Error in the ODBCCOMMIT function.
• Delete command
  • 12 - Error in the ODBCprepare function.
  • 13 - Error in the ODBCEXECUTE function.
  • 14 - Error in the ODBCCOMMIT function.
Math worksheet

A Math worksheet is used to implement program logic (using the Built-in Scripting Language) that should be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.

**Note:** The Math worksheet is functionally similar to the Script worksheet, except that it uses the Built-in Scripting Language instead of VBScript.

To create a new Math worksheet, do one of the following:
- On the Insert tab of the ribbon, in the Task Worksheets group, click **Math**;
- Right-click the **Math** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the Application menu, click the **File** tab, and then select **Math Worksheet**.

To edit an existing Math worksheet, double-click it in the Project Explorer.

The Math worksheet is divided into two areas:
- **Header area** (top section), which contains information for the whole group; and
- **Body area** (bottom section), where you define each tag, expression, and Programming Lines (logical routines and mathematical calculations through functions and logical operations) in the group.

Use the **Header** parameters on this worksheet as follows:
- **Description** field: Type a description of the worksheet for documentation purposes.
- **Execution** field: Type an expression, a single tag, or a constant value to determine when the worksheet should execute.

**Note:** The project executes the worksheet only when the **Execution** field result is *not zero*. If you always want the worksheet to execute, type a 1 (constant value) in the **Execution** field.

Use the **Body** parameters on this worksheet as follows:
- **Tag Name** field: Type a tag to receive a return value from the specified calculation in the **Expression** column.
- **Expression** field: Type an expression to return a return value to the specified tag in the **Tag Name** column.

To execute a Math worksheet at a specific time, separate from whatever is configured in the **Execution** field, use the **Math** function anywhere an expression is allowed.
A Script worksheet is used to implement program logic (using VBScript) that should be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.

**Note:** The Script worksheet is functionally similar to the Math worksheet, except that it uses VBScript instead of the Built-in Scripting Language.

To create a new Script worksheet, do one of the following:
- On the Insert tab of the ribbon, in the Task Worksheets group, click **Script**;
- Right-click the **Script** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the Application menu, click the **File** tab, and then select **Script Worksheet**.

To edit an existing Script worksheet, double-click it in the Project Explorer.

The code configured in each Script worksheet is executed by the Background Task. The project scans the worksheets sequentially (based on the worksheet number) and executes only the groups in which the condition configured in the **Execution** field of the worksheet is TRUE (i.e., non-zero).

**Note:** You must use the syntax supported by the Built-in Scripting Language in the **Execution** field. Only the body of the worksheet supports VBScript.

Variables declared in the worksheet have local scope for that specific group only. They are not available for any other VBScript interface.

You cannot define procedures (i.e., functions and subs) in the Script worksheet. However, you can call procedures defined in the **Global Procedures** or in the **Startup Script**.

Example:

```vbs
' Variables available only for this group can be declared here
Dim myVar, myTest
myTest = 1

' The code configured here is executed while the condition configured in the Execution
' field is TRUE
myVar = $FindFile("c:\*.txt")
If MyVar > 0 Then
   $TagNumOfFiles = myVar
End If
```

**Caution:** When any Script worksheet is saved during runtime (on-line configuration), the Startup Script will be executed again and the current value of the local variables of any Script worksheet will be reset.
**Startup Script worksheet**

The Startup Script worksheet is a VBScript interface that is automatically executed when the project is run. To edit the Startup Script worksheet, double-click it in the Project Explorer. (It is located on the Tasks tab, in the Script folder.) The worksheet is displayed:

![Startup Script worksheet](image)

The code configured in this worksheet is executed just once when the Background Task module (BGTask) is started. This interface is useful for initializing variables or executing logics that must be implemented when the project is run.

You can declare and initialize variables and define procedures. However, variables or procedures declared in this interface will be available ONLY to the Script worksheets executed by the Background Task module — they are not available to any VBScript interface from the Graphic Module.

Example:

```vbscript
' Variables available for all Script groups from the Script task can be declared and initialized here
Dim MyVar, Counter
MyVar = 100

' Procedures available for all Script groups from the Script task can be implemented here
Function AreaEquTriangle(base, high)
    AreaEquTriangle = (base * high) / 2
End Function

Sub CheckLimits(myValue, myHiLimit, myLoLimit)
    If (myValue > myHiLimit Or myValue < myLoLimit) Then
        MsgBox("Value out of range")
    End If
End Sub

' The code configured here is executed just once when the Background task is started
If $GetOS() = 3 Then
    MsgBox("Welcome! This project is running under Microsoft Windows Embedded operating system.")
Else
    MsgBox("Welcome! This project is running under Microsoft Windows desktop operating system.")
End If
```
Scheduler worksheet

A Scheduler worksheet is used to execute program logic (using the Built-in Scripting Language) at a specific date/time, on a regular time interval, or upon a triggering event.

To create a new Scheduler worksheet, do one of the following:

• On the Insert tab of the ribbon, in the Task Worksheets group, click **Scheduler**;
• Right-click the **Scheduler** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
• Click **New** on the Application menu, click the **File** tab, and then select **Scheduler Worksheet**.

To edit an existing Scheduler worksheet, double-click it in the Project Explorer.

The **Scheduler** worksheet is divided into two areas:

• **Header** area (top section), which contains information for the whole group
• **Body** area (bottom section), where you define each tag, expression, and condition for the group.

Use the parameters on this worksheet as follows:

• **Description** field: Type a description of the worksheet for documentation purposes.
• **Event** drop-down list: Click to select an event type from the following:
  • **Calendar**: Generates time bases greater than 24 hours. For example, You can define an event that prints a report every Friday at a specific time.
  - **Note**: Be sure to complete the **Date** field if you want a specific date for event execution.
  • **Clock**: Generates time bases smaller than 24 hours (intervals in minutes or seconds). This function is frequently used with trend graphics. For example, you can define a tag that will be incremented each hour.
  • **Change**: Event related to the change of a tag in the **Trigger** field.
  - **Note**: This only works for tag changes on the project server, regardless of a tag's defined scope.
  • **Trigger** field: This field is used only with the **Change** Event type. Type the name of a project tag in this field, and when the value of the tag changes, **Expression** is evaluated.
  • **Time** field: This field is used with the **Calendar** and **Clock** Event types.

If the Event type is **Calendar**, then **Time** is a specific time of the day on **Date**. When that **Date** and **Time** occurs, **Expression** is evaluated.

If the Event type is **Clock**, then **Time** is a time interval starting from when the project was run. Every time the interval occurs, **Expression** is evaluated.
Either way, type a time using the **HH:MM:SS.ms** format. Valid values are **00** to **23** for hours, **00** to **59** for minutes, **00** to **59** for seconds, and **1** to **9** for milliseconds. (Milliseconds are optional.) Examples: **03:00:00** is every three hours, **00:00:00.1** is every 100 milliseconds.

- **Date** field: This field is used only with the Calendar Event type. Type a specific date using the **MM/DD/YYYY** format, and when that **Date** and **Time** occurs, **Expression** is evaluated.
  
  Valid values are **01** to **12** for months, **01** to **31** for days, and **1900** to **2099** for years. If the field is left blank, then the event occurs daily at the specified **Time**.

- **Tag** field: Type a tag that will receive the value returned by **Expression** (if any).

- **Expression** field: Type an expression to be evaluated. This field is used by all events.

- **Disable** field: Contains a disable condition for the specified function. Leave this field blank or use an expression value equal to zero to execute the function. Use an expression value equal to one and the function will not execute (Disable = 1).
Database/ERP worksheet

In addition to ODBC, IWS also supports Microsoft .NET ActiveX Data Objects (ADO.NET) for interfacing between the project tags database and other external databases. A Database/ERP worksheet is used to associate project tags with external database fields.

**Note:** For more information about ADO.NET support in IWS — including how to communicate with remote databases using the IWS Database Gateway software — please see Database Interface.

To interface with an external database, you must first configure a connection to the database and then build a worksheet that associates project tags with the database fields.

**Database Connections**

To create a new connection to a target database:

1. In the Project Explorer, open the Database/ERP folder and then right-click on Connections.
2. Choose Insert from the shortcut menu.
   
   The Database Connection dialog is displayed.

3. In the Name field, enter the name that you want to use to reference the target database. You can create multiple database connections, but each connection must have a unique name.
4. In the Connection String field, click the browse button ... to open a standard Data Link Properties dialog. Use the dialog to configure a connection string for the target database.

   **Note:** The list of Database Providers shown in the Data Link Properties dialog depends on the providers actually installed and available in the station where you are running the development application. For more information about using the Data Link Properties dialog, please refer to Windows Help.

5. In the User Name and Password fields, enter an appropriate login for the target database. The login should already be created on the database server, and it should have enough privileges to read from and write to the database tables.
6. If you are connecting to a remote database through the Studio Database Gateway, then click the Advanced button to open the advanced settings dialog, as shown below.

   **Database Connection (Advanced) dialog**
7. In the Host field, enter the IP address of the station that is running the IWS Database Gateway software (STADOSvr.exe). In the Port field, enter the port number on which the software has been configured to run.

Other settings to configure, if necessary:

- **Disable Primary Keys** checkbox: IWS will try to define a primary key to the table in order to speed up the queries. If you are using a database that does not support primary keys (e.g., Microsoft Excel), then you should check this box.

- **Disable Milliseconds in Date/Time Columns** checkbox: IWS will try to include milliseconds when saving a date/time in the database. If you are using a database that does not support milliseconds, then you should check this box.

8. Click **OK** to close the dialog and save the connection configuration.

Database connections are saved as XML files in the \project_name\Config sub-folder. Each file is given the same name as the name of the connection (as entered in the Name field of the Database Connection dialog), with the .XDC file extension. For example, the connection configuration DB1 is saved in the file...

\project_name\Config\DB1.XDC

**Database Worksheet**

> **Note:** This feature emulates Structured Query Language (SQL) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

Database worksheets allow asynchronous execution of database operations, and they offer a user-friendly interface for building SQL commands. Use one of the following methods to create a new database worksheet:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Database**; or

- Right-click on the Database/ERP folder in the Project Explorer, and then click **Insert** on the shortcut menu; or

A new worksheet is displayed, as shown below:

Database worksheets are saved with the .XDB file extension, in the Config sub-folder of your project folder. Each new worksheet is automatically numbered in the order of its creation. For example, the first worksheet created is saved in the file...

[..]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\DB001.XDB

Database worksheets are executed under the **Database Client Runtime** task. However, creating a new worksheet does not automatically enable the task; you must use the **Execution Tasks** dialog (Tasks on the
Home tab of the ribbon) to configure the task to start at runtime. For more information, please see Execution Tasks.

Also, database worksheets run only on the server, and all triggers must be configured with server tags.

**Worksheet Header**

The header of the database worksheet is configured as follows:

- **Description** field: Enter a description of the worksheet, for documentation purposes.
- **Status** field: Enter the name of a numeric tag that will receive status codes for database operations during runtime:

  **Status codes for external database operations**

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Result set is empty</td>
</tr>
<tr>
<td>3</td>
<td>Cursor released and query successfully closed</td>
</tr>
<tr>
<td>2</td>
<td>Beginning of result set reached, usually while moving cursor to previous row</td>
</tr>
<tr>
<td>1</td>
<td>End of result set reached, usually while moving cursor to next row</td>
</tr>
<tr>
<td>0</td>
<td>No errors; status normal</td>
</tr>
<tr>
<td>-1</td>
<td>Error while connecting to specified database (see <strong>Connection</strong> below)</td>
</tr>
<tr>
<td>-2</td>
<td>Error while selecting result set</td>
</tr>
<tr>
<td>-3</td>
<td>Error while moving cursor to next row (see <strong>Next</strong> trigger below)</td>
</tr>
<tr>
<td>-4</td>
<td>Error while moving cursor to previous row (see <strong>Previous</strong> trigger below)</td>
</tr>
<tr>
<td>-5</td>
<td>Error while closing the query (see <strong>Close Query</strong> trigger below)</td>
</tr>
<tr>
<td>-6</td>
<td>Error while inserting rows in result set (see <strong>Insert</strong> trigger below)</td>
</tr>
<tr>
<td>-7</td>
<td>Error while updating result set (see <strong>Update</strong> trigger below)</td>
</tr>
<tr>
<td>-8</td>
<td>Error while deleting result set (see <strong>Delete</strong> trigger below)</td>
</tr>
</tbody>
</table>

- **Completed** field: Enter the name of a numeric tag that will be toggled when database commands are successfully executed.
- **Error Message** field: Enter the name of a string tag that will receive detailed error messages, if errors occur during runtime.
- **Connection** combo-box: Click to select a connection to the target database. All available connections are listed, as configured with the Database Connection dialog described above.
- **Type** combo-box: Click to specify how the result set will be selected for the worksheet:
  - **Table**: Enter a table name and an optional filter condition. (The filter condition is equivalent to the SQL "Where" clause.) All rows of the table that match the filter condition are selected.
  - **SQL**: Enter a custom SQL "Select" statement.

  **Note:** For **Table**, **Condition** and **SQL Statement**, you can enter the names of project tags that contain the desired information. This lets you programmatically change the selection during runtime. However, tag names must be enclosed in curly brackets ({}), to distinguish them from literal strings. Also, you must release an existing selection before you open a new one; see **Close Query** below.

- **Cursor Triggers** area...
  - **Select** field: Enter any tag; when the value of the tag changes, a new cursor opens the first row of the result set and copies those values to the tags configured in the worksheet body.
  - **Next** field: Enter any tag; when the value of the tag changes, the cursor moves to the next row of the result set and copies those values to the tags configured in the worksheet body.
• Advanced button: Click to open the Advanced Cursor Options dialog...

![Advanced Cursor Options dialog]

- Close Query field: Enter any tag; when the value of the tag changes, the cursor releases the result set.
- Previous field: Enter any tag; when the value of the tag changes, the cursor moves to the previous row of the result set and copies those values to the tag configured in the worksheet body.
- Total number of rows field: Enter a numeric tag that will receive the total number of rows in the result set.
- Current row number field: Enter a numeric tag that will receive the number of the current row (i.e., the position of the cursor). When a result set is first opened using the Select trigger, this number is 1. Each Next trigger increments this number, and each Previous trigger decrements it.

• Table Triggers area...

- Insert field: Enter any tag; when the value of the tag changes, a new row is inserted with the current values of the tags configured in the worksheet body.
- Update field: Enter any tag; when the value of the tag changes, all rows of the result set are overwritten with the current values of the tags configured in the worksheet body.
- Delete field: Enter any tag; when the value of the tag changes, all rows of the result set are deleted.

Note: Table triggers are available only when Type is set to Table, because these operations work on the entire table row.

Worksheet Body

In the body of the worksheet, you can map project tags to the columns (fields) of the result set. For each row of the body, enter a Tag Name and its corresponding Column. Which columns are available depends on how the result set is selected, and how it is selected may change during runtime, so be sure to map all necessary columns.

Note: You may have up to 2048 rows per worksheet. If you need more than that, then try creating additional worksheets and adjusting the result set for each worksheet.
Sort or filter the rows in a worksheet

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item.

Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications.

Please note that you can sort or filter rows only in the following types of worksheets:

- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.

Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.

Alarm worksheet rows in their original order

![Alarm worksheet rows in their original order](image)

Alarm worksheet rows sorted by Type

![Alarm worksheet rows sorted by Type](image)
Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.

![Alarm worksheet rows filtered where Tag Name is "Tag3"](image1)

![Alarm worksheet rows filtered where Type is "Lo"](image2)

**Tip:** You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order.
   The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   **Note:** You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.

3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either Tab or Return.
   You may include * and ? as wildcard characters in your string:
   - * matches any number of characters, including none. For example, Tag* would match Tag, Tag3, Tag34567, TagA, and Tag_TEMP.
   - ? matches exactly one character. For example, Tag? matches Tag3 and TagA, while Tag?????? matches Tag34567 and Tag_TEMP.
   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.

4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either Tab or Return.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.
Communication with Other Devices

Communication tasks/worksheets are used to exchange tag values with other IWS projects, remote devices such as PLCs and transmitters, and any other systems that implement supported protocols like OPC and DDE.

To enable communication, configure the worksheets in the Comm tab of the Project Explorer.
About tag integration

Tag integration allows you to directly integrate tags from a third-party control application or device into your IWS project.

Tag integration is an enhanced framework for device communication. It is built on the same communication drivers that are described in the Drivers section, but instead of manually configuring driver worksheets to associate project tags with device registers, you can use the Object Finder to browse the device registers and add them directly to your project.

Registers added in this way appear as tags in the Shared Database folder. They are "live" — that is, they are continuously and bilaterally updated during run time as long as the source is also running and properly connected — and they can be used anywhere in your project that you would normally use project tags.

These tags count against your target system's tag limit only when you use them in your project. This is different from the driver worksheets, where you might create a group of tags in order to associate them with device registers but then never actually use them in your project.

Tip: By default, the project runtime will update integrated tags every 600 milliseconds, which is the rate at which the system tag BlinkSlow toggles. To adjust the rate, manually edit the project file (projectname.APP) to add the following entry:

```plaintext
[Options]
MainDrvAlwaysTrigger=tagname
```

tagname can be either another system tag (e.g., BlinkFast, Second, Minute) or a tag that you have created. Whenever the value of the tag changes, the integrated tags will be updated.

This works because the project runtime automatically creates a virtual Main Driver Sheet to manage the integrated tags. Please keep in mind that the same trigger is used by all Main Driver Sheets in your project.

Integrate tags from TwinCAT

More than simply importing the tags and interface from a TwinCAT PLC program, you can fully integrate your IWS project database with a running TwinCAT PLC so that tags are synchronized between the two systems, without the extra configuration required by an OPC connection or direct communication driver.

Before you begin this task, you must do the following:

1. Configure your TwinCAT project to export a new symbol file with the correct variables. The symbol file is like an index of the TwinCAT tags that you want to integrate into your IWS project. For more information, see Export a symbol file from your TwinCAT project on page 379.

2. Rebuild your TwinCAT project and send it to the PLC.

3. Make sure the PLC is running and available on your network, and note its AMS Net ID and runtime system port number.

4. Copy the new symbol file to your IWS project folder ([…]\Documents\InduSoft Web Studio v7.1 Projects\projectname).

This feature works with all versions of TwinCAT through version 3.x.

To configure TwinCAT tag integration:

1. On the Project tab of the ribbon, in the Settings group, click Communication.

   The Project Settings dialog box is displayed, with the Communication tab selected.

2. In the Tag Integration area, click Add.
The Tag Integration Source dialog box is displayed.

3. In the Type list, select Integrated if it is not already selected.

4. In the Provider list, select TwinCAT if it is not already selected.

5. In the Name box, type a name for the source. This name will be used as a prefix for all tags received from the source. For example, if the source name is DEV, a TwinCAT PLC tag named switch1 would subsequently be named DEV_switch1 in your project.

6. Click Add. The TwinCAT Interface Configuration dialog box is displayed.

7. In the AMS Net ID box, type the AMS Net ID of the TwinCAT PLC that you want to communicate with. For example: 5.0.112.206.1.1.

8. Select the port on which the PLC’s runtime system has been configured to run. You can either select one of the standard ports (e.g., 801, 811, 821, 831) or specify a custom port number.

9. To the right of the Symbol File box, click Browse. A standard Open dialog box is displayed.

10. Use the Open dialog box to locate and select the TwinCAT symbol file (*.TPY) that you exported earlier.

11. Click OK to finish the configuration.

If the configuration is successful, your TwinCAT PLC tags will be immediately available in the Object Finder in InduSoft Web Studio.

EXPORT A SYMBOL FILE FROM YOUR TWINCAT PROJECT

Configure the TwinCAT project development software to export a symbol file that provides information about your TwinCAT project database. The symbol file can then be used to bring your TwinCAT project database into your InduSoft Web Studio project database, via either tag integration or the database import wizard.

The TwinCAT software automatically exports your project database to a symbol file every time you rebuild your project. TwinCAT exports the entire database by default, however, including many system and library variables that InduSoft Web Studio cannot import. You must reconfigure your project options to export only the POUs and Global Variables, and then rebuild your project to generate a fresh symbol file.
**Note:** As of version 2.8, the TwinCAT software exports the symbol file in both *.SYM and *.TPY formats:

- *.SYM is a legacy format that is included for backward compatibility. Beckhoff recommends that it be used only with TwinCAT OPC Server.
- *.TPY is a new, XML-based format that should be used in all other situations, including importing into IWS.

To export the symbol file:

1. Open your TwinCAT project using the TwinCAT software.

2. On the **Project** menu, click **Options**.
   The **Options** dialog box is displayed.

3. In the **Category** list, click **Symbol configuration**.

4. Select **Dump symbol entries** and **Dump XML symbol table**.

5. Click **Configure symbol file**.
   The **Set object attributes** dialog box is displayed.
6. For the sake of expediency, you should first disable the export of all objects and then re-enable only the objects that you want to export — typically, the POUs and Global Variables. Select all of the objects in the tree, and then clear all options for them at the bottom of the dialog box.

![Clearing the options for all objects]

**Tip:** You may need to select Export variables of object in order to activate the other check boxes before clearing them.

7. Reselect only the POUs and Global Variables that you want to export. Do not select libraries. With the objects selected, select all of the options at the bottom of the dialog box.

![Selecting the options for only POUs and Global Variables]

8. Click **OK** to close the Set object attributes dialog box, and then click **OK** again to close the Options dialog.

9. On the **Project** menu, click **Clean All**.

10. On the **Project** menu, click **Rebuild All**.
    The TwinCAT software will rebuild the project, and in the process of rebuilding, it will export a symbol file that contains only the selected POUs and Global Variables.
Once the TwinCAT project is configured to export the selected variables, the next time you send the project to the PLC, it will include the new symbol file and your IWS project to be able to communicate with it.

You can also copy the TwinCAT symbol file to your IWS project folder, which will allow you to browse the tags when the PLC is not running.

**Integrate tags from CoDeSys**

This task describes how to add a CoDeSys 2.x or CoDeSys 3.x project as a tag integration source in your IWS project.

Before you begin this task, you must do the following:

- Configure your CoDeSys project to generate a new symbol file with the correct variables. The symbol file is like an index of the CoDeSys tags that you want to integrate into your IWS project. For more information, see either Configuring your CoDeSys 3.x project or Configuring your CoDeSys 2.x project.
- Rebuild your CoDeSys project and send it to the PLC.
- Make sure the PLC is running and available on your network, and note its IP address and/or runtime address.

If you want to be able to browse the CoDeSys tags offline — that is, when the PLC is not running — then you must also copy the new symbol file to your IWS project folder (\Documents\InduSoft Web Studio v7.1\Projects\projectname).

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**. The **Project Settings** dialog is displayed, with the **Communication** tab selected.

2. In the **Tag Integration** area, click **Add**. The **Tag Integration Source** dialog is displayed.

3. In the **Type** list, click **Integrated**.

4. In the **Provider** list, click **CoDeSys**.
5. In the Name box, type an appropriate name for this tag integration source. Note that the name will be used as a prefix on names of the integrated tags.

6. Click Add. The CoDeSys Tag Integration dialog is displayed.

7. From the Runtime version list, select either CoDeSys 3.x or CoDeSys 2.x, depending on the version of your CoDeSys project.

8. If you selected CoDeSys 3.x, configure the corresponding settings.

![CoDeSys Tag Integration dialog for CoDeSys 3.x](image)

   a) In the Gateway IP and Port boxes, type the IP address and port number of the device that is actually running your CoDeSys project.

   b) In the Runtime Address box, type the specific address of your CoDeSys project on the device. This address can be found when you open the CoDeSys project in the CoDeSys development application. A typical address is 0194.

9. If you selected CoDeSys 2.x, configure the corresponding settings.

![CoDeSys Tag Integration dialog for CoDeSys 2.x](image)

   a) In the PLC IP Address and Port boxes, type the IP address and port number of the device that is actually running your CoDeSys project.

   b) If the device is located behind a network gateway, select Gateway IP and then type the IP address of the gateway.

   c) From the Level list, select the network protocol to communicate with the device. In most cases, you should select L4.

10. If you also want to browse the CoDeSys tags when the PLC is not running, then enable offline browsing and locate the symbol file that you copied to your IWS project folder.

   a) Select Symbol file browsing (offline).

   b) Click Browse, and then locate the symbol file.

11. Click OK to finish the configuration.
If the configuration is successful, then the CoDeSys tags will be immediately available in the Object Finder.

**CONFIGURE YOUR CODESYS 3.X PROJECT FOR TAG INTEGRATION**

This task describes how to configure a CoDeSys 3.x project to communicate with external programs, such as InduSoft Web Studio, during runtime. It is a prerequisite to integrating CoDeSys tags into your IWS project.

By default, the CoDeSys 3.x project development software does not generate a symbol file when you rebuild your CoDeSys project. You must add a Symbol Configuration object to your CoDeSys project and then configure the object to include the variables that you want to export to the symbol file.

1. Open your CoDeSys 3.x project.

   ![Opening a CoDeSys 3.x project](image)

   **Opening a CoDeSys 3.x project**

2. In the project explorer, right-click **Application**, and then click **Add Object** on the shortcut menu. The **Add Object** dialog is displayed.

   ![Add Object dialog](image)

   **Add Object dialog**

3. From the list of objects, select **Symbol configuration**, and then click **Open**. A new Symbol Configuration object is added to your project and it is opened for editing.
4. In the Symbol Configuration object, add the variables you want to communicate with. You need to move them from the **Available variables** list on the left to the **Selected variables** list on the right.

If you do not see your variables in the **Available variables** list, check the following:
- For Local Variables (POU variables), the POU containing them must be called in a Task:
  1. Add a Task Configuration object to the application.
  2. Add a Task to the Task Configuration object.
  3. Add the POU to the Task.
- For Global Variables, at least one of the variables from the Group must be used in at least one POU that is being called by one Task.

5. Close the Symbol Configuration object.
6. On the **Build** menu, click **Rebuild Application**.

Once the CoDeSys project is configured to export the selected variables, the next time you send the project to the PLC, it will include the new symbol file and your IWS project to be able to communicate with it.

You can also copy the CoDeSys symbol file to your IWS project folder, which will allow you to browse the tags when the PLC is not running.

**CONFIGURE YOUR CODESYS 2.X PROJECT FOR TAG INTEGRATION**

This task describes how to configure a CoDeSys 2.x project to communicate with external programs, such as InduSoft Web Studio, during runtime. It is a prerequisite to integrating CoDeSys tags into your IWS project.

The CoDeSys 2.x project development software automatically exports its project database to a symbol file every time you rebuild your CoDeSys project. However, CoDeSys exports the entire database by default, including many system and library variables that IWS cannot import. You must reconfigure your CoDeSys project options to export only the POUs and Global Variables and then rebuild your CoDeSys project to generate a fresh symbol file.
1. Open your CoDeSys 2.x project.

2. In the project explorer on the right, click the Resources tab.

3. In the list of resources, double-click Target Settings. The Target Settings dialog is displayed.

4. Click the General tab.

5. Make sure that Download symbol file is selected.

6. Click OK to close the Target Settings dialog.

7. On the Project menu, click Options. The Options dialog is displayed.
8. In the **Category** list, click **Symbol configuration**.

![Symbol configuration dialog](image1)

**Selecting "Symbol configuration"**

9. Select **Dump symbol entries** and **Dump XML symbol table**.

10. Click **Configure symbol file**. The **Set object attributes** dialog is displayed.

11. For the sake of expediency, you should first disable the export of all objects and then reenable only the objects that you want to export to IWS — typically, the POUs and Global Variables. Select all of the objects in the tree and then clear all options for them at the bottom of the dialog.

![Clearing the options for all objects](image2)

**Clearing the options for all objects**

**Tip**: You may need to select **Export variables of object** in order to activate the other checkboxes before clearing them.

12. Reselect only the POUs and Global Variables that you want to export to IWS. Do not select libraries. With the objects selected, select all of the options at the bottom of the dialog.

13. Click **OK** to close the **Set object attributes** dialog, and then click **OK** again to close the **Options** dialog.
14. On the **Project** menu, click **Clean All**.

15. On the **Project** menu, click **Rebuild All**.
   The CoDeSys development software will rebuild the project, generating a symbol file that contains only the selected POUs and Global Variables.

Once the CoDeSys project is configured to export the selected variables, the next time you send the project to the PLC, it will include the new symbol file and your IWS project to be able to communicate with it.

You can also copy the CoDeSys symbol file to your IWS project folder, which will allow you to browse the tags when the PLC is not running.

**Integrate tags from RSLogix 5000 Family**

This task describes how to add a Rockwell Automation RSLogix 5000 PLC as a tag integration source in your IWS project.

Before you begin this task, you should do the following:

- Review the manufacturer’s documentation for your RSLogix 5000 PLC;
- Use the RSLogix 5000 programming software to export a new symbol file (*.L5K) from your PLC program;
- Rebuild your PLC program, and then download it to the PLC; and
- Make sure the PLC is running and available on your network, and note its network address.

To add an RSLogix 5000 PLC as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
   The **Project Settings** dialog is displayed, with the **Communication** tab selected.

2. In the **Tag Integration** area, click **Add**.
   The **Tag Integration Source** dialog box is displayed.

3. In the **Type** list, click **Integrated**.
4. In the **Provider** list, click **RSLogix 5000 Family**.

5. In the **Name** box, type an appropriate name for this tag integration source. Note that the name will be used as a prefix on names of the integrated tags.

6. Click **Add**. The **RSLogix Tag Integration** dialog box is displayed.

7. If you want to communicate directly with the PLC using the RSLogix communication driver, then do the following:
   a) In the **Connection Type** list, click **Driver**.
   b) In the **PLC IP Address** box, type the address of the PLC.
   c) In the **CPU Slot Number** box, type the number of the PLC slot in which the CPU module is installed. The default is slot 0.

8. If you want to communicate with the PLC through an OPC server, then do the following:
   a) In the **Connection Type** list, click **OPC**.
   b) In the **OPC Server** list, select the type of server. At this time, only two OPC servers support RSLogix 5000 PLCs: Software Toolbox and Rockwell Automation's own RSLinx.
   c) In the **CPU Slot Number** box, type the number of the PLC slot in which the CPU module is installed. The default is slot 0.
   d) In the **Remote Server** box, type the address of the OPC server.

9. Select the symbol file that you exported from your PLC program.
   a) Click **Browse**. A standard **Open** dialog box is displayed.
   b) Locate and select the symbol file (***.L5K**).
      In most cases, the file should be saved in the Config sub-folder of your IWS project folder at: `[...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\`
   c) Click **Open**. For more information, see Export symbol file for RSLogix 5000 Family on page 390.

10. Click **OK** to finish the configuration and add the source.

    If the source is added successfully, then the RSLogix 5000 PLC tags will be immediately available in the **Object Finder**.

    **Note:** Some complex tag structures, such as arrays of nested structures and aliases of members of modules, are not supported.
**EXPORT SYMBOL FILE FOR RSLOGIX 5000 FAMILY**

Export a symbol file from your RSLogix 5000 PLC program in order to be able to add the PLC as a tag integration source.

The symbol file is exported from the RSLogix 5000 development software. It contains information about all of the tags used in your PLC program, and the information can be imported into your IWS project.

Note: Tag integration supports up to RSLogix 5000 V20, including Add-On Instructions (AOI).

To export the file:

1. Run the RSLogix 5000 PLC programming software, and then open your PLC program.
2. On the File menu, click Save As.
   A Save As dialog box is displayed.
3. Use the file browser to locate where you want to save the file.
   In most cases, you should save it in the Config sub-folder of your IWS project folder at: [...]\Documents \InduSoft Web Studio v7.1 Projects\projectname\Config\.
4. In the File name box, type a name for the file.
5. In the Save as type list, click RSLogix 5000 Import/Export File (*.L5K).
6. Click Save.

The file is saved at the specified location.

**Integrate tags from Allen-Bradley PLC5, SLC500**

This task describes how to add an Allen-Bradley PLC2, PLC5, or SLC500 as a tag integration source in your IWS project.

This tag integration is based on the ABTCP driver, which communicates with Allen-Bradley devices (and others) using the DF1 protocol.

Before you begin this task, you should do the following:
- Review the manufacturer's documentation for your Allen-Bradley device;
- Read the ABTCP driver documentation (on the Help tab of the ribbon, in the Documentation group, click Communication Drivers, and then select ABTCP);
- Familiarize yourself with how memory areas — that is, groups of memory addresses — are configured on Allen-Bradley devices; and
- Make sure the source device is running and available on your network, and note its network address.

To add an Allen-Bradley device as a tag integration source:

1. On the Project tab of the ribbon, in the Settings group, click Communication.
The *Project Settings* dialog is displayed, with the **Communication** tab selected.

![Communication tab of the Project Settings dialog](image)

2. In the **Tag Integration** area, click **Add**.
   The *Tag Integration Source* dialog box is displayed.

3. In the **Type** list, click **Integrated**.

4. In the **Provider** list, click **Allen-Bradley PLC5, SLC500**.

5. In the **Name** box, type an appropriate name for this tag integration source.
   Note that the name will be used as a prefix on names of the integrated tags.

6. Click **Add**.
   The *ABTCP Provider* dialog box is displayed.

![ABTCP Provider dialog box](image)

7. In the **PLC IP Address** box, type the network address of the Allen-Bradley device.
   By default, port 2222 is assumed. If the device uses another port, then include it in the address.

8. In the **PLC Family** list, click the device family.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC2</td>
<td>Allen-Bradley PLC-2 Control System</td>
</tr>
<tr>
<td>PLC5</td>
<td>Allen-Bradley PLC-5 Control System</td>
</tr>
</tbody>
</table>
9. Check the default memory areas. (The defaults are the same for all device families.) If you need to add another memory area to match how you have configured your device, then do the following:
   a) Click Add. The Memory Area dialog box is displayed.

   ![Memory Area dialog box](image)

   Memory Area dialog box

   b) In the Type list, click the memory address type.

   c) In the Slot Number / File Number box, type the number of the slot (for O, I, S) or file (for B, N, T, C, R, F, ST).

   ![Note:](image)

   Note: If the family is PLC2 or PLC5 and the type is O, I, or S, then the slot number is automatically 0.

   d) In the Size box, type the size (in bits) of the memory area.

10. Click OK to close the Memory Area dialog box, and then repeat the previous step as needed.

11. Click OK to finish the configuration and add the source.

If the source is added successfully, then the Allen-Bradley PLC tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 398.

**Integrate tags from Schneider Unity Modbus**

This task describes how to add a Schneider Modicon M340 PAC or Modicon Premium PAC as a tag integration source in your IWS project.

This tag integration is based on the SCHNE driver, which communicates with Schneider Modicon devices using the Modbus protocol over Ethernet.

Before you begin this task, you should do the following:

- Review the manufacturer’s documentation for your Schneider Modicon device and the Schneider Unity Pro software;
- Use the Schneider Unity Pro software to export an I/O configuration file (*.XY) from your PLC program;
- Read the SCHNE driver documentation (on the Help tab of the ribbon, in the Documentation group, click Communication Drivers, and then select SCHNE); and
- Make sure the source device is running and available on your network, and note its network address.

To add a Schneider Modicon device as a tag integration source:

1. On the Project tab of the ribbon, in the Settings group, click Communication.
The *Project Settings* dialog is displayed, with the *Communication* tab selected.

2. In the **Tag Integration** area, click **Add**. The **Tag Integration Source** dialog box is displayed.
3. In the **Type** list, click **Integrated**.
4. In the **Provider** list, click **Schneider Unity Modbus**.
5. In the **Name** box, type an appropriate name for this tag integration source. Note that the name will be used as a prefix on names of the integrated tags.
6. Click **Add**. The **Schneider Unity Tag Integration** dialog box is displayed.

7. In the **Platform** list, click the platform of the source device.
   
   **Option** | Description
   --|---
   M340 | Schneider Modicon M340 PAC
   Premium | Schneider Modicon Premium PAC

8. In the **PLC IP** box, type the network address of the source device.
9. In the **Port** box, type the port number of the source device if it is different from the default port 502.
10. In the **Slave ID** box, type the Modbus slave ID of the source device if it is different from the default ID 1.
11. Select the I/O configuration file that you exported from your PLC program.
   a) Click **Browse**. A standard *Open* dialog box is displayed.
b) Locate and select the I/O configuration file (*.XSY).

In most cases, the file should be saved in the Config sub-folder of your IWS project folder at:

```
 [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config
```

c) Click Open.

For more information, see Export I/O configuration file for Schneider Unity Modbus on page 394.

The selected file is displayed in the XSY File box.

12. If you want to get only tags that are flagged as HMI variables, then select HMI Only.

13. Click OK to finish the configuration and add the source.

If the source is added successfully, then the Schneider Modicon device tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 398.

**EXPORT I/O CONFIGURATION FILE FOR SCHNEIDER UNITY MODBUS**

Export an I/O configuration file from your Schneider Modicon PLC program in order to be able to add the PLC as a tag integration source.

The I/O configuration file (*.XSY) is exported from the Schneider Unity Pro development software. It contains information about all of the tags used in your PLC program, and the information can be imported into your IWS project.

To export the file:

1. Run the Schneider Unity Pro development software, and then open your PLC program.

2. In the program browser, right-click the Variables & FB Instances file, and then click Export on the shortcut menu.

   A standard Export dialog is displayed.

3. Use the file browser to locate where you want to save the file.

   In most cases, you should save it in the Config sub-folder of your IWS project folder at:

   ```
   [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config
   ```

4. In the File name box, type a name for the file.

5. Click Export.

The file is saved at the specified location.

**Integrate tags from AutomationDirect Koyo**

This task describes how to add a Koyo DirectLOGIC PLC (supplied by AutomationDirect) as a tag integration source in your IWS project.

This tag integration is based on the KOYO driver, which communicates with Koyo DirectLOGIC PLCs over both serial communication and Ethernet.

Before you begin this task, you should do the following:

- Review the manufacturer’s documentation for your Koyo DirectLOGIC PLC;
- Download and install the DirectSOFT programming software from AutomationDirect, and then use it to export your PLC program as a CSV file;
- Read the KOYO driver documentation (on the Help tab of the ribbon, in the Documentation group, click Communication Drivers, and then select KOYO); and
- Make sure the source device is running and available on your network, and note its network address.

To add Koyo DirectLOGIC PLC as a tag integration source:

1. On the Project tab of the ribbon, in the Settings group, click Communication.
The Project Settings dialog is displayed, with the Communication tab selected.

2. In the Tag Integration area, click Add. The Tag Integration Source dialog box is displayed.
3. In the Type list, click Integrated.
4. In the Provider list, click AutomationDirect Koyo.
5. In the Name box, type an appropriate name for this tag integration source. Note that the name will be used as a prefix on names of the integrated tags.
6. Click Add. The Koyo Tag Integration dialog box is displayed.

7. In the Communication Mode list, click the platform of the source device.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>Direct serial communication</td>
</tr>
<tr>
<td>EthernetName</td>
<td>Ethernet communication by Module Name, when a broadcast router is used</td>
</tr>
<tr>
<td>EthernetIP</td>
<td>Ethernet communication by IP address</td>
</tr>
<tr>
<td>EthernetID</td>
<td>Ethernet communication by Module ID, when a broadcast router is used</td>
</tr>
</tbody>
</table>

8. In the Device box, type the name, address, or ID of the source device. The format of Device depends on what you selected for Communication Mode.
9. Select the CSV file that you exported from your PLC program.
Communication with Other Devices

a) Click **Browse**.
   A standard **Open** dialog box is displayed.

b) Locate and select the CSV file (* .CSV).
   In most cases, the file should be saved in the Config sub-folder of your IWS project folder at:
   [...] \Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\ 

c) Click **Open**.
   For more information, see Export CSV file for AutomationDirect Koyo on page 396.
   The selected file is displayed in the **CSV File** box.

10. Click **OK** to finish the configuration and add the source.

If the source is added successfully, then the Koyo DirectLOGIC PLC tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 398.

**EXPORT CSV FILE FOR AUTOMATIONDIRECT KOYO**

Export a CSV file from your Koyo DirectLOGIC PLC program in order to be able to add the PLC as a tag integration source.

The CSV file (* .CSV) is exported from the DirectSOFT programming software. It contains information about all of the elements in your PLC program, and the information can be imported into your IWS project.

**Tip:** You can download the DirectSOFT programming software from AutomationDirect at:
   support.automationdirect.com/products/directsoft.html

To export the file:

1. Run the DirectSOFT programming software, and then open your PLC program.
2. Click **File**, and then on the File menu, click **Export > Element Documentation**.
   The **Export Documentation** dialog box is displayed.
3. Use the file browser to locate where you want to save the file.
   In most cases, you should save it in the Config sub-folder of your IWS project folder at: [...]
   Documents \InduSoft Web Studio v7.1 Projects\projectname\Config\ 
4. In the **File name** box, type a name for the file.
5. Click **Save**.

The file is saved at the specified location.

**Integrate tags from AutomationDirect PAC 3000**

This task describes how to add a Productivity3000 PAC (supplied by AutomationDirect) as a tag integration source in your IWS project.

This tag integration is based on the PAC3K driver, which communicates with Productivity3000 PACs over Ethernet using the Modbus Extended protocol.

Before you begin this task, you should do the following:

- Review the manufacturer's documentation for your Productivity3000 PAC;
- Download and install the Productivity Suite Programming Software from AutomationDirect, and use it to export your PLC program as a CSV file;
- Read the PAC3K driver documentation (on the Help tab of the ribbon, in the Documentation group, click Communication Drivers, and then select PAC3K); and
- Make sure the source device is running and available on your network, and note its network address.

To add Productivity3000 PAC as a tag integration source:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Communication**.
The *Project Settings* dialog is displayed, with the *Communication* tab selected.

![Communication tab of the Project Settings dialog](image1)

2. In the *Tag Integration* area, click *Add*. The *Tag Integration Source* dialog box is displayed.
3. In the *Type* list, click *Integrated*.
4. In the *Provider* list, click *AutomationDirect PAC 3000*.
5. In the *Name* box, type an appropriate name for this tag integration source. Note that the name will be used as a prefix on names of the integrated tags.
6. Click *Add*. The *PAC 3000 Tag Integration* dialog box is displayed.

![PAC 3000 Tag Integration dialog box](image2)

7. In the *PLC Station* box, type the IP address of the source device.
8. Select the CSV file that you exported from your PAC program.
   a) Click *Browse*. A standard *Open* dialog box is displayed.
   b) Locate and select the CSV file (*.CSV*).
      In most cases, the file should be saved in the Config sub-folder of your IWS project folder at: ` [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config`.
   c) Click *Open*.
     For more information, see Export CSV file for AutomationDirect PAC 3000 on page 398. The selected file is displayed in the *CSV File* box.
9. Click *OK* to finish the configuration and add the source.

If the source is added successfully, then the Productivity3000 PAC tags will be immediately available in the Object Finder. For more information, see Use the Object Finder to select integrated tags on page 398.
Communication with Other Devices

EXPORT CSV FILE FOR AUTOMATIONDIRECT PAC 3000

Export a CSV file from your Productivity3000 PAC program in order to be able to add the PAC as a tag integration source.

The CSV file (*.CSV) is exported from the Productivity Suite Programming Software. It contains information about all of the tags in your PAC program, and the information can be imported into your IWS project.

Tip: You can download the Productivity Suite Programming Software from AutomationDirect at: support.automationdirect.com/products/p3000.html

To export the file:
1. Run the Productivity Suite Programming Software, and then open your PAC program (*.ADPRO).
2. Click File, and then on the File menu, click Export > Tags.
   The Export Tag Database dialog box is displayed.
3. Click Browse, and then use the file browser to locate where you want to save the file.
   In most cases, you should save it in the Config sub-folder of your IWS project folder at: [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\n4. In the File name box, type a name for the file.
5. Click Select.
   The file browser is closed, and the selected location is displayed in the To File box.
6. Select Include I/O Tags.
7. Click Export.
   The file is saved at the specified location.

Use the Object Finder to select integrated tags

After you have added a tag integration source to your project, the tags from that source can be viewed and selected in the Object Finder like any other project tags.

Tip: To access the Object Finder, simply double-click in any box or field where you would normally configure a project tag.

Integrated tags are listed in the Object Finder tree-view under Devices, with each device being a tag integration source that you added. The names of the devices are those that you specified when you added them.

Browsing integrated tags in the Object Finder

Devices are shown with three possible icons:
The folder structure for each device is determined by the tag integration source, and the individual tags are listed to the right.

To refresh the devices and tag lists — either by rereading the symbol files or by trying to reconnect to the devices — click Refresh Devices on the right side of the window. All tag integration sources will be refreshed at the same time.

Tags listed in gray have not yet been imported into your project. When you select and import a tag — so that it is added to your project’s Shared Database folder — it is listed in black. You can select Show only imported tags to filter the list, in the event that you only need to get a tag that you have already imported and you do not want to scroll through the entire list to find it.

To quickly import multiple tags and add them all to the Shared Database folder, select them and then click Import Selected Tags. Importing tags in this way, however, will not configure the tags in a box or field. That can only be done one tag at a time, by selecting a tag and then clicking OK.

Changes in the tag integration might break tag references in your project. To find broken tag references, verify your project.

Note: IWS does not support pointer variables from devices. They will be shown in the Object Finder with the rest of the device tags, but they will be marked with a red X icon to indicate that they cannot be used.

**How integrated tags may be renamed in your project**

When IWS integrates tags from third-party devices and software, it cannot directly transcribe the tag names. Some changes are made to improve tag management and to adhere to the local tag name syntax.

Tip: The full, original name of an integrated tag can always be retrieved by referencing the DeviceTag property on the tag. For example, `tagname->DeviceTag`.

**Inserting the tag prefix**

First of all, since your project may connect to multiple devices that have the same control program and device tags, IWS will automatically insert the tag prefix that you specified when you added the tag integration source. For example, for an integrated tag named...

```
MyDeviceTAG[1].ClassMember
```

...the corresponding IWS project tag will be named...

```
tagprefix.MyDeviceTAG[1].ClassMember
```

This allows you to differentiate between similar tags from different sources.

**Multidimensional arrays**

IWS does not support multidimensional arrays, so for integrated tags that have more than one array index, each index after the first will be represented with `_Index_`. For example, for an integrated tag named...

```
MyDeviceTAG[1][2][3].ClassMember
```
...the corresponding IWS project tag will be named...

\texttt{tagprefix.MyDeviceTAG[1]_2\_3\_.ClassMember}

\textbf{Nested classes}

IWS does not support nested classes, so for integrated tags that have more than one class member, each class member after the first will be represented with \texttt{_.ClassMember}. For example, for an integrated tag named...

\texttt{MyDeviceTAG.ClassMember.ClassMember2}

...the corresponding IWS project tag will be named...

\texttt{tagprefix.MyDeviceTAG.ClassMemberOfClassMember2}
Configuring direct communication with a remote device

A communication driver is a DLL containing specific information about the remote equipment and implements the communication protocol. Drivers for dozens of common and not-so-common devices are installed with InduSoft Web Studio.

(InduSoft also provides a toolkit to develop new communication drivers. For more information, please contact Customer Support.)

The Drivers task/worksheet allows you to define the communication interface (or interfaces) between the project and remote equipment; such as a PLC, a single-loop, and transmitters.

**Note:** Consult the Help menu for a description of the functions and characteristics that are standard for all drivers. When developing a project, you can also refer to the specific documentation provided with each communication driver. This documentation is usually located in the DRV directory.

To configure a communication driver, you must specify the interface parameters (for example, the station address and the baud rate), specify the equipment addresses, and then link them to project tags.

Use one of the following methods to add or remove a driver:

- On the Insert tab of the ribbon, in the Communication group, click Add/Remove Driver; or
- Right-click the Drivers folder in the Project Explorer, and then click Add/Remove drivers on the shortcut menu.

Both methods open a Communication Drivers dialog, which displays a list of available drivers.

**Communication Drivers dialog**

Use the parameters on this dialog, as follows:

- **Available Drivers** field: Lists all available drivers and a brief description of each.
- **Help** button: Click to open the Help menu, which contains detailed configuration instructions for the driver currently highlighted in the Available Drivers field.
- **Select** button: Click to select the driver currently highlighted in the Available Drivers field.
- **Selected Drivers** field: Lists all selected drivers and their descriptions (if available).
- **Remove** button: Click to remove a driver currently highlighted in the Selected Drivers field.

When you click OK in the Communications Driver dialog, you create a subfolder for the selected driver(s) in the Drivers folder located on the Comm tab.
You can right-click on a driver subfolder to access the **Settings** option, which opens the *Communications Parameters* dialog.

![Sample Communications Parameters dialog](image)

Use the parameters on this dialog, as follows:

- **Serial Encapsulation** field: Enables serial drivers to communicate with modem, TCP/IP or UDP connections. This setting is supported only for serial drivers developed with the UNICOMM library, which includes most of the serial drivers available in the product.

  **Caution:** The **Modem** option is not supported for Pocket PC v3.00 or older.

- **COM** field: Click to select a serial communication port.
- **Baud Rate**, **Data Bits**, **Stop Bits**, and **Parity** fields: Click to select parameters for a serial port configuration.
- **Long1**, **Long2**, **String1**, and **String2** fields: These fields are driver custom settings. In the example above, the driver uses **Long1** to set up the error detection method and **String1** to define the PLC family type.
- **Advanced** button: Click to open the *Advanced settings* dialog. Use this dialog to change the default driver parameters.

![Advanced Settings dialog](image)

Specify or change the default driver parameters as follows:

- **Timeout (ms)** area:
  - **Start message** field: Specify the timeout for the message start.
  - **End message** field: Specify the timeout for the message end.
Communication with Other Devices

- **Interval between char** field: Specify the timeout between each character.
- **Wait CTS** field: Specify the timeout for the Clear to Send wait.

**Handshake area:**
- **Control RTS** drop-down list: Specify whether to use the "Request to Send" control.
- **Verify CTS** drop-down list: Specify whether to use the "Clear to Send" type of verification.
- **Disable DTR** checkbox: Click (enable) this box to disable the DTR function (the driver will not set the DTR signal before starting the communication).
- **Enable IR** checkbox (only available on Windows Embedded target systems): Click (enable) this box to enable the serial driver to use an Infrared interface (COM2 port) instead of a standard serial port to communicate with the device (such as the PLC, I/O, hand-held computers, and so forth).

**Protocol area:**
- **Station** field: Some slave drivers such as the Modbus Slave (MODSL) require a slave network address. Use this field to specify the slave address.
- **Retries** field: Type a numeric value to specify how many times the driver will attempt to execute the same communication command before considering a communication error for this command.

**Buffers length (bytes) area:**
- **Tx Buffer** field: Specify the transmission buffer length (in bytes).
- **Rx Buffer** field: Specify the reception buffer length (in bytes).

**Simultaneous Requests area** (available only on selected drivers):
- **Maximum** field: Specify the maximum number of requests that may be sent simultaneously to all connected devices.
- **Maximum per station** field: Specify the maximum number of requests that may be sent simultaneously to a single device.

**Note:** The maximum number of simultaneous requests depends on the device and protocol specifications. Please consult the device manufacturer's documentation.

The development application provides two interfaces, which you can use to configure the driver (associating project tags to device addresses):

- **MAIN DRIVER SHEET:** Provides the easiest method for configuring communication between project tags and device addresses. This interface allows you to automatically group tags to provide the best performance during runtime. You cannot use this interface to control the time needed to scan a group of tags individually.

- **STANDARD DRIVER SHEETS:** Allows you to control the time needed to scan a group of tags individually. You can use both sheets at the same time.

**Using TCP/IP and UDP Encapsulation**

Most of the serial drivers allow the use of TCP/IP or UDP/IP encapsulation. The encapsulation mode has been designed to provide communication with serial devices connected to terminal servers on your ethernet or wireless networks. A terminal server can be seen as a virtual serial port. It converts TCP/IP or UDP/IP messages on your Ethernet or Wireless network to serial data. Once the message has been converted to a
serial form, you can connect standard devices that support serial communications to the terminal server. The following diagram provides one example of applying this solution:

TCP/IP Encapsulation

You can enable the encapsulation by following the steps below:
1. Right-click on the driver’s folder, and then choose Settings from the shortcut menu.
   This will give you access to the communication parameters.
2. In the Serial Encapsulation field, select TCP/IP or UDP/IP:

   The following fields are available:
   - **IP Address** field: Specify the IP Address for the Terminal Server. This field accepts tags in curly brackets.
   - **Port Number** field: Enter the TCP/IP or UDP/IP port number.
   - **Status Tag** field: This field is available only when using TCP/IP. The tag on this field receives the value 1 when the TCP/IP connection is established; otherwise, it receives 0.
   - **Server Mode** field: The TCP/IP encapsulation allows the Server Mode, making the remote client responsible for establishing the connection to enable the communication.

Using Modem Connections

Most of the serial drivers allow the use of modem connections. The modem connection has been designed to enable communications with remote serial devices connected through a phone line. The following diagram provides one example of applying this solution:

Modem Connection

You can enable the modem connection by following the steps below:
1. Right-click on the driver's folder, and then choose **Settings** from the shortcut menu. This will give you access to the communication parameters.

2. In the **Serial Encapsulation** menu, select **Modem**:

![Modem Settings](image)

**Caution:** The **Modem** option is not supported for Pocket PC v3.00 or older.

The following fields are available:

- **Name** drop-down list: Select the modem that the driver will use to establish the connection. If you do not know the modem name, use the Auto Detect option. The **Auto Detect 1** will use the first modem available, **Auto Detect 2** will use the second, **Auto Detect 3** will use the third, and **Auto Detect 4** will use the fourth.

- **Phone** field: Enter a phone number that the driver will use to connect to the remote device. This field accepts tags between curly brackets.

- **Settings** button: Click on this button to configure the modem settings. The window that displays when you click on this button depends on the operating system that you are using and on the modem type.

**Caution:** The settings configured by clicking on this button are not saved with your project. The information is saved on the operating system registry, and they are valid only in the computer that you are interacting with. If you install your project on another computer, you will have to reconfigure these settings.

- **Connection** button: Click to open the **Connection Control** window. The default connection settings should suffice for most of the projects. However, you can take full control over the connection, and also enable incoming calls, by clicking on this button.

![Connection Control](image)

**Connection Control dialog**

- **Dial out trigger** field: When the value of the tag configured in this field changes, the driver will try to connect to the remote device. If the connection has already been established, the command is ignored. You do not have to use this field if you are using Auto Connect.

- **Hang up trigger** field: When the value of the tag configured in this field changes, the driver will disconnect from the remote device. If the device is disconnected the command is ignored. You do not have to use this field if you are using Disconnect call if idle for more than.

- **Auto Connect** field: When this option is enabled, the driver will try to connect to the remote device before sending any information. If the connection fails, the next attempt will be made after the Retry Interval has expired.
• **Disconnect call if idle for more than** field: When this option is checked, the driver will automatically disconnect from the remote device if no communication is performed after the time you specified.

• **Enable incoming calls** field: Check this option if you want to enable the driver to receive calls from the remote device. You can use the Hang up trigger to drop the call once it has been established. Notice that one driver can use both incoming calls and outgoing calls.

• **Status area**
  - **Code** field: Enter with a tag that will receive one of the following codes when the driver is running:
    - 0 = Disconnected
    - 1 = Connected
    - 2 = Dialing
    - 3 = Dropping
    - 4 = Closing Line
  - **Description** field: Enter with a tag that will receive a complete description of the current status. The description is associated with the **Code** field; however, it brings some additional information about the current status.

**Serial Encapsulation Tests**

Most of the serial drivers should work with every serial encapsulation mode. However, most of the drivers were developed before the encapsulation modes had been created. The following table lists the drivers fully tested with certain encapsulation modes; if the driver that you intend to use is not listed and you are unsure whether it will work, please contact your distributor.

<table>
<thead>
<tr>
<th>Driver</th>
<th>Modem</th>
<th>TCP/IP</th>
<th>UDP/IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODSL</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ABKE</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MODBU</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>OMETH</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X = Item has been tested

**Main Driver Sheet**

The development application automatically inserts the MAIN DRIVER SHEET into the driver folder as soon as you add the driver to your project.

**Note:** The MAIN DRIVER SHEET is not available for all drivers.

To configure the MAIN DRIVER SHEET, right-click on the icon, and select **Open** from the pop-up or just double-click on the icon.
The MAIN DRIVER SHEET dialog displays (see the following figure).

![Sample MAIN DRIVER SHEET](image)

The MAIN DRIVER SHEET worksheet is divided into two areas:

- **Header area** (top section), contains parameters that affect the all tags configured in the Body area of this worksheet; and
- **Body area** (bottom section), where you define the relationship between tags in the project and their field equipment address.

Use the **Header** area parameters as follows:

- **Description** field: Type a description of the MAIN DRIVER SHEET for documentation purposes.
- **Disable** field: Type a tag or an expression to enable and disable the communication of each MAIN DRIVER SHEET on the fly.
  - Type a value (or expression result) that is greater than zero to disable the MAIN DRIVER SHEET.
  - Type a zero (or leave this field blank) to enable the MAIN DRIVER SHEET.
- **Read Completed** field: Type in a tag and the communication driver toggles the tag when it completes a read command.
- **Read Status** field: Type in a tag, which is updated with the status of the last read command.
- **Write Completed** field: Type in a tag and the communication driver toggles the tag when it completes a write command.
- **Write Status** field: Type in a tag, which is updated with the status of the last write command.
- **Min and Max** checkbox: Click (check) to specify minimum and maximum values for data from the field equipment.
- **Min and Max** fields (become active only when you enable the Min and Max checkbox): Type a range of values, which can be converted into an engineering format.

The project uses these fields to determine a minimum/maximum range of values for data from the field equipment. The scaling is done automatically. You must configure the engineering range using the Min and Max parameters on the Tag Properties dialog. This range affects all tags in the worksheet, except those with customized Min and Max values, as specified in the Body area of the driver sheet (Min and Max columns).

Use the **Body** area parameters as follows:

- **Tag Name** field: Type the name of a project tag to be used by the communication driver.
- **Station** field: Type the number of the equipment station within the network. The syntax in this field varies with each communication driver. Refer to the appropriate driver’s documentation for further information.
Tip: For some drivers, if you’ve configured the driver to do serial encapsulation via TCP/IP or UDP/IP, then the station may be specified using the following format:

\[ \text{IP	extunderscore address:port	extunderscore number\textbar station} \]

For example:

\[ 10.169.25.18:1234\textbar \text{Station5} \]

To see if this feature is supported on your selected driver, refer to the driver’s documentation.

Tip: You can configure a tag name (string) between curly brackets in this field. In this case, the tag value will be the Station used by the driver. Therefore, you can change the station dynamically during runtime.

Configuring a string tag between curly brackets in the Station field of the Main Driver Sheet (MDS) is especially useful when configuring projects for redundant PLCs. Changing the value of the tag configured in the Station field, you can switch automatically from one PLC to the other in case of a failure of the primary PLC (hot/Stand-by).

- **I/O Address** field: Type the address of the field equipment related to the project tag. The syntax in this field varies with each communication driver. Refer to the appropriate driver’s documentation for further information.

- **Action** field: Specify the communication direction, using one of the following options:
  - **Read** (the project continuously reads the address from the field device and updates the Tag value.)
  - **Write** (the project writes the tag value to the field device when the tag value changes.)
  - **Read+Write** (Combines the procedures of both the Read and Write parameters.)

- **Scan** field: Specify the condition under which the tag value is read from the remote device or server and then updated in the project database, using one of the following options:
  - **Always** means the tag is read and updated during every scan of the communication worksheet, regardless of whether the tag is used in any other project screens, scripts, or worksheets.
    This option is recommended for tags that must be continuously monitored in the background, such as tags that trigger alarms, tags used in recipes, tags that are recorded in the historical database, and so on.
  - **Screen** means the tag is read and updated only if it is being used in at least one open project screen, either locally or on another client station.
    This option is recommended for tags that are used in screen objects, because the project may not need to update tags that are not being visualized anywhere. Selecting this option can improve project performance.
  - **Auto** means the project will automatically choose either Always or Screen, depending on where the tag is used in your project. If the tag is only used in a screen object on a project screen, then the scan will default to Screen. But if the tag is configured in any other interface (e.g., Script, Math, Alarm, Trend, Recipe, Report, Scheduler), then the scan will switch to Always and remain there until the project is stopped.

If you are not sure of which option to select, select Always. This will guarantee the tag is read and updated.

- **Div** field: Specify the division constant when scale adjustment is required. This value is a division factor in a read operation and a multiplication factor in a write operation. Do not use this field if you are already using Min or Max in the configuration body.

- **Add** field: Specify the addition constant when scale adjustment is required. This value is an addition factor in a read operation and a subtraction factor in a write operation. Do not use this field if you are already using Min or Max in the configuration body.
Note: The Main Driver Sheet can have up to 32767 rows. If you need to configure more than 32767 communication addresses, then either configure additional Standard Driver Sheets or create additional instances of the driver.

Tip: By default, the project will scan the communication worksheet every 600 milliseconds, which is the rate at which the system tag BlinkSlow toggles. To adjust the rate, manually edit the project file (i.e., projectname.APP) to add the following entry:

```plaintext
[Options]
MainDrvAlwaysTrigger=tagname
```

tagname can be either another system tag (e.g., BlinkFast, Second, Minute) or a tag that you have created. Whenever the value of the tag changes, the worksheet will be scanned and the tags will be read.

Standard Driver Sheets

In addition to the unique MAIN DRIVER SHEET that is available for each driver, you can create several STANDARD DRIVER SHEETS for each driver. The STANDARD DRIVER SHEETS provide additional fields, which you can use to control communication.

Note: You can have a total of 9,999 Standard Driver Sheets for all drivers in your project.

To open a STANDARD DRIVER SHEET, right-click on a driver subfolder and select Insert from the resulting popup (see the following figure).

![Sample STANDARD DRIVER SHEET](image)

The STANDARD DRIVER SHEET dialog is divided into two areas:

- **Header area** (top section), contains parameters that affect the all tags configured in the Body area of this worksheet
- **Body area** (bottom section), where you define the relationship between tags in the project and their field equipment address.

Use the **Header area** parameters as follows:

- **Description** field: Type a description of the STANDARD DRIVER SHEET for documentation purposes.
• **Increase Priority** checkbox: Click (check) to keep the reading and writing commands for this sheet on the top of the communication queue whenever they are triggered.

**Caution:** You must give special attention to this worksheet when you enable the **Increase Priority** option. If the worksheet keeps triggering communication commands, the project may never be able to execute the other driver sheets.

• **Read Trigger** field: Type a tag that triggers the project to read the worksheet automatically when you change this tag's value.

• **Enable Read when Idle** field: Type a tag or constant value. Use a tag (or constant) value greater than zero, to enable reading from the equipment.

**Caution:** If you use a constant value (other than zero), be sure that your project requires a continuous reading because this value places a reading request in every communication scan.

• **Read Completed** field: Type in a tag and the communication driver toggles the tag when it completes a **read** command.

• **Read Status** field: Type in a tag and the communication driver updates the tag with the status of the last **read** command.

• **Write Trigger** field: Type a tag value to activate a group reading. Whenever you change this tag value, the program writes an equipment worksheet.

• **Enable Write on TagChange** field: Type a tag or constant value (not zero) to enable the communication driver to check the worksheet continuously for changes in the tag value. If a change occurs, the project writes this value to an address in the field equipment.

• **Write Completed** field: Type in a tag and the communication driver toggles the tag in this field when a **write** command completes.

• **Write Status** field: Type in a tag and the communication driver updates the tag with the status of the last **write** command.

• **Station** field: Type the equipment station number within the network. The syntax in this field varies with each communication driver. Refer to the appropriate driver's documentation for further information.

**Tip:** For some drivers, if you've configured the driver to do serial encapsulation via TCP/IP or UDP/IP, then the station may be specified using the following format:

```
IP_address:port_number|station
```

For example:

```
10.169.25.18:1234|Station5
```

To see if this feature is supported on your selected driver, refer to the driver's documentation.

• **Header** field: Specify the data type and/or initial address to be read or written in the equipment. The syntax in this field varies with each communication driver. Refer to the appropriate driver's documentation for further information.

**Note:** You can use text in the **Station** and **Header** fields with tag values using the **text {tag}** syntax.

• **Min** and **Max** checkbox (not labeled): Click (check) to specify the minimum and maximum values for field equipment data.

• **Min** and **Max** fields (become active only when you enable the **Min and Max** checkbox): Type a range of values to be converted into an engineering format. These fields determine the minimum and maximum range of values. These values affect all tags in the worksheet.

For example, Memory holds values from 0 to 4095, which means 0% to 100% in the user interface. So for this example, you must specify 0 to 100 for the min and max tag parameters.
Use the **Body** area parameters as follows:

- **Tag Name** field: Type a tag name for the communication driver to use.
- **Address** field: Type a field equipment address (or address offset) related to the project tag. The syntax in this field varies with each communication driver. Refer to the appropriate driver’s documentation for further information.
- **Div** field: Specify a division constant to use when scale adjustment is required. The project uses this value as a division factor in a read operation and a multiplication factor in a write operation. Do not use this field if you are already using **Min** or **Max** in the configuration body.
- **Add** field: Specify an addition constant to use when scale adjustment is required. The project uses this value as an addition factor in a read operation and a subtraction factor in a write operation. Do not use this field if you are already using **Min** or **Max** in the configuration body.

For read operations:

\[ \text{tag} = \frac{\text{value in the equipment}}{\text{Div}} + \text{Add} \]

For write operations:

\[ \text{value in the equipment} = (\text{tag} - \text{Add}) \times \text{Div} \]

If you leave the cells empty in the **Div** and **Add** fields, this function is ignored.

---

**Note:** Each Standard Driver Sheet can have up to 4096 rows. However, the **Read Trigger**, **Enable Read When Idle**, and **Write Trigger** commands attempt to communicate the entire block of addresses that is configured in the sheet, so if the block of addresses is larger than the maximum block size that is supported by the driver protocol, then you will receive a communication error (e.g., “invalid block size”) during run time. Therefore, the maximum block size imposes a practical limit on the number of rows in the sheet, and that limit varies by driver. For more information, please refer to the driver documentation for your selected driver.

---

**Driver Error Codes**

The **Read Status** and **Write Status** fields return error codes to the tags you specified for those fields. These error codes can be standard or customized for each driver. For a description of customized error codes, consult the documentation for the driver.

The following are standard error codes, which are always negative values:

- 0=OK
- -1=Invalid serial port
- -2=Invalid baud rate
- -3=Invalid number of bits
- -4=Invalid number of stop bits
- -5=Invalid parity
- -6=Invalid irq
- -7=Serial port already in use
- -8=Invalid buffer size
- -9=Memory not enough
- -10=Tx buffer empty
- -11=Tx buffer full
- -12=Rx buffer empty
- -13=Rx buffer full
• -14=Timeout waiting CTS
• -15=Timeout waiting start a message
• -16=Timeout waiting a message to finish
• -17=Timeout between rx char
• -18=Timeout between tx char
• -19=No carrier detected
• -20=No DSR detected
• -21=Could not find a 8250 in address
• -22=Tx line is busy
• -23=User abort
• -24=Function not supported
• -25=Overrun
• -26=Parity
• -27=Overrun and parity
• -28=Framing
• -29=Framing and overrun
• -30=Framing and parity
• -31=Framing, overrun and parity
• -32=Timeout waiting a tx message to finish

The UNICOMM.MSG file in the [...]\InduSoft Web Studio v7.1\Bin directory also contains descriptions of the standard error codes.

The DriverName.MSG file in the [...]\InduSoft Web Studio v7.1\Drv directory contains descriptions of the customized error codes for each driver.
Configuring an OPC Client connection to an OPC Server

The OPC Client task/worksheet is used to communicate with any system that implements the OPC Server protocol.

This task implements the OPC standard as described in the OLE for Process Control Data Access Standard Version 1.0A document, which is available at the OPC Foundation web site.

**Note:** Before using the OPC Client task/worksheet in your project, you must make sure the OPC server software is properly installed and configured on the system to which you want to connect.

To configure a new connection, insert a new OPC Client worksheet on the Comm tab of the Project Explorer.

**Sample OPC Client worksheet**

Use the following parameters in the configuration table for OPC:

- **Description** text box: Type a description of the OPC task for documentation purposes only. (The OPC Client task ignores this information.)

- **Server Identifier**: Type the name of the server you want to connect. If the server is already installed on the computer, you can select the server name from the list.

- **Disable**: Type a tag or a constant with a value other than 0, to disable communication with the OPC server. Specify 0, or leave the field blank to enable communication.

- **Read Update Rate**: Specify how often the server should update this group (in milliseconds). Specify 0 to indicate the server should use the fastest practical rate.

- **Percent Deadband** (valid for analog items only): Specify how much percent change in an item value should cause a notification by the server.

- **Status**: Type the name of a tag to receive the status of the connection. Good status is 1.

- **Remote Server Name**: Node name or IP address of server on node network.

- **Read before writing** checkbox: Check this option to force your project to read the original values of items on the OPC server just before writing new values to the server. The project does this by first buffering the new values to be written and then reading the original values from the server. Only after the project is synchronized with the server are the new values written from the buffer to the server.

- **Read after writing** checkbox: Check this option to force your project to read back the new values of items on the OPC server just after the project has written those values.

**Caution:** The Read before writing and Read after writing options are offered because the OPC Client/Server specification says that the value of an item on the client — in this case, your project —
should not change unless the server sends the change. That way, the client always stays in sync with the server.

Your project, however, may be designed to change those values according to runtime processes or user input. Therefore, the best way to change the values while staying in sync with the server is to make it seem like the changes originate on the server. With both options enabled, the following sequence of events happens on every scan of the OPC worksheet:

1. The new values on the client are buffered.
2. The original values on the server are read to the client — that is, the client is synchronized with the server.
3. The new values are written from the buffer to the server.
4. The new values on the server are read to the client — that is, the client is again synchronized with the server.

At the end of each scan, the values reflect what's happening in your project even though, technically speaking, the project is merely staying in sync with the server.

Both options should be enabled in most projects. In some projects, however, this may cause items to bounce between the original values and the new values. If this is a problem, try moving those items to another OPC worksheet where the Read before writing and Read after writing options are disabled.

- **Accept Tag Name in the Item column** checkbox: When this option is checked, the text configured between curly brackets in the Item field is resolved as a Tag Name (string tag). In this case, the value of this tag is used as the name of the item from the OPC Server, allowing the user to point to different item names during runtime, by changing the value of the tag(s) configured in the OPC Client worksheet (Item column).

  When the Accept Tag Name in the Item column option is unchecked, all characters configured in the Item column are considered part of the Item name (including the curly brackets).

- **Tag Name**: Type the names of tags linked to the server items.

- **Item**: Enter the name of the server’s items. After selecting an OPC Server, you can select items from the Server using the OPC Browser. Right-click in the Item field and select the OPC Browser option.

  **Tip**: You can configure a tag name between curly brackets (e.g., \{TagName\}) in this field, allowing the user to change the item names dynamically, during runtime.

- **Scan** field: Specify the condition under which the tag value is read from the remote device or server and then updated in the project database, using one of the following options:

  - **Always** means the tag is read and updated during every scan of the communication worksheet, regardless of whether the tag is used in any other project screens, scripts, or worksheets.

    This option is recommended for tags that must be continuously monitored in the background, such as tags that trigger alarms, tags used in recipes, tags that are recorded in the historical database, and so on.

  - **Screen** means the tag is read and updated only if it is being used in at least one open project screen, either locally or on another client station.

    This option is recommended for tags that are used in screen objects, because the project may not need to update tags that are not being visualized anywhere. Selecting this option can improve project performance.

  - **Auto** means the project will automatically choose either Always or Screen, depending on where the tag is used in your project. If the tag is only used in a screen object on a project screen, then the scan will default to Screen. But if the tag is configured in any other interface (e.g., Script, Math, Alarm, Trend, Recipe, Report, Scheduler), then the scan will switch to Always and remain there until the project is stopped.

If you are not sure of which option to select, select **Always**. This will guarantee the tag is read and updated.

- **Div** field: Specify the division constant when scale adjustment is required. This value is a division factor in a read operation and a multiplication factor in a write operation.
• **Add** field: Specify the addition constant when scale adjustment is required. This value is a addition factor in a read operation and a subtraction factor in a write operation.

**Note:** The OPC DA specification supports custom item qualities using the high byte of the two-byte quality field. However, such qualities are often vendor-specific or even hand-coded, so it is not possible for InduSoft Web Studio to interpret them. All item qualities other than GOOD (192) will be ignored.

To run the OPC Client runtime task, you can choose to run it automatically on start up, or run the task manually by clicking **Tasks** (either local or remote) on the Home tab of the ribbon. After running this program, a small icon displays in your system tray.

To close the OPC Client runtime task, right-click the icon in the system tray, and click **Exit**.

InduSoft Web Studio, EmbeddedView, and CEView also provide an OPC Server communication task named **Studio.Scada.OPC**. This task starts automatically when any OPC Client (local or remote) attempts to connect to the **Studio.Scada.OPC** server. An OPC Client can exchange data with the tags database (Project Tags, System Tags, and Shared tags) using the OPC interface.

In addition, you can start the OPC Server task automatically when you run the project. Select the OPC Server task in the **Execution Tasks** dialog (**Tasks** on the Home tab of the ribbon), click the **Startup** button, and specify **Automatic**.

**Tip:** You can also use the OPC interface to exchange data between remote stations running InduSoft Web Studio, EmbeddedView, or CEView. You must configure the OPC Client in one station and you must execute the OPC Server in the other station.

### Reading Array Items on the Server

The OPC Client runtime task is capable of reading array items on the specified OPC server. However, it cannot read only certain elements of an array item; the entire item is read during every scan. Also, the way in which the values of the individual array elements are written to project tags in IWS depends on what tag you configure.

**Scenario 1: The project tag is an array (i.e., the tag's Array property is greater than 0)**

In this scenario, each element of the array item on the server is written one-for-one to an element of the project tag. If you specify an array index (e.g., `MyArrayTag[3]`), then writing will begin with that element. If you do not specify an array index, then writing will begin with element 0 by default.

You must make the **Array** property of the project tag large enough to accommodate the elements of the array item on the server. If it is not large enough, then elements will be lost.

**Scenario 2: The project tag is not an array, but the tag name ends with a number**

In this scenario, you can specify the first of a series of sequentially numbered project tags, and the values of the array elements are written to those tags.

**Examples of sequentially numbered project tags**

<table>
<thead>
<tr>
<th>When Tag Name is configured as...</th>
<th>...the values of the array element are written to the following tags:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyTag1</td>
<td>MyTag1, MyTag2, MyTag3</td>
</tr>
<tr>
<td>MyTag3</td>
<td>MyTag3, MyTag4, MyTag5</td>
</tr>
<tr>
<td>MyTag001</td>
<td>MyTag001, MyTag002, MyTag003</td>
</tr>
</tbody>
</table>

You must actually create enough project tags to accommodate the elements of the array item on the server. If there are not enough tags, then elements will be lost. Project tags cannot be created during runtime.

**Scenario 3: The project tag is a class, and the class member ends with a number**
This scenario is similar to Scenario 2 above — you can specify the first of a series of sequentially numbered class members, and the values of the array elements are written to those members.

**Examples of sequentially numbered class members**

<table>
<thead>
<tr>
<th>When Tag Name is configured as...</th>
<th>...the values of the array element are written to the following members:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyClass.Member1</td>
<td>MyClass.Member1, MyClass.Member2, MyClass.Member3</td>
</tr>
<tr>
<td>MyClass.Member3</td>
<td>MyClass.Member3, MyClass.Member4, MyClass.Member5</td>
</tr>
<tr>
<td>MyClass.Member001</td>
<td>MyClass.Member001, MyClass.Member002, MyClass.Member003</td>
</tr>
</tbody>
</table>

You must actually create the class with enough members to accommodate the elements of the array item on the server. If there are not enough members, then elements will be lost. Class members cannot be created during runtime.

**Scenario 4: The project tag is both a class and an array**

If the class member ends with a number, then it is handled the same as in Scenario 3 above, regardless of the fact that the tag is also an array.

If the class member does not end with a number, then it is handled the same as in Scenario 1 above.

**Examples of tags that are both classes and arrays**

<table>
<thead>
<tr>
<th>When Tag Name is configured as...</th>
<th>...the values of the array element are written to the following members:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyArrayClass.Member</td>
<td>MyArrayClass[0].Member, MyArrayClass[1].Member, MyArrayClass[2].Member</td>
</tr>
<tr>
<td>MyArrayClass[3].Member</td>
<td>MyArrayClass[3].Member1, MyArrayClass[3].Member2, MyArrayClass[3].Member3</td>
</tr>
<tr>
<td>MyArrayClass[3].Member</td>
<td>MyArrayClass[3].Member3, MyArrayClass[3].Member4, MyArrayClass[3].Member5</td>
</tr>
<tr>
<td>MyArrayClass[3].Member</td>
<td>MyArrayClass[3].Member001, MyArrayClass[3].Member002, MyArrayClass[3].Member003</td>
</tr>
</tbody>
</table>

**Note:** This functionality is not supported in projects running on Windows Mobile or Windows Embedded Compact.

**OPC Troubleshooting**

When you are using OPC and have problems establishing communication, you should first verify the messages in the LogWin.

If you are running the project on a Windows Embedded target system, there are two ways to check the log:

1. Remote LogWin
2. Local Log

For information about using these logs, please refer to **Using the LogWin task**.

If you find error messages in the log, look them up in this manual/help system, and follow the documented steps for solving the problems. (Use `<CTRL>` + F to find them in the manual; use the Index to find them in the context sensitive help system.)

If you feel that you need to contact your distributor for technical support, make sure that you provide them with the following information:

1. Log file
2. Software vendor and product name of the OPC Server/Client that you are using
3. If possible, a copy or an evaluation version of the OPC Server for testing purposes
4. The contact information for your OPC Server/Client technical support

Three possible errors and their resolutions are listed below...

**Security**

Error Code: **0x80070005 or -2147024891**

Reason for error: When the OPC Client tries to connect to the OPC Server, the DCOM layer usually requires authentication. The computer that is running the OPC Server needs to recognize the user logged on to the OPC Client computer, and such a user needs to have privileges to access the OPC Server.

Solution: The first step is to create a single user on both computers that has Administrator privileges and the same password. Log on with this user to both ends, and then try to establish the connection.

**Name Resolution**

Error: **Couldn't create connection with advise sink, error: -2147022986 (0x80070776)**

Reason for error: There is a problem resolving the computer name.

Solution: This problem can be solved by specifying the IP address of the server instead of specifying the computer name.

**Proxy for Windows CE**

Error: **OPCServer: IIndCP::Advise - Could not query callback interface: 0x80040155**

Reason for error: Your Windows Embedded device is missing the OPCCOMN_PS.dll.

Solution: You should download the .dll to the device and register it. The .dll should be available with your IWS distribution, most probably in \InduSoft Web Studio v7.1\Redist\Wince x.x\processor\ If you do not find the .dll in the folder for your processor, contact your IWS distributor.
Configure an OPC UA Client worksheet

Use the OPC UA Client task/worksheet to communicate with any system that implements the OPC UA server standard for data access.

About OPC UA

The OPC UA Client task/worksheet uses the new OPC Unified Architecture standard introduced by the OPC Foundation. According to the foundation:

The existing OPC COM based specifications have served the OPC Community well over the past 10 years, but as technology moves on so must our interoperability standards. Here are the factors that influenced the decision to create a new architecture:

- Microsoft has deemphasized COM in favor of cross-platform capable Web Services and SOA (Service Oriented Architecture)
- OPC Vendors want a single set of services to expose the OPC data models (DA, A&E, HDA …)
- OPC Vendors want to implement OPC on non-Microsoft systems, including embedded devices
- Other collaborating organizations need a reliable, efficient way to move higher level structured data

In other words, OPC UA is intended to be a platform- and language-independent standard. For more information, go to http://www.opcfoundation.org/UA/.

Note: This feature includes cryptographic software written by Eric Young (eay@cryptsoft.com).

Configure a new OPC UA connection

To configure a new connection to an OPC UA server:

1. Do one of the following:
   - On the Insert tab of the ribbon, in the Communication group, click OPC Client, and then select OPC UA Connection from the drop-down list; or
   - In the Comm tab of the Project Explorer, right-click OPC UA Connections, and then click Insert on the shortcut menu.

   The Server Configuration dialog box is displayed:

2. In the Connection Name box, type a name for the connection.
This name will be displayed in the OPC UA Connections folder in the Project Explorer, and it is the name you will look for when you configure the OPC UA Client worksheet.

3. In the **End point** box, type the URL of the OPC UA server to which you want to connect.

   **Note:** At this time, OPC UA in InduSoft Web Studio only supports end points using OPC.TCP (e.g., `opc.tcp://host:8000/server`). HTTP and HTTPS are not supported.

   If you do not know the URL, click the browse button to the right of the box. The **UA Discovery** dialog box is displayed:

   ![Discovery dialog box](image)

   Use this dialog box to find the discovery server, which publishes a list of OPC UA servers on the network, and then select the server to which you want to connect.

4. In the **User Name** and **Password** boxes, type your login credentials for the OPC UA server.

   You may leave these boxes empty if you want to connect to the server anonymously and you are authorized to do so.

5. If the OPC UA server is configured to require secure communication (also called a ”SecureChannel” in the OPC UA specification), you must take extra steps to exchange certificates between that server and the OPC UA Client task in InduSoft Web Studio. IWS uses a file-based certificate store, which means that the certificates are saved as files in your project folder rather than as entries in a database or system registry.

   The steps to exchange certificates and configure the security settings differ somewhat depending on whether you are using self-signed certificates or certificates signed by a trusted Certificate Authority (CA):
   - **Connect to an OPC UA server using self-signed certificates** on page 426
   - **Connect to an OPC UA server using CA-signed certificates** on page 427

6. In certain situations, you may need to tune the server connection by adjusting the various timeouts. To do so, click Advanced to open the following dialog box:

![Advanced dialog box](image)

Review the settings and configure as needed:

**Session**

- **Session Timeout**
  The session timeout (in milliseconds) for the server connection. If a session times out from inactivity, a new session must be started to resume communication.

- **SecureChannel Lifetime**
  The time (in milliseconds) after which the SecureChannel between the client and server is automatically renewed.

  For more information about the SecureChannel Services, please consult the OPC UA specification.

- **Watchdog Time**
  The time (in milliseconds) between watchdog checks.

- **Watchdog Timeout**
  The timeout (in milliseconds) for a specific watchdog check.

**Server Calls**

- **First Read Max Age**
  The maximum acceptable age (in seconds) of item values when the client connects to the server and requests the values for the first time. If the age of a value in the server’s cache is less than the maximum age, the server will return that cached value to the client. In contrast, if the age of the value is greater than the maximum age, the server will refresh it from the field before returning it to the client.

  Besides an actual age in seconds, this box can accept two special values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Set the maximum age to be equal to the maximum possible value of an integer, which is 2147483647 seconds or approximately 68 years. This has the practical effect of accepting all cached values regardless of age.</td>
</tr>
<tr>
<td>-2</td>
<td>Ignore the first read altogether and wait until the server normally refreshes the item values. Please note that the client cannot write to the server until it has read.</td>
</tr>
</tbody>
</table>
Call Timeout
The timeout (in milliseconds) for individual server calls.

Verify time synchronization
When this option is selected, the time stamp on each value received from the server is compared against the system time on the client, and if the difference between the times is equal to the specified number of hours (i.e., the number of time zones between the server and client), the server and client are deemed to be synchronized and the value is accepted. Otherwise, if the difference is not equal to the specified number of hours, the value is rejected and communication is interrupted.

The default number of hours is 25, which is a special setting that means the hours portion of the time stamp should be disregarded and only the minutes and seconds should be compared. This enables the client to receive values from the server without knowing the server’s time zone.

When this option is cleared, the time stamp on each value received from the server is ignored and the client’s system time is used instead to save the value in the project database.

Browse
Max nodes per call
When browsing for a server item (see below), the maximum number of items/nodes to be returned per server call.

Check item type
When browsing for a server item (see below), check the item’s data type. In some situations, the OPC UA server may not be able to get data types from field devices, and that may cause browsing to become slow and unusable. If that happens, clear this option.

7. Click OK. The connection is saved in the OPC UA Connections folder in the Project Explorer.

Configure a new OPC UA Client worksheet
To configure a new OPC UA Client worksheet:
1. Do one of the following:
   • On the Insert tab of the ribbon, in the Communication group, click OPC Client, and then select OPC UA Client from the drop-down list; or
   • In the Comm tab of the Project Explorer, right-click OPC UA, and then click Insert on the shortcut menu.
A new OPC UA Client worksheet is displayed:

![OPC UA Client worksheet](image)

2. In the **Description** box, type a description of the worksheet. This is for documentation purposes only and does not affect the execution of the worksheet.
3. In the **Connection** list, select the server connection that you configured earlier.
4. For more connection options, click **Advanced**.

   The *Advanced* dialog box is displayed.

   ![Advanced dialog box](image)

   **Advanced dialog box**

   Review the settings and configure as needed:

   **Read actions**

   **Enable subscription**

   When this value is TRUE (non-zero), the client will subscribe to the server nodes so that it can receive notifications when the node values change.

   **Note:** This is enabled by default. If it is disabled, you must use read triggers (see below).

   **Maximum group size**

   The maximum number of server nodes that may be read in a single read operation.

   For example, if you have 1000 items/rows configured in the worksheet and *Maximum group size* is set to 100, 10 read operations will be performed when a read is triggered.

   **Synchronous read trigger**

   When the value of this tag/expression changes, the client reads all of the node values from the server. The read operation is performed synchronously, which means all other client operations are blocked until the read operation is finished. When the read operation is finished, the client increments the tag configured in *Read count*.

   **Asynchronous read trigger**

   The same as *Synchronous read trigger* except that the read operation is performed asynchronously, which means that other client operations may continue while the read operation is being performed.

   **Read count**

   The name of a project tag (integer type) that will receive a count of the number of read operations performed since the project was run.

   **Status**
The name of a project tag (Integer type) that will receive a status code for the last read operation performed by a trigger. If the status code is 0, the read operation finished successfully. For all other status codes, see List of read/write status codes and messages for OPC UA on page 429.

**Status message**
The name of a project tag (String type) that will receive the corresponding status message.

**Maximum aging**
The maximum age (in milliseconds) of values that will be accepted from the server's cache. If a value is older than this, the server will be forced to get the latest value from the field device.

**Sampling rate**
When this option is selected, you can change the rate (in milliseconds) at which the OPC server reads from field devices. By default, the sampling rate is half the publishing rate.

### Write actions

**Enable write on tag change**
When this value is TRUE (non-zero), an asynchronous write operation will be performed automatically whenever the value of a project tag changes. All tag changes that occurred since the last scan of the worksheet will be written in a single write operation, or in multiple write operations if the number of tag changes exceeds the maximum group size.

**Maximum group size**
The maximum number of tag changes that may be written in a single write operation.

For example, if you have 1000 items/rows configured in the worksheet and **Maximum group size** is set to 100, 10 write operations will be performed during each scan of the worksheet.

**Synchronous write trigger**
When the value of this tag/expression changes, the client writes all of the tag values to the server. The write operation is performed synchronously, which means all other client operations are blocked until the write operation is finished. When the write operation is finished, the client increments the tag configured in **Write count**.

**Asynchronous write trigger**
The same as **Synchronous write trigger** except that the write operation is performed asynchronously, which means that other client operations may continue while the write operation is being performed.

**Write count**
The name of a project tag (Integer type) that will receive a count of the number of write operations performed since the project was run.

**Status**
The name of a project tag (Integer type) that will receive a status code for the last write operation performed by a trigger. If the status code is 0, the write operation finished successfully. For all other status codes, see List of read/write status codes and messages for OPC UA on page 429.

**Status message**
The name of a project tag (String type) that will receive the corresponding status message.

**Reload trigger**
The name of a project tag (Boolean, Integer, or Real type) that can be used as a trigger. When the value of this tag changes, the worksheet is reloaded. Tags in curly brackets (e.g., `{tagname}`) configured in the body of the worksheet are refreshed only when the worksheet is reloaded. After the worksheet is reloaded, the value of this tag will be reset to 0.

**Refresh IDs on startup**
When this option is selected, the node IDs in the worksheet will be refreshed from the specified paths every time the project is run.
Communication with Other Devices

**Note:** Refreshing IDs like this may cause the project to take longer to start up, so if you select this option, you should also specify a root node (see below) to limit how much of the server's list of items must scanned.

**Ensure cache synchronization**

When this option is selected, the client will wait after each write operation for confirmation from the server that the node values actually changed. If the client does not receive confirmation, it will restore the previous tag values.

**Enable bit notation**

When this option is selected, bit notation is allowed in the Item column of the worksheet body.

**Note:** IWS’s OPC client does not normally use the Triggering Mode that is defined the OPC protocol. Instead, it allows any change in any tag/expression to be used as a trigger. If you want to use Triggering Mode, configure one worksheet to read the trigger values, and then configure another worksheet that specifies the read values as triggers.

5. Click OK to close the Advanced dialog box and return to the worksheet.

6. In the Status box, type the name of a project tag (Integer type) that will receive connection status codes during project runtime.

7. In the Status Message box, type the name of a project tag (String type) that will receive the corresponding status messages.

8. In the Publish Rate box, type the frequency (in milliseconds) at which the client will request updates from the server.

9. In the Disable box, type a tag/expression. When the value is TRUE (non-zero), the worksheet will not be executed.

10. In the Root node or view box, specify the server node that will serve as the root for all items in the worksheet body. Specifying a root node makes it easier to find items and improves runtime performance.

11. For each row of the worksheet body, specify the following:

   - **Tag Name:** The name of a project tag.
     
     You may use tags in curly brackets (e.g., `{tagname}`), but if you do, make sure to configure the Reload trigger option above.

   - **Item:** The server item with which the project tag should be associated.
     
     To browse the server's list of items, right-click in this field, and then click Browse on the shortcut menu.

   **Note:** If Enable bit notation is selected in the advanced settings, you can select a specific bit of a server node value by appending the bit number to the item name (e.g., `nodepath.bit`). Please note that this is supported only for unsigned variables and 32-bit signed variables; if you try to do it with 16-bit or 8-bit signed variables, the most significant bit (MSB) will not work properly.

   - **Scan:** If Always, the row is continuously scanned during runtime. If Screen, the row is scanned only when a project screen that uses the specified tag is open.

   - **Div:** For scaling — the value is divided by this number when reading from the server, and it is multiplied by this number when writing to the server.

   - **Add:** For scaling — the value is added to the value when reading from the server, and it is subtracted from the value when writing to the server.

   - **Node ID:** The node ID is automatically generated from the full path of the server item.

   **Note:** If the OPC server item is a single-dimension array, you can associate it with an array tag in IWS using a single worksheet row, without array indices. The elements of the OPC server item will be associated automatically, one-for-one, with the elements of the array tag.
Please note that this capability only supports reading from the OPC server item.

12. Save and close the worksheet.

**Enable the OPC UA Client task**

To enable the OPC UA Client task for runtime:

1. On the Home tab of the ribbon, in either the Local Management or Remote Management group (depending on where the project server will be), click **Tasks**.

   The *Execution Tasks* dialog box is displayed.

2. In the list of tasks, select **OPC UA Client Runtime**.

3. Click **Startup**.

   The *Startup* dialog box is displayed.

4. Select **Automatic**, and then click **OK**.

5. Click **OK** again to close the *Execution Tasks* dialog box.

**Troubleshooting**

If you are having trouble with OPC UA Client operations, you can have the client generate a trace log with more detailed information. To do this, manually edit your project file (*projectname*.APP) and add the following settings:

```
[OPC]
UaLogPath= file path and name
UaTraceLevel= level
```

For **UaLogPath**, the file path is relative to your project folder.

For **UaTraceLevel**, select one of the following:

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NoTrace – Disables the trace. This is the default if <strong>UaLogPath</strong> is not configured.</td>
</tr>
<tr>
<td>1</td>
<td>Errors – Internal system errors that require bug fixing.</td>
</tr>
<tr>
<td>2</td>
<td>Warnings – Internal system warnings and external errors. This is the default if <strong>UaLogPath</strong> is configured.</td>
</tr>
<tr>
<td>3</td>
<td>Info – More detailed information about system events.</td>
</tr>
<tr>
<td>4</td>
<td>InterfaceCall – Information needed for debugging.</td>
</tr>
<tr>
<td>5</td>
<td>CtorDtor – Information needed for debugging.</td>
</tr>
<tr>
<td>6</td>
<td>Program – All message content.</td>
</tr>
<tr>
<td>7</td>
<td>FlowData – All messages.</td>
</tr>
</tbody>
</table>

**Note:** The levels are cumulative, so Level 2 includes Level 1, Level 3 includes Levels 1 and 2, and so on.
Connect to an OPC UA server using self-signed certificates

This task describes how to configure the OPC UA server and client to communicate securely using self-signed certificates.

This task is a supplement to the previous task, Configure an OPC UA Client worksheet on page 418. It assumes that you have already created a new OPC UA connection and are now configuring the security settings for that connection.

A self-signed certificate is a certificate signed by the person or organization that uses it. In most cases, it is safe to use only when you directly control both the server and the client. If you do not directly control both, you should consider using certificates signed by a trusted Certificate Authority (CA) instead.

Please note that in this task, "OPC UA server" is a generic reference to any of the available hardware or software products that can run as an OPC UA server. For more information about how to complete certain steps, you should consult the manufacturer's documentation for your specific server. We have included examples based on the free OPC UA Sample Server and UA Configuration Tool offered by the OPC Foundation, but these examples should be used only as a guideline.

To connect to an OPC UA server using self-signed certificates:

1. On your OPC UA server, create a new, self-signed certificate for the server application.
   
   If there is an old server certificate in the certificate store, you may be prompted to overwrite or delete it. It should be safe to do so as long as there are no other clients connected to the server.
   
   Example: Run the UA Configuration Tool. Click Manage Application. In the Application To Manage list, make sure Opc.Ua.SampleServer is selected, and then click Create Application Certificate. The Create Certificate dialog box is displayed. Configure the certificate settings as desired, but leave the CA Key File and CA Password boxes empty; that is what will make the certificate self-signed. Click OK to create the new certificate, and then restart the OPC UA Sample Server to make sure it uses the certificate.

2. In InduSoft Web Studio, in the UA Server Configuration dialog box, click Security.

3. In the Message Security Mode and Security Policy lists, select the settings required by your OPC UA server. In a typical configuration, Message Security Mode is Sign and Encrypt and Security Policy is Basic128Rsa15.

4. Click Trust server certificate.

   In most cases, you do not need to change the Trust List or Issuer Certificate List settings. These are the default locations in your project folder where certificates are stored. You might change the locations if, for example, you have a single folder where you store certificates from several different programs, but we do not recommend it.

   The development application attempts to connect to the OPC UA server, and if it is successful, it gets the server certificate. A warning message is displayed, asking you to confirm that you trust the server certificate.

5. Click OK to confirm.

   The server certificate file is copied to your project folder at: [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\TrustList\Certs\connectionname.der

6. Click Create self-signed certificate.

   The Certificate Creation dialog box is displayed.
7. In the **Certificate Creation** dialog box, complete the certificate information as needed, and then click **Create**. The development application creates a new client certificate file and saves it in your project folder at: ` […]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config \UAClientCertificate.der`

8. On your OPC UA server, import the client certificate (from the location given in the previous step) into the server’s trust list.
   Example: In the UA Configuration Tool, click **Manage Security**, and then click **Import Certificate to Trust**. A standard **Open File** dialog box is displayed. Use the dialog box to locate and select the client certificate, and then click **Open**. You will be asked to confirm the import. Click **Yes**.

Once the certificates have been exchanged — that is, once the server certificate is in the client’s (i.e., your project’s) trust list and the client certificate is in the server’s trust list — the OPC UA connection should be properly configured and ready for secure, two-way communication.

To test the connection, insert a new OPC UA Client worksheet, select the connection you have created, and then try to browse for items on the server.

### Connect to an OPC UA server using CA-signed certificates

This task describes how to configure the OPC UA server and client to communicate securely using CA-signed certificates.

This task is a supplement to the previous task, **Configure an OPC UA Client worksheet** on page 418. It assumes that you have already created a new OPC UA connection and are now configuring the security settings for that connection.

This task also assumes that you have acquired a CA-signed certificate that you are authorized to use with your project. The certificate should be a Distinguished Encoding Rules file (*.der). You should also have the password for the certificate.

A CA-signed certificate is a certificate signed by a trusted Certificate Authority (CA). CA-signed certificates are safe to use in most situations, even when you do not directly control both the server and the client, because you agree to trust the CA.

You can buy certificates from Certificate Authority companies such as VeriSign, DigiCert, and GeoTrust, although InduSoft does not endorse any particular company. You can also create your own Certificate Authority on some servers, but you should do so only if you understand the security issues involved. Specifically, you should understand the difference between self-signed certificates and certificates signed by a self-created CA. That is beyond the scope of this documentation.

Once you have a CA-signed certificate, you can use it to sign other certificates that you create for your server and client applications. Each subsequent certificate is considered to be signed by the same trusted CA.

Please note that in this task, “OPC UA server” is a generic reference to any of the available hardware or software products that can run as an OPC UA server. For more information about how to complete certain steps, you should consult the manufacturer’s documentation for your specific server. We have included examples based on the free OPC UA Sample Server and UA Configuration Tool offered by the OPC Foundation, but these examples should be used only as a guideline.

To connect to an OPC UA server using CA-signed certificates:

1. On your OPC UA server, import the CA-signed certificate into the server’s certificate store.
   The CA-signed certificate will be used to sign the server and client certificates that you create in the following steps.
   Example: Run the UA Configuration Tool. Click **Manage Certificates**, and then click **Import Certificate to Store**. A standard **Open File** dialog box is displayed. Use the dialog to locate and select the CA-signed certificate, and then click **Open**. You will be asked to confirm the import. Click **Yes**.

2. On your OPC UA server, create a new, signed server certificate.
   If there is an old server certificate in the certificate store, you may be prompted to overwrite or delete it. It should be safe to do so as long as there are no other clients connected to the server.
   Example: In the UA Configuration Tool, click **Manage Application**. In the **Application To Manage** list, make sure **Opc.Ua.SampleServer** is selected, and then click **Create Application Certificate**. The **Create Certificate** dialog is displayed. Configure the certificate settings as desired. In the **CA Key File** box, click **Browse** to select the CA-signed certificate that you imported in the previous step. In the **CA Password** box, type the password for the
CA-signed certificate. Click OK to create the new certificate, and then restart the OPC UA Sample Server to make sure it uses the certificate.

3. Add the CA-signed certificate your project's issuer list — copy (but not move) the certificate file to your project folder at: [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\IssuerList\Certs\ The issuer list determines which other certificates your project will trust. When you tell your project to trust the signed server certificate, that certificate will be checked against the CA-signed certificate in the issuer list to make sure the signatures match.


5. In the Message Security Mode and Security Policy lists, select the settings required by your OPC UA server. In a typical configuration, Message Security Mode is Sign and Encrypt and Security Policy is Basic128Rsa15.

6. Click Trust server certificate. In most cases, you do not need to change the Trust List or Issuer Certificate List settings. These are the default locations in your project folder where certificates are stored. You might change the locations if, for example, you have a single folder where you store certificates from several different programs, but we do not recommend it.

   The development application attempts to connect to the OPC UA server, and if it is successful, it gets the server certificate. A warning message is displayed, asking you to confirm that you trust the server certificate.

7. Click OK to confirm.

   The server certificate file is copied to your project folder at: [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\TrustList\Certs\connectionname.der

8. Click Create self-signed certificate.

   Although you are creating a self-signed certificate here, you will use the server to reissue it as a signed certificate.

   The Certificate Creation dialog is displayed.

9. In the Certificate Creation dialog, complete the certificate information as needed, and then click Create.

   The development application creates a new client certificate file and saves it in your project folder at: [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\UAClientCertificate.der

10. On your OPC UA server, import the self-signed client certificate (from the location given in the previous step) and then reissue it as a signed client certificate. Make sure the certificate is issued with a PEM-format key.

    Example: In the UA Configuration Tool, click Manage Certificates, and then click Import and Issue Certificate. A standard Open File dialog is displayed. Use the dialog to locate and select the client certificate, and then click Open. The Create Certificate dialog is displayed. Note the Application Name, because this will be the name of the certificate and key files that you create. In the CA Password box, type the password for the CA-signed certificate. In the Key Format list, select PEM. Click OK.

    The reissued client certificate and key files are created in the server's certificate store.
11. Copy the reissued client certificate and key files from the server's certificate store back to your project folder.

   These files will replace the existing, self-signed certificate and key files that you created previously.

   a) Find the reissued certificate file.
      Example: For the OPC UA Sample Server, the file is located at: C:\ProgramData\OPC Foundation\CertificateStores\UA Certificate Authorities\certs\applicationname [IDstring].der

   b) Copy (but not move) the certificate file to: [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\UAClientCertificate.der

   c) Find the reissued key file.
      Example: For the OPC UA Sample Server, the file is located at: C:\ProgramData\OPC Foundation\CertificateStores\UA Certificate Authorities\private\applicationname [IDstring].pem

   d) Copy (but not move) the key file to: [...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Config\UAClientCertificatePrivateKey.pem

   Once the certificates have been exchanged — that is, once the server certificate is in the client's (i.e., your project's) trust list and the client certificate is in the server's trust list — the OPC UA connection should be properly configured and ready for secure, two-way communication.

   To test the connection, insert a new OPC UA Client worksheet, select the connection you have created, and then try to browse for items on the server.

**List of read/write status codes and messages for OPC UA**

This is a list of the possible status codes and messages that may be generated by read/write operations in OPC UA.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Status Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2159476736</td>
<td>Bad – Max Connections Reached</td>
</tr>
<tr>
<td>-215941200</td>
<td>Bad – Syntax Error</td>
</tr>
<tr>
<td>-2159345664</td>
<td>Bad – Would Block</td>
</tr>
<tr>
<td>-2159280128</td>
<td>Bad – Expected Stream To Block</td>
</tr>
<tr>
<td>-2159214592</td>
<td>Bad – Operation Abandoned</td>
</tr>
<tr>
<td>-2159149056</td>
<td>Bad – Waiting For Response</td>
</tr>
<tr>
<td>-215903520</td>
<td>Bad – No Data Available</td>
</tr>
<tr>
<td>-2159017984</td>
<td>Bad – End Of Stream</td>
</tr>
<tr>
<td>-2147352576</td>
<td>Bad – Internal Error</td>
</tr>
<tr>
<td>-2147287040</td>
<td>Bad – Out Of Memory</td>
</tr>
<tr>
<td>-2147221504</td>
<td>Bad – Resource Unavailable</td>
</tr>
<tr>
<td>-2147155968</td>
<td>Bad – Communication Error</td>
</tr>
<tr>
<td>-2147090432</td>
<td>Bad – Encoding Error</td>
</tr>
<tr>
<td>-2147024896</td>
<td>Bad – Decoding Error</td>
</tr>
<tr>
<td>-2146959380</td>
<td>Bad – Encoding Limits Exceeded</td>
</tr>
<tr>
<td>-2146893824</td>
<td>Bad – Unknown Response</td>
</tr>
<tr>
<td>-2146826288</td>
<td>Bad – Timeout</td>
</tr>
<tr>
<td>-2146762752</td>
<td>Bad – Service Unsupported</td>
</tr>
<tr>
<td>-2146697216</td>
<td>Bad – Shutdown</td>
</tr>
<tr>
<td>-2146631680</td>
<td>Bad – Server Not Connected</td>
</tr>
<tr>
<td>-2146566144</td>
<td>Bad – Server Halted</td>
</tr>
<tr>
<td>Status Code</td>
<td>Status Message</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>-2146500608</td>
<td>Bad – Nothing To Do</td>
</tr>
<tr>
<td>-2146435072</td>
<td>Bad – Too Many Operations</td>
</tr>
<tr>
<td>-2146369536</td>
<td>Bad – Data Type Id Unknown</td>
</tr>
<tr>
<td>-2146304000</td>
<td>Bad – Certificate Invalid</td>
</tr>
<tr>
<td>-2146238464</td>
<td>Bad – Security Checks Failed</td>
</tr>
<tr>
<td>-2146172928</td>
<td>Bad – Certificate Time Invalid</td>
</tr>
<tr>
<td>-2146107392</td>
<td>Bad – Certificate Issuer Time Invalid</td>
</tr>
<tr>
<td>-2146041856</td>
<td>Bad – Certificate Host Name Invalid</td>
</tr>
<tr>
<td>-2145976320</td>
<td>Bad – Certificate Uri Invalid</td>
</tr>
<tr>
<td>-2145910784</td>
<td>Bad – Certificate Use Not Allowed</td>
</tr>
<tr>
<td>-2145845248</td>
<td>Bad – Certificate Issuer Use Not Allowed</td>
</tr>
<tr>
<td>-2145779712</td>
<td>Bad – Certificate Untrusted</td>
</tr>
<tr>
<td>-2145714176</td>
<td>Bad – Certificate Revocation Unknown</td>
</tr>
<tr>
<td>-2145648640</td>
<td>Bad – Certificate Issuer Revocation Unknown</td>
</tr>
<tr>
<td>-2145583104</td>
<td>Bad – Certificate Revoked</td>
</tr>
<tr>
<td>-2145517568</td>
<td>Bad – Certificate Issuer Revoked</td>
</tr>
<tr>
<td>-2145452032</td>
<td>Bad – User Access Denied</td>
</tr>
<tr>
<td>-2145386496</td>
<td>Bad – Identity Token Invalid</td>
</tr>
<tr>
<td>-2145320960</td>
<td>Bad – Identity Token Rejected</td>
</tr>
<tr>
<td>-2145255424</td>
<td>Bad – Secure Channel Id Invalid</td>
</tr>
<tr>
<td>-2145189888</td>
<td>Bad – Invalid Timestamp</td>
</tr>
<tr>
<td>-2145124352</td>
<td>Bad – Nonce Invalid</td>
</tr>
<tr>
<td>-2145058816</td>
<td>Bad – Session Id Invalid</td>
</tr>
<tr>
<td>-214493280</td>
<td>Bad – Session Closed</td>
</tr>
<tr>
<td>-2144927744</td>
<td>Bad – Session Not Activated</td>
</tr>
<tr>
<td>-2144862208</td>
<td>Bad – Subscription Id Invalid</td>
</tr>
<tr>
<td>-2144731136</td>
<td>Bad – Request Header Invalid</td>
</tr>
<tr>
<td>-2144656500</td>
<td>Bad – Timestamps To Return Invalid</td>
</tr>
<tr>
<td>-2144600064</td>
<td>Bad – Request Cancelled By Client</td>
</tr>
<tr>
<td>-2144272384</td>
<td>Bad – No Communication</td>
</tr>
<tr>
<td>-2144206848</td>
<td>Bad – Waiting For Initial Data</td>
</tr>
<tr>
<td>-2144141312</td>
<td>Bad – Node Id Invalid</td>
</tr>
<tr>
<td>-2144075776</td>
<td>Bad – Node Id Unknown</td>
</tr>
<tr>
<td>-2144010240</td>
<td>Bad – Attribute Id Invalid</td>
</tr>
<tr>
<td>-2143944704</td>
<td>Bad – Index Range Invalid</td>
</tr>
<tr>
<td>-2143879168</td>
<td>Bad – Index Range No Data</td>
</tr>
<tr>
<td>-2143813632</td>
<td>Bad – Data Encoding Invalid</td>
</tr>
<tr>
<td>-2143748096</td>
<td>Bad – Data Encoding Unsupported</td>
</tr>
<tr>
<td>-2143682560</td>
<td>Bad – Not Readable</td>
</tr>
<tr>
<td>Status Code</td>
<td>Status Message</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>-2143617024</td>
<td>Bad – Not Writable</td>
</tr>
<tr>
<td>-2143551488</td>
<td>Bad – Out Of Range</td>
</tr>
<tr>
<td>-2143485952</td>
<td>Bad – Not Supported</td>
</tr>
<tr>
<td>-2143420416</td>
<td>Bad – Not Found</td>
</tr>
<tr>
<td>-2143354880</td>
<td>Bad – Object Deleted</td>
</tr>
<tr>
<td>-2143289344</td>
<td>Bad – Not Implemented</td>
</tr>
<tr>
<td>-2143223808</td>
<td>Bad – Monitoring Mode Invalid</td>
</tr>
<tr>
<td>-2143158272</td>
<td>Bad – Monitored Item Id Invalid</td>
</tr>
<tr>
<td>-2143092736</td>
<td>Bad – Monitored Item Filter Invalid</td>
</tr>
<tr>
<td>-2143027200</td>
<td>Bad – Monitored Item Filter Unsupported</td>
</tr>
<tr>
<td>-2142961664</td>
<td>Bad – Filter Not Allowed</td>
</tr>
<tr>
<td>-2142896128</td>
<td>Bad – Structure Missing</td>
</tr>
<tr>
<td>-2142830592</td>
<td>Bad – Event Filter Invalid</td>
</tr>
<tr>
<td>-2142765056</td>
<td>Bad – Content Filter Invalid</td>
</tr>
<tr>
<td>-2142699520</td>
<td>Bad – Filter Operand Invalid</td>
</tr>
<tr>
<td>-2142633984</td>
<td>Bad – Continuation Point Invalid</td>
</tr>
<tr>
<td>-2142568448</td>
<td>Bad – No Continuation Points</td>
</tr>
<tr>
<td>-2142502912</td>
<td>Bad – Reference Type Id Invalid</td>
</tr>
<tr>
<td>-2142437376</td>
<td>Bad – Browse Direction Invalid</td>
</tr>
<tr>
<td>-2142371840</td>
<td>Bad – Node Not In View</td>
</tr>
<tr>
<td>-2142306304</td>
<td>Bad – Server Uri Invalid</td>
</tr>
<tr>
<td>-2142240768</td>
<td>Bad – Server Name Missing</td>
</tr>
<tr>
<td>-2142175232</td>
<td>Bad – Discovery Uri Missing</td>
</tr>
<tr>
<td>-2142109696</td>
<td>Bad – Sempahore File Missing</td>
</tr>
<tr>
<td>-2142044160</td>
<td>Bad – Request Type Invalid</td>
</tr>
<tr>
<td>-2141978624</td>
<td>Bad – Security Mode Rejected</td>
</tr>
<tr>
<td>-2141913088</td>
<td>Bad – Security Policy Rejected</td>
</tr>
<tr>
<td>-2141847552</td>
<td>Bad – Too Many Sessions</td>
</tr>
<tr>
<td>-2141782016</td>
<td>Bad – User Signature Invalid</td>
</tr>
<tr>
<td>-2141716480</td>
<td>Bad – Application Signature Invalid</td>
</tr>
<tr>
<td>-2141650944</td>
<td>Bad – No Valid Certificates</td>
</tr>
<tr>
<td>-2141585408</td>
<td>Bad – Request Cancelled By Request</td>
</tr>
<tr>
<td>-2141519872</td>
<td>Bad – Parent Node Id Invalid</td>
</tr>
<tr>
<td>-2141454336</td>
<td>Bad – Reference Not Allowed</td>
</tr>
<tr>
<td>-2141388800</td>
<td>Bad – Node Id Rejected</td>
</tr>
<tr>
<td>-2141323264</td>
<td>Bad – Node Id Exists</td>
</tr>
<tr>
<td>-2141257728</td>
<td>Bad – Node Class Invalid</td>
</tr>
<tr>
<td>-2141192192</td>
<td>Bad – Browse Name Invalid</td>
</tr>
<tr>
<td>-2141126656</td>
<td>Bad – Browse Name Duplicated</td>
</tr>
<tr>
<td>Status Code</td>
<td>Status Message</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>-2141061120</td>
<td>Bad – Node Attributes Invalid</td>
</tr>
<tr>
<td>-2140995584</td>
<td>Bad – Type Definition Invalid</td>
</tr>
<tr>
<td>-2140930048</td>
<td>Bad – Source Node Id Invalid</td>
</tr>
<tr>
<td>-2140864512</td>
<td>Bad – Target Node Id Invalid</td>
</tr>
<tr>
<td>-2140798976</td>
<td>Bad – Duplicate Reference Not Allowed</td>
</tr>
<tr>
<td>-2140733440</td>
<td>Bad – Invalid Self Reference</td>
</tr>
<tr>
<td>-2140679004</td>
<td>Bad – Reference Local Only</td>
</tr>
<tr>
<td>-2140602368</td>
<td>Bad – No Delete Rights</td>
</tr>
<tr>
<td>-2140536832</td>
<td>Bad – Server Index Invalid</td>
</tr>
<tr>
<td>-2140471296</td>
<td>Bad – View Id Unknown</td>
</tr>
<tr>
<td>-2140340224</td>
<td>Bad – Too Many Matches</td>
</tr>
<tr>
<td>-2140274688</td>
<td>Bad – Query Too Complex</td>
</tr>
<tr>
<td>-2140209152</td>
<td>Bad – No Match</td>
</tr>
<tr>
<td>-2140143616</td>
<td>Bad – Max Age Invalid</td>
</tr>
<tr>
<td>-2140079080</td>
<td>Bad – History Operation Invalid</td>
</tr>
<tr>
<td>-2140012544</td>
<td>Bad – History Operation Unsupported</td>
</tr>
<tr>
<td>-2139947008</td>
<td>Bad – Write Not Supported</td>
</tr>
<tr>
<td>-2139881472</td>
<td>Bad – Type Mismatch</td>
</tr>
<tr>
<td>-2139815936</td>
<td>Bad – Method Invalid</td>
</tr>
<tr>
<td>-2139750400</td>
<td>Bad – Arguments Missing</td>
</tr>
<tr>
<td>-2139684864</td>
<td>Bad – Too Many Subscriptions</td>
</tr>
<tr>
<td>-2139619328</td>
<td>Bad – Too Many Publish Requests</td>
</tr>
<tr>
<td>-2139553792</td>
<td>Bad – No Subscription</td>
</tr>
<tr>
<td>-2139488256</td>
<td>Bad – Sequence Number Unknown</td>
</tr>
<tr>
<td>-2139422720</td>
<td>Bad – Message Not Available</td>
</tr>
<tr>
<td>-2139357184</td>
<td>Bad – Insufficient Client Profile</td>
</tr>
<tr>
<td>-2139291648</td>
<td>Bad – Tcp Server Too Busy</td>
</tr>
<tr>
<td>-2139226112</td>
<td>Bad – Tcp Message Type Invalid</td>
</tr>
<tr>
<td>-2139160576</td>
<td>Bad – Tcp Secure Channel Unknown</td>
</tr>
<tr>
<td>-2139095040</td>
<td>Bad – Tcp Message Too Large</td>
</tr>
<tr>
<td>-2139029504</td>
<td>Bad – Tcp Not Enough Resources</td>
</tr>
<tr>
<td>-2138963968</td>
<td>Bad – Tcp Internal Error</td>
</tr>
<tr>
<td>-2138898432</td>
<td>Bad – Tcp Endpoint Url Invalid</td>
</tr>
<tr>
<td>-2138832896</td>
<td>Bad – Request Interrupted</td>
</tr>
<tr>
<td>-2138767360</td>
<td>Bad – Request Timeout</td>
</tr>
<tr>
<td>-2138701824</td>
<td>Bad – Secure Channel Closed</td>
</tr>
<tr>
<td>-2138636288</td>
<td>Bad – Secure Channel Token Unknown</td>
</tr>
<tr>
<td>-2138570752</td>
<td>Bad – Sequence Number Invalid</td>
</tr>
<tr>
<td>-2138505216</td>
<td>Bad – Configuration Error</td>
</tr>
<tr>
<td>Status Code</td>
<td>Status Message</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>-2138439680</td>
<td>Bad – Not Connected</td>
</tr>
<tr>
<td>-2138374144</td>
<td>Bad – Device Failure</td>
</tr>
<tr>
<td>-2138308608</td>
<td>Bad – Sensor Failure</td>
</tr>
<tr>
<td>-2138243072</td>
<td>Bad – Out Of Service</td>
</tr>
<tr>
<td>-2138177536</td>
<td>Bad – Deadband Filter Invalid</td>
</tr>
<tr>
<td>-2137587712</td>
<td>Bad – Refresh In Progress</td>
</tr>
<tr>
<td>-2137522176</td>
<td>Bad – Condition Already Disabled</td>
</tr>
<tr>
<td>-2137456640</td>
<td>Bad – Condition Disabled</td>
</tr>
<tr>
<td>-2137391104</td>
<td>Bad – Event Id Unknown</td>
</tr>
<tr>
<td>-2137325568</td>
<td>Bad – No Data</td>
</tr>
<tr>
<td>-2137194496</td>
<td>Bad – Data Lost</td>
</tr>
<tr>
<td>-2137128960</td>
<td>Bad – Data Unavailable</td>
</tr>
<tr>
<td>-2137063424</td>
<td>Bad – Entry Exists</td>
</tr>
<tr>
<td>-2136997888</td>
<td>Bad – No Entry Exists</td>
</tr>
<tr>
<td>-2136932352</td>
<td>Bad – Timestamp Not Supported</td>
</tr>
<tr>
<td>-2136276992</td>
<td>Bad – Invalid Argument</td>
</tr>
<tr>
<td>-2136211456</td>
<td>Bad – Connection Rejected</td>
</tr>
<tr>
<td>-2136145920</td>
<td>Bad – Disconnect</td>
</tr>
<tr>
<td>-2136080384</td>
<td>Bad – Connection Closed</td>
</tr>
<tr>
<td>-2136014846</td>
<td>Bad – Invalid State</td>
</tr>
<tr>
<td>-2135425024</td>
<td>Bad – Request Too Large</td>
</tr>
<tr>
<td>-2135359488</td>
<td>Bad – Response Too Large</td>
</tr>
<tr>
<td>-2135228416</td>
<td>Bad – Event Not Acknowledgeable</td>
</tr>
<tr>
<td>-2135097344</td>
<td>Bad – Invalid Timestamp Argument</td>
</tr>
<tr>
<td>-2135031808</td>
<td>Bad – Protocol Version Unsupported</td>
</tr>
<tr>
<td>-2134966272</td>
<td>Bad – State Not Active</td>
</tr>
<tr>
<td>-2134835200</td>
<td>Bad – Filter Operator Invalid</td>
</tr>
<tr>
<td>-2134769664</td>
<td>Bad – Filter Operator Unsupported</td>
</tr>
<tr>
<td>-2134704128</td>
<td>Bad – Filter Operand Count Mismatch</td>
</tr>
<tr>
<td>-2134638592</td>
<td>Bad – Filter Element Invalid</td>
</tr>
<tr>
<td>-2134573056</td>
<td>Bad – Filter Literal Invalid</td>
</tr>
<tr>
<td>-2134507520</td>
<td>Bad – Identity Change Not Supported</td>
</tr>
<tr>
<td>-2134376448</td>
<td>Bad – Not Type Definition</td>
</tr>
<tr>
<td>-2134310912</td>
<td>Bad – View Timestamp Invalid</td>
</tr>
<tr>
<td>-2134245376</td>
<td>Bad – View Parameter Mismatch</td>
</tr>
<tr>
<td>-2134179840</td>
<td>Bad – View Version Invalid</td>
</tr>
<tr>
<td>-2134114304</td>
<td>Bad – Condition Already Enabled</td>
</tr>
<tr>
<td>-2134048768</td>
<td>Bad – Dialog Not Active</td>
</tr>
<tr>
<td>-2133983232</td>
<td>Bad – Dialog Response Invalid</td>
</tr>
<tr>
<td>Status Code</td>
<td>Status Message</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>-2133917696</td>
<td>Bad – Condition Branch Already Acked</td>
</tr>
<tr>
<td>-2133952160</td>
<td>Bad – Condition Branch Already Confirmed</td>
</tr>
<tr>
<td>-2133786624</td>
<td>Bad – Condition Already Shelved</td>
</tr>
<tr>
<td>-2133721088</td>
<td>Bad – Condition Not Shelved</td>
</tr>
<tr>
<td>-2133655552</td>
<td>Bad – Shelving Time Out Of Range</td>
</tr>
<tr>
<td>-21335900016</td>
<td>Bad – Aggregate List Mismatch</td>
</tr>
<tr>
<td>-2133524480</td>
<td>Bad – Aggregate Not Supported</td>
</tr>
<tr>
<td>-2133458944</td>
<td>Bad – Aggregate Invalid Inputs</td>
</tr>
<tr>
<td>-2133393408</td>
<td>Bad – Bound Not Found</td>
</tr>
<tr>
<td>-2133327872</td>
<td>Bad – Bound Not Supported</td>
</tr>
<tr>
<td>-2133196800</td>
<td>Bad – Aggregate Configuration Rejected</td>
</tr>
<tr>
<td>2949120</td>
<td>Good – Subscription Transferred</td>
</tr>
<tr>
<td>3014656</td>
<td>Good – Completes Asynchronously</td>
</tr>
<tr>
<td>3080192</td>
<td>Good – Overload</td>
</tr>
<tr>
<td>3145728</td>
<td>Good – Clamped</td>
</tr>
<tr>
<td>9830400</td>
<td>Good – Local Override</td>
</tr>
<tr>
<td>10616832</td>
<td>Good – Entry Inserted</td>
</tr>
<tr>
<td>10682368</td>
<td>Good – Entry Replaced</td>
</tr>
<tr>
<td>10813440</td>
<td>Good – No Data</td>
</tr>
<tr>
<td>10878976</td>
<td>Good – More Data</td>
</tr>
<tr>
<td>10944512</td>
<td>Good – Communication Event</td>
</tr>
<tr>
<td>11010048</td>
<td>Good – Shutdown Event</td>
</tr>
<tr>
<td>11075584</td>
<td>Good – Call Again</td>
</tr>
<tr>
<td>11141120</td>
<td>Good – Non Critical Timeout</td>
</tr>
<tr>
<td>12189696</td>
<td>Good – Results May Be Incomplete</td>
</tr>
<tr>
<td>14221312</td>
<td>Good – Data Ignored</td>
</tr>
<tr>
<td>1080819712</td>
<td>Uncertain – Reference Out Of Server</td>
</tr>
<tr>
<td>1083113472</td>
<td>Uncertain – No Communication Last Usable Value</td>
</tr>
<tr>
<td>1083179008</td>
<td>Uncertain – Last Usable Value</td>
</tr>
<tr>
<td>1083244544</td>
<td>Uncertain – Substitute Value</td>
</tr>
<tr>
<td>1083310080</td>
<td>Uncertain – Initial Value</td>
</tr>
<tr>
<td>1083375616</td>
<td>Uncertain – Sensor Not Accurate</td>
</tr>
<tr>
<td>1083441152</td>
<td>Uncertain – Engineering Units Exceeded</td>
</tr>
<tr>
<td>1083506688</td>
<td>Uncertain – Sub Normal</td>
</tr>
<tr>
<td>1084489728</td>
<td>Uncertain – Data Sub Normal</td>
</tr>
<tr>
<td>1086062592</td>
<td>Uncertain – Reference Not Deleted</td>
</tr>
<tr>
<td>1086324736</td>
<td>Uncertain – Not All Nodes Available</td>
</tr>
</tbody>
</table>
Configure an OPC .NET Client connection to an OPC .NET server

The OPC .NET Client task/worksheet is to communicate with any system that implements the OPC .NET Server protocol.

**About OPC .NET**

The OPC .NET Client task/worksheet uses the new OPC .NET communication protocol introduced by the OPC Foundation. According to the foundation:

**OPC .NET 3.0 (WCF) (formerly known as OPC Express Interface (Xi)) is the continued evolution of OPC Foundation native support for Microsoft platforms that previously included: the OPC Foundation Automation Wrapper for Visual Basic 6 and the OPC .NET 2.0 API for VB.NET and C#. It bridges the gap between Microsoft.NET and the world of OPC Classic.**

In other words, OPC .NET is intended to be a new version of OPC that leverages Microsoft’s latest technologies. For more information, go to the OPC Foundation website.

**Configuring a new OPC .NET server connection**

To configure a new connection to an OPC .NET server:

1. Do one of the following:
   - On the Insert tab of the ribbon, in the Communication group, click **OPC Client** and then select **OPC .NET Connection** from the drop-down list; or
   - In the Comm tab of the Project Explorer, right-click **OPC .NET Connections** and then click **Insert** on the shortcut menu.

The **OPC Server Connection** dialog is displayed:

2. In the **Connection Name** box, type a name for the connection.
   
   This name will be displayed in the OPC .NET Connections folder in the Project Explorer, and it is the name you will look for when you configure the OPC .NET Client worksheet.

3. In the **Discovery Server** list, type or select the URL of the discovery server that publishes a list of OPC .NET servers on the network. If no discovery server appears to be available, click **Refresh** to scan the network again.
4. In the Server list, type or select the URL of the OPC .NET server to which you want to connect. If the server you want doesn’t appear to be available, click Refresh the update the list from the discovery server.

5. In the Protocol list, select the network protocol to be used to connect to the server. (For more information about the available protocols, please refer to Microsoft’s documentation for Windows Communication Foundation.)

   **Note:** If your OPC .NET configuration has different servers/endpoints for each operation, then do the following:
   
   1. In the Protocol list, select custom. The Custom button becomes enabled.
   2. Click the Custom button. The Endpoints dialog is displayed.

   3. Use the dialog to specify the server/endpoint for each operation.

6. In the User Name and Password boxes, type your login credentials for the OPC .NET server.

   **Note:** If the OPC .NET server is using Microsoft Active Directory, then User Name must be in the domain\username format.

7. Select Callback to have the server send values to the client only when the values change, or select Polling to have the client periodically request values from the server.

   In most cases, you should select Polling to keep the connection active.

8. Click OK. The connection is saved in the OPC .NET Connections folder in the Project Explorer.

**Configuring a new OPC .NET Client worksheet**

To configure a new OPC .NET Client worksheet:

1. Do one of the following:
   
   • On the Insert tab of the ribbon, in the Communication group, click OPC Client and then select OPC .NET Client from the drop-down list; or
   
   • In the Comm tab of the Project Explorer, right-click OPC .NET and then click Insert on the shortcut menu.
A new OPC .NET Client worksheet is displayed:

![OPC .NET Client worksheet](image)

2. In the **Description** box, type a description of the worksheet. This is for documentation purposes only and does not affect the execution of the worksheet.

3. In the **Connection** list, select the OPC Server connection that you configured earlier.

4. For more connection options, click **Advanced**.

   The **Advanced** dialog is displayed:

   ![Advanced dialog](image)

   Review the options and configure as needed:

<table>
<thead>
<tr>
<th>Area / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read actions</td>
<td>Enable subscription</td>
</tr>
</tbody>
</table>
### Area / Element | Description
--- | ---
| **Note:** This is enabled by default. If it is disabled, then you must use a read trigger (see below). | **Maximum group size**
- The maximum number of server nodes that may be read in a single read operation.
- For example, if you have 1000 items/rows configured in the worksheet and **Maximum group size** is set to 100, 10 read operations will be performed when a read is triggered.

| **Note:** This is enabled by default. If it is disabled, you must use write triggers (see below). | **Synchronous read trigger**
- When the value of this tag/expression changes, the client reads all of the node values from the server.
- The read is performed synchronously, which means all other client operations are blocked until the read is finished. When the read is finished, the client increments the tag configured in **Read count**.

| N/A for OPC .NET Client. | **Asynchronous read trigger**

| The name of a tag (Integer type) that will receive a count of the number of read operations performed since the project was run. | **Read count**

| The name of a tag (Integer type) that will receive a status code for the last read operation performed by a trigger. | **Status**

| The name of a tag (String type) that will receive the corresponding status message. | **Status message**

| N/A for OPC .NET Client. | **Maximum aging**

| N/A for OPC .NET Client. | **Sampling rate**

| **Note:** This is enabled by default. If it is disabled, you must use write triggers (see below). | **Enable write on tag change**
- When this value is TRUE (non-zero), a write operation will be performed automatically whenever the value of a project tag changes. All tag changes that occurred since the last scan of the worksheet will be written in a single write operation, or in multiple write operations if the number of tag changes exceeds the maximum group size.

| **Maximum group size**
- The maximum number of tag changes that may be written in a single write operation.
- For example, if you have 1000 items/rows configured in the worksheet and **Maximum group size** is set to 100, then 10 write operations will be performed during each scan of the worksheet. | **Synchronous write trigger**
- When the value of this tag/expression changes, the client writes all of the tag values to the server. The write operation is performed synchronously, which means all other client operations are blocked until the write operation is finished. When the write operation is finished, the client increments the tag configured in **Write count**.

| N/A for OPC .NET Client. | **Asynchronous write trigger**

| The name of a tag (Integer type) that will receive a count of the number of write operations performed since the project was run. | **Write count**

| The name of a tag (Integer type) that will receive a status code for the last write operation performed by a trigger. | **Status**

| The name of a tag (String type) that will receive the corresponding status message. | **Status message**

| **Reload trigger**
- The name of a tag (Boolean, Integer, or Real type) that can be used as a trigger. When the value of this tag changes, the worksheet is reloaded. Tags in curly brackets (e.g., `{tagname}`) configured in the body of the worksheet are refreshed only when the worksheet is reloaded. After the worksheet is reloaded, the value of this tag will be reset to 0. | **Reload trigger**

| N/A for OPC .NET Client. | **Refresh IDs on startup**

| When this option is selected, the client will wait after each write operation for confirmation from the server that the node values actually changed. If the client does not receive confirmation, it will restore the previous tag values. | **Ensure cache synchronization**

| When this option is selected, bit notation is allowed in the **Item** column of the worksheet body. | **Enable bit notation**

5. Click **OK** to close the *Advanced* dialog and return to the worksheet.
6. In the **Status** box, type the name of a tag (Integer type) that will receive connection status codes during project runtime.

7. In the **Status Message** box, type the name of a tag (String type) that will receive the corresponding status messages.

8. In the **Publish Rate** box, type the frequency (in milliseconds) at which the client will request updates from the server.

9. In the **Disable** box, type a tag/expression. When the value is TRUE (non-zero), the worksheet will not be executed.

10. In the **Root node or view** box, specify the server node that will serve as the root for all items in the worksheet body. Specifying a root node makes it easier to find items and improves runtime performance.

11. For each row of the worksheet body, specify the following:
   - **Tag Name**: The name of a project tag.
     You may use tags in curly brackets (e.g., `{tagname}`), but if you do, then be sure to configure the **Reload trigger** option above.
   - **Item**: The server item with which the project tag should be associated.
     To browse the server’s list of items, right-click in this field and then click **Browse** on the shortcut menu.
   - **Scan**: If **Always**, then the row is continuously scanned during runtime. If **Screen**, then the row is scanned only when a project screen that uses the specified tag is open.
   - **Div**: For scaling — the value is divided by this number when reading from the server, and it is multiplied by this number when writing to the server.
   - **Add**: For scaling — this number is added to the value when reading from the server, and it is subtracted from the value when writing to the server.
   - **Instance ID**: The instance ID is automatically generated from the full path of the server item.

12. Save and close the worksheet.

### Enabling the OPC .NET Client task

To enable the OPC .NET Client task for runtime:

1. On the Home tab of the ribbon, in either the Local Management or Remote Management group (depending on where the project server will be), click **Tasks**.
   The **Execution Tasks** dialog is displayed.

2. In the list of tasks, select **OPC .NET Client**.

3. Click **Startup**.
   The **Startup** dialog is displayed.

4. Select **Automatic**, and then click **OK**.

5. Click **OK** again to close the **Execution Tasks** dialog.
Configuring an OPC XML/DA Client connection to an OPC XML-DA server

The OPC XML/DA Client task/worksheet is used to communicate with any system that implements the OPC XML-DA protocol.

About OPC XML/DA

The OPC XML/DA Client task/worksheet uses the new OPC XML-DA communication protocol that was introduced by the OPC Foundation. OPC XML-DA is an improvement on OPC DA (also known as OPC Classic): it is based on the XML, SOAP, and WSDL standards for web services, instead of the original DCOM/OLE model that is used by OPC DA. It also standardizes the SOAP messages exchanged between clients and server, which allows the protocol to be implemented on different operating systems.

For more information, go to the OPC Foundation website.

Configuring a new OPC XML/DA server connection

To configure a new connection to an OPC XML/DA server:

1. Do one of the following:
   • On the Insert tab of the ribbon, in the Communication group, click OPC Client and then select OPC XML/DA Connection from the drop-down list; or
   • In the Comm tab of the Project Explorer, right-click OPC XML/DA Connections and then click Insert on the shortcut menu.

   The Server Connection dialog is displayed:

2. In the Connection Name box, type a name for the connection.

   This name will be displayed in the OPC XML/DA Connections folder in the Project Explorer, and it is the name you will look for when you configure the OPC XML/DA Client worksheet.

3. From the Specification list, select the exact OPC specification that you want to use to communicate with the server.

4. From the Host Name list, select the name or address of the host to which you want to connect. Hosts should broadcast their availability on the network. If the host you want doesn’t appear to be available, click Refresh to update the list.

5. From the Server Url list, select a specific OPC server process on the host. If the process you want doesn’t appear, click Refresh the update the list from the host.

6. In the User Name and Password boxes, type your login credentials for the OPC server process.

7. In the Ping Rate box, type the frequency (in milliseconds) at which the client should ping the server to make sure the connection is still active. By default, the client pings the server once per minute.

8. Click OK. The connection is saved in the OPC XML/DA Connections folder in the Project Explorer.
Configuring a new OPC XML/DA Client worksheet
To configure a new OPC XML/DA Client worksheet:

1. Do one of the following:
   - On the Insert tab of the ribbon, in the Communication group, click **OPC Client** and then select **OPC XML/DA Client** from the drop-down list; or
   - In the Comm tab of the Project Explorer, right-click **OPC XML/DA** and then click **Insert** on the shortcut menu.

   A new OPC XML/DA Client worksheet is displayed:

   ![OPC XML/DA Client worksheet](image)

   **OPC XML/DA Client worksheet**

   2. In the **Description** box, type a description of the worksheet. This is for documentation purposes only and does not affect the execution of the worksheet.

   3. In the **Connection** list, select the server connection that you configured earlier.

   4. For more connection options, click **Advanced**.
The *Advanced* dialog box is displayed.

![Advanced dialog box]

Review the settings and configure as needed:

**Read actions**

*Enable subscription*

When this value is TRUE (non-zero), the client will subscribe to the server nodes so that it can receive notifications when the node values change.

**Note**: This is enabled by default. If it is disabled, you must use read triggers (see below).

*Maximum group size*

The maximum number of server nodes that may be read in a single read operation.

For example, if you have 1000 items/rows configured in the worksheet and *Maximum group size* is set to 100, 10 read operations will be performed when a read is triggered.

*Synchronous read trigger*

When the value of this tag/expression changes, the client reads all of the node values from the server. The read operation is performed synchronously, which means all other client operations are blocked until the read operation is finished. When the read operation is finished, the client increments the tag configured in *Read count*.

*Asynchronous read trigger*

The same as *Synchronous read trigger* except that the read operation is performed asynchronously, which means that other client operations may continue while the read operation is being performed.

*Read count*

The name of a project tag (Integer type) that will receive a count of the number of read operations performed since the project was run.

*Status*
The name of a project tag (Integer type) that will receive a status code for the last read operation performed by a trigger. If the status code is 0, the read operation finished successfully. For all other status codes, see List of read/write status codes and messages for OPC UA on page 429.

**Status message**
The name of a project tag (String type) that will receive the corresponding status message.

**Maximum aging**
The maximum age (in milliseconds) of values that will be accepted from the server's cache. If a value is older than this, the server will be forced to get the latest value from the field device.

**Sampling rate**
When this option is selected, you can change the rate (in milliseconds) at which the OPC server reads from field devices. By default, the sampling rate is half the publishing rate.

**Write actions**

**Enable write on tag change**
When this value is TRUE (non-zero), an asynchronous write operation will be performed automatically whenever the value of a project tag changes. All tag changes that occurred since the last scan of the worksheet will be written in a single write operation, or in multiple write operations if the number of tag changes exceeds the maximum group size.

| Note: This is enabled by default. If it is disabled, you must use write triggers (see below). |

**Maximum group size**
The maximum number of tag changes that may be written in a single write operation.

For example, if you have 1000 items/rows configured in the worksheet and Maximum group size is set to 100, 10 write operations will be performed during each scan of the worksheet.

**Synchronous write trigger**
When the value of this tag/expression changes, the client writes all of the tag values to the server. The write operation is performed synchronously, which means all other client operations are blocked until the write operation is finished. When the write operation is finished, the client increments the tag configured in Write count.

**Asynchronous write trigger**
The same as Synchronous write trigger except that the write operation is performed asynchronously, which means that other client operations may continue while the write operation is being performed.

**Write count**
The name of a project tag (Integer type) that will receive a count of the number of write operations performed since the project was run.

**Status**
The name of a project tag (Integer type) that will receive a status code for the last write operation performed by a trigger. If the status code is 0, the write operation finished successfully. For all other status codes, see List of read/write status codes and messages for OPC UA on page 429.

**Status message**
The name of a project tag (String type) that will receive the corresponding status message.

**Reload trigger**
The name of a project tag (Boolean, Integer, or Real type) that can be used as a trigger. When the value of this tag changes, the worksheet is reloaded. Tags in curly brackets (e.g., \{tagname\}) configured in the body of the worksheet are refreshed only when the worksheet is reloaded. After the worksheet is reloaded, the value of this tag will be reset to 0.

**Refresh IDs on startup**
When this option is selected, the node IDs in the worksheet will be refreshed from the specified paths every time the project is run.
Note: Refreshing IDs like this may cause the project to take longer to start up, so if you select this option, you should also specify a root node (see below) to limit how much of the server’s list of items must scanned.

Ensure cache synchronization
When this option is selected, the client will wait after each write operation for confirmation from the server that the node values actually changed. If the client does not receive confirmation, it will restore the previous tag values.

Enable bit notation
When this option is selected, bit notation is allowed in the Item column of the worksheet body.

Note: IWS’s OPC client does not normally use the Triggering Mode that is defined the OPC protocol. Instead, it allows any change in any tag/expression to be used as a trigger. If you want to use Triggering Mode, configure one worksheet to read the trigger values, and then configure another worksheet that specifies the read values as triggers.

5. Click OK to close the Advanced dialog box and return to the worksheet.
6. In the Status box, type the name of a project tag (Integer type) that will receive connection status codes during project runtime.
7. In the Status Message box, type the name of a project tag (String type) that will receive the corresponding status messages.
8. In the Publish Rate box, type the frequency (in milliseconds) at which the client will request updates from the server.
9. In the Disable box, type a tag/expression. When the value is TRUE (non-zero), the worksheet will not be executed.
10. In the Percent Deadband box, you may type a value between 0.0 and 100.0, as a percentage of the full engineering units scale. (The scale is calculated using the specified minimum and maximum values of the server item.) This tells the server to publish only changes in item values that are greater than the specified deadband. For example, if the minimum value is 1000, the maximum value is 5000, and the deadband is 0.1, then only changes greater than 4 (i.e., 0.1% of 4000) will be published by the server.
If you do not specify a deadband, then the default is 0.0, which means the server will publish all changes in item values.

Note: Percent Deadband only applies to server items that have the Engineering Units Type attribute (dwEUType) set to Analog (1). For more information about this attribute, see either the OPC specification or the documentation for the OPC server.

Caution: If Ensure cache synchronization (in the advanced settings) is selected, then Percent Deadband should not be used.

11. In the Root node or view box, specify the server node that will serve as the root for all items in the worksheet body. Specifying a root node makes it easier to find items and improves runtime performance.
12. For each row of the worksheet body, specify the following:
   • Tag Name: The name of a project tag.
     You may use indirect tags (e.g., {tagname}), but if you do, then be sure to configure the Reload trigger option above.
   • Item: The server item with which the project tag should be associated.
     To browse the server’s list of items, right-click in this field and then click Browse on the shortcut menu.

Note: If Enable bit notation is selected in the advanced settings, you can select a specific bit of a server node value by appending the bit number to the item name (e.g., nodepath.bit). Please note that this is supported only for unsigned variables and 32-bit signed variables; if you
try to do it with 16-bit or 8-bit signed variables, the most significant bit (MSB) will not work properly.

- **Scan**: If *Always*, then the row is continuously scanned during runtime. If *Screen*, then the row is scanned only when a project screen that uses the specified tag is open.

- **Div**: For scaling — the value is divided by this number when reading from the server, and it is multiplied by this number when writing to the server.

- **Add**: For scaling — this number is added to the value when reading from the server, and it is subtracted from the value when writing to the server.

- **Array Index**: If the server item is an array, then the array index with which the project tag should be associated.

13. Save and close the worksheet.

**Enabling the OPC XML/DA Client task**

To enable the OPC XML/DA Client task for runtime:

1. On the Home tab of the ribbon, in either the Local Management or Remote Management group (depending on where the project server will be), click **Tasks**.

   The *Execution Tasks* dialog is displayed.

2. In the list of tasks, select **OPC XML/DA Client Runtime**.

3. Click **Startup**.

   The *Startup* dialog is displayed.

4. Select **Automatic**, and then click **OK**.

5. Click **OK** again to close the *Execution Tasks* dialog.

**Note:** The OPC XML/DA specification supports custom item qualities using the high byte of the two-byte quality field. However, such qualities are often vendor-specific or even hand-coded, so it is not possible for InduSoft Web Studio to interpret them. All item qualities other than GOOD (192) will be ignored.
Configuring a TCP/IP Client connection to another project

The TCP/IP Client task/worksheet is used to communicate with another IWS project server. The TCP/IP Client Runtime and TCP/IP Server Runtime tasks enable two or more projects to keep their databases synchronized. These tasks use the TCP/IP protocol to provide communication between projects. Before using the tasks, you must make sure that TCP/IP (Ethernet) communication is properly configured and running on both servers.

- To configure the server: You do not have to configure anything on the server itself. You just have to run the TCP/IP Server Runtime task. You can choose to run it automatically on start up, or run the task manually by clicking Tasks (either local or remote) on the Home tab of the ribbon. After you start the task, a small icon displays in your system tray.

- To stop the TCP/IP Server Runtime task: Right-click the TCP/IP Server icon in the system tray, and then click Exit on the shortcut menu.

- To configure the client: You must use the TCP/IP Client worksheet to specify the server IP address and the tags you want to share with the server.

The TCP/IP Client worksheet is located on the Comm tab of the Project Explorer, and it uses the same commands as the Driver worksheet.

TCP/IP Client worksheet

Use the following parameters to complete the TCP/IP Client configuration:

- **Description** box: Type a description of the TCP/IP Client worksheet, for documentation purposes only. The TCP/IP Client task ignores this information.

- **Connection Status** box: Type a tag name and the TCP/IP Client Configuration task will update this tag according to its connection status. A tag value of zero indicates the connection is okay. Any other tag value indicates an error code returned by the Windows Socket library.

- **Disable** box: Type a tag name in this field. When this tag has any value other than 0, this TCP/IP worksheet will be disabled. Using this field, you can enable/disable the TCP/IP Client worksheet during runtime.

- **Server IP Address** box: Type the IP address and Port (optional) of the target server — for example, 169.254.182.158:123. The Port should be the same on both the Client and Server stations.

You can also specify a String tag enclosed in curly brackets (e.g., \{tagname\}) if you want to dynamically change this address during runtime.

- **Send Fields** option: When this option is selected, the tag properties/fields (i.e., Min, Max, Ack, Unit, LoLoLimit, LoLimit, HiLimit, HiHiLimit, RateLimit, DevSetPoint, DevPLimit, and DevMLimit) are sent with the tag values to the specified server. When this option is cleared, only the tag values (including TimeStamp and Quality, which are required) are sent.

**Note:** It is possible to add or remove fields in the list of fields sent. For more information, contact Support.
• **Read Only** option: When this option is selected, all communication is one-way and no tag values are written back to the specified server. This is useful when you only need to use the TCP/IP Client to retrieve data from other projects, and it can improve runtime safety and stability.

• **Send Values On Connection** box: When this option is selected and the project is run, the client will ignore the first tag values that it receives from the specified server and instead send its own tag values to the server.

• **Tag Name** field: Type the tags you want to share with the server.

  If the tag is an array or a class (or both), the project automatically enables every array position and class member for TCP/IP communication by default.

  To configure a specific array position and/or a specific class member, type the array position and/or class member in square brackets following the tag name. For example, `level[3].member`.

• **Remote Tag** field (optional): Type the name of a tag to be linked with the tag you specified in the **Tag Name** field. If you leave this field blank, the project uses the same tag name used in the client and in the server.

  ![Note]

  If you need to share an array, the tag in the server should contain the same number of elements as the tag in the client. If the tag is a class, the class definition should be the same in both server and client programs. If you do not follow these rules, unpredictable results can occur.

Only the Client task uses the `ConnectionRetryTimeout` parameter.
Configuring a DDE Client connection to a DDE Server

The DDE Client task/worksheet is used to configure a DDE Client connection to a DDE Server application such as Microsoft Excel (or any other Windows program supporting this interface).

Dynamic Data Exchange (DDE) is a protocol for dynamic data exchange between Windows applications, such as Excel. A DDE conversation is an interaction between server and client programs. IWS provides interfaces that run as clients or as servers. See DDE Client Runtime and DDE Server in the Runtime Tasks (Tasks on the Home tab of the ribbon).

- To run as a server, start the DDE or NetDDE server task as described in Runtime Tasks.
- To run as a client, configure the DDE interface worksheet on the Comm tab.

Network Dynamic Data Exchange (NetDDE) is an extension of DDE that works across computers on a network.

- To run IWS as a server to a NetDDE connection, you must start the DDE Server application.
- To run IWS as a client to a NetDDE connection, use the same DDE interface worksheets with the proper configuration to address a IWS project.

Note: When running NetDDE, IWS accepts the WRITE triggers only. To read data, you must configure a write command on the server computer.

To open a new DDE worksheet, right-click on the DDE folder and click the prompt screen. A new DDE worksheet displays, as in the following figure.

![DDE Worksheet](image)

The DDE worksheet dialog is divided into two areas:

- **Header area** (top section), contains information for the whole group and defines the tags to start the reading and writing and to receive connection status
- **Body area** (bottom section), where you define tags in the project and items related to the DDE server application

Every DDE interface is based on addressing an application using the following three structures:

- Application Name
- Topic
- Item

The first task is to find these identifiers in the DDE Server application.

Use the parameters in the DDE client worksheet Header area as follows:
Communication with Other Devices

- **Description** field: Type a description of the DDE worksheet for documentation purposes.
- **Application Name** field: Type the DDE server application name.
- **Topic** field: Specify a topic in the server application.
- **Connect** field: Type a tag to control the connection of the IWS DDE client and DDE server application. When this tag is set to 1, it requests a connection to the server. If the connection is not possible or if it fails, IWS sets the tag to zero again. If the connection is OK, this value remains set to 1.
- **Read Trigger** field: Type a tag to command a reading of the table. When this tag changes value, IWS generates polling to the DDE server. You can use this parameter with local DDE only; you cannot use it with NetDDE servers.
- **Enable Read when Idle** field: Type a tag value higher than zero to enable a reading of the equipment.
- **Read Status** field: Contains the status of the reading command.
- **Write Trigger** field: Type a tag enabling IWS to generate poke commands to the server.
- **Enable Write on Tag Change** field: Type a tag value higher than zero to enable the communication driver to check continuously for changes in a tag value in the worksheet. When the driver detects a change occurs, it writes the changed tag on the equipment, along with the tag's address.
- **Write Status** field: Contains the status of the writing command.

Use the DDE client Body area parameters as follows:

- **Tag Name** field: Type a tag to read or write the IWS database from the DDE server application.
- **Item** field: Type the ITEM part of the DDE address on the server. Refer to your server software documentation for information about the proper syntax for **APP**, **TOPIC**, and **ITEM**.

You can configure the **Topic** and **Item** fields with tags on the address using the syntax: `text{tag}`. IWS evaluates the value of `{tag}` to a string and uses it on the address. For example:

- **Topic**: `topic_{tag_topic_name}_example`
- **Item**: `{tag_item_name}` or `A{tag_number}`

Configuring a NetDDE connection is similar to configuring a DDE connection, except for the Header Application name and topic. Before starting your tests, verify that you enable the DDE Server on the station with which you want to exchange data.

**Note:** When connecting to servers other than IWS, please refer to the server documentation for information about the proper syntax of **APP**, **TOPIC**, and **ITEM**.

Use the NetDDE Client worksheet Header parameters to define the tags that start reading and writing, and tags that receive the connection status, as follows:

- **Application Name** field: Type `computer name\NDDE$`, where `computer name` must be a valid network computer name.
- **Topic** field: Type the **UNISOFT$** topic name to connect to another IWS station.

Use the DDE client worksheet Body parameters to relate each tag to each ITEM part of the DDE server address, as follows:

- **Tag Name** field: Specify the IWS local database tag name that is related to some remote tag name.
- **Item** field: Specify the remote tag name that is related to the local tag name.

**Note:** By default, the DDE Client module supports DDE Servers that handle string data in the UNICODE format. If the DDE Server handles string data in the ASCII ANSI format, the following setting must be configured manually in the `project_name.app` file (you can use Notepad to edit this file):

```
[Options]
DDEANSI=1
```
Sort or filter the rows in a worksheet

Sort or filter the rows in a worksheet in order to make it easier to browse the rows or find a specific item. Before you begin this task, you must have already inserted a worksheet and opened it for editing. You should also be familiar with how sorting and filtering is done in general-purpose spreadsheet applications. Please note that you can sort or filter rows only in the following types of worksheets:

- The Project Tags, Shared Tags, and System Tags datasheets;
- The Translation Table worksheet;
- All task worksheets except Report and Script, which do not have rows; and
- All communication worksheets.

None of the other worksheets have rows to sort or filter.

Sorting is done alphanumerically, by the selected column, in either ascending (0–9, A–Z) or descending (Z–A, 9–0) order.
Filtering is done according to whatever string you enter in the selected column. Only the rows that match the string will be displayed.

![Alarm worksheet rows filtered where Tag Name is "Tag3"](image)

![Alarm worksheet rows filtered where Type is "Lo"](image)

**Tip:** You can still delete rows while they are sorted or filtered.

To sort or filter rows:

1. To sort the rows, click the header of the column by which you want to sort. Click once to sort in ascending order, and then click again to sort in descending order.

   The current order (i.e., the direction of the sort) is indicated by the arrow to the right of the column name.

   **Note:** You cannot sort by multiple columns.

2. To undo the sorting and restore the rows to their original order, click the header of the first (numbered) column.

3. To filter the rows, type the string that you want to match in the top (zero) row of the worksheet and then press either **Tab** or **Return**.

   You may include * and ? as wildcard characters in your string:

   - * matches any number of characters, including none. For example, Tag* would match Tag, Tag3, Tag34567, TagA, and Tag_TEMP.
   - ? matches exactly one character. For example, Tag? matches Tag3 and TagA, while Tag?????? matches Tag34567 and Tag_TEMP.

   Also, you may filter by multiple columns. Only the rows that match the filter strings in all columns will be displayed.

4. To undo the filtering and restore the rows to their original order, delete the string that you typed and then press either **Tab** or **Return**.

Please keep in mind that sorting or filtering the rows of a worksheet only helps you to edit that worksheet. It does not change how the worksheet is executed during run time. The rows will be executed in their original numbered order (i.e., the leftmost column) unless you actually move or delete a row.
InduSoft Web Studio has built-in support for Microsoft Visual Studio Team Foundation Server 2010, which allows workgroup collaboration and source control from within the development environment.

**Note:** You must specifically select the Collaboration feature when you install the InduSoft Web Studio software, and to use the feature, your software license must have the Collaboration add-on. For more information, see Install the full InduSoft Web Studio software on page 37.

Collaboration helps to integrate IWS project development into a broader strategy of application lifecycle management (ALM). Development is more than just writing code — you also need to determine business requirements, create work items that describe those requirements, assign the work items to individual developers, track the completion of the work items, test the submitted code, and collect bugs and user requests for additional development. This can all be done with Team Foundation Server.

For more information about ALM and how to implement it with Team Foundation Server, go to:

- [msdn.microsoft.com/library/fda2bad5](http://msdn.microsoft.com/library/fda2bad5)

When the Collaboration feature is properly installed and licensed, it effectively makes the InduSoft Web Studio development application a Team Foundation Server client with many of the same commands. As such, before you use the Collaboration feature, you should be familiar with basic source control concepts in Team Foundation Server, such as:

- Connecting to servers, creating workspaces, and mapping folders;
- Adding files to source control;
- Checking files out of source control for editing;
- Viewing pending changes and then checking them into source control;
- Applying labels and getting specific versions; and
- Branching and merging.

Once you have enabled the Collaboration feature in InduSoft Web Studio, you can add your IWS projects directly to source control in Team Foundation Server.
Connect to your collaboration server

Connect to your collaboration server in order to make all other collaboration commands available.

Before you begin this task, you must have:

• Microsoft Visual Studio Team Foundation Server 2010 or 2012 running somewhere on your network, with a team project already created and configured to allow you to access it;

• Microsoft Visual Studio Team Explorer 2010 installed on the same computer as the InduSoft Web Studio development application; and

• The Collaboration feature installed and licensed with the InduSoft Web Studio development application.

For more information, see Install the full InduSoft Web Studio software on page 37.

Note: Team Foundation Server 2013 is not supported at this time for the collaboration server.

Note: Team Explorer 2012 and Team Explorer 2013 are not supported at this time for the collaboration client.

To connect to your collaboration server:

1. On the Project tab of the ribbon, in the Settings group, click Collaboration.

   The Collaboration Server dialog box is displayed.

   ![Collaboration Server dialog box]

2. In the Server area at the top of the dialog box, click Browse.

   The Connect to Team Project dialog box is displayed.

   Note: This dialog box is the same as in Team Explorer, and it should show your existing Team Explorer server connection(s).

3. If you have already configured your server connection in Team Explorer, select it from the Team Foundation Server list. Otherwise, do the following:

   a) To the right of the Team Foundation Server list, click Servers.

      The Add/Remove Team Foundation Server dialog box is displayed.

   b) Click Add.
The **Add Team Foundation Server** dialog box is displayed.

c) Specify the address and connection details for your Team Foundation Server, and then click **OK**.

> **Tip:** You can get this information from your network or server administrator.


d) Enter your user name and password for the server, and then click **OK**.

Remember to include your Windows network domain, if necessary. The server connection is saved, and the **Add Team Foundation Server** dialog box is closed.

e) In the **Add/Remove Team Foundation Server** dialog box, click **Close**.

The team projects that reside on the selected server are displayed.

4. In the **Team Projects** list, select the project where you want to save your IWS projects, and then click **Connect**. In the **Collaboration Server** dialog box, the server connection and team project are displayed, and at the bottom, the status message changes to "Authenticated to server. Workspace created but unmapped."

5. If you have already configured your workspace in Team Explorer, select it from the **Workspace Name** list. Otherwise, do the following:

   a) In the **Workspace Name** box, either accept the automatically generated name or type a new name.

   b) In the **Source Control Folder** box, either accept the automatically generated path or type a new path.

   c) To the right of the **Local Folder** box, click **Browse**. A standard Windows folder browser is displayed.

   d) Locate the folder on your computer that you want to map to the source control folder, and then click **OK**.

   In most cases, you should map the folder that contains your IWS projects (e.g., C:\Users \username\Documents\InduSoft Web Studio v7.1 Projects\).

   In the **Collaboration Server** dialog box, the local folder is mapped and the status message at the bottom changes to "Connected to server. Workspace mapped."

   > **Note:** The workspace that is created here should also be available in Team Explorer.

6. Click **Close** to close the **Collaboration Server** dialog box.

You are now connected to Team Foundation Server and can add your IWS project files to source control. The **Collaboration** command on the ribbon will remain active to show that you are connected.

The collaboration settings are saved with the project, so in the future, whenever you open a project that has been added to source control, you will be prompted to log on to the server.
Add a project to source control

Add a project to source control in order to begin workgroup collaboration and version tracking for that project. To perform this task, you must be connected to your collaboration server.

After you add a project, there are two changes to how the individual project files are handled. First, when you insert a new screen or task worksheet into your project, the worksheet file is automatically added to source control. Second, when you open a screen or task worksheet for editing, the worksheet is automatically checked out of source control.

You can add a new project to source control when you actually create the project, assuming you are connected to the collaboration server at that moment. For more information, see Creating a new project on page 90.

Otherwise, to add a project to source control:
1. Save and close all open worksheets.
2. Right-click the project root (e.g., Project: projectname.APP) in the Project Explorer, and then click Add Project to Collaboration Server on the shortcut menu.

Files added to source control are marked with a green "add" badge.

Added files are considered to be checked out for editing, and they still need to be checked in to make any changes permanent.

Also, remember that if you added your entire project to source control, then each time you insert a new screen or task worksheet, it will be automatically added to source control with the rest of your project.
Check out project files for editing

Check project files out of source control in order to edit them.
To perform this task, you must be connected to your collaboration server.
If you open a file and make any change, then it is automatically checked out.
Otherwise, to manually check out a folder or file:

In the Project Explorer, right-click the folder or file that you want to check out, and then click **Check Out for Edit** on the shortcut menu.
If you right-click a folder (e.g., Screens), then you will check out all of the files in that folder.
Files checked out for editing are marked with a green "check" badge.

"Check Out for Edit" badge

These files must be checked in to make any changes permanent.
View, undo, or check in pending changes

View pending changes on project files, and then choose whether to undo the changes or check them into source control.

To perform this task, you must be connected to your collaboration server.

When you check out project files and edit them, the editing that you do is cached locally as "pending changes". These changes are not checked into source control until you explicitly choose to do so. You can also choose to undo changes on a project file and revert it to its most recently checked-in version.

To view pending changes:

1. In the Project Explorer, right-click the folder or file for which you want to view changes, and then click View Pending Changes on the shortcut menu.

   Please note that you will only be able to view changes in the location that you right-clicked. For example, if you right-click the Screens folder or a screen file (*.scr) in that folder, then you will only see pending changes to the screen files. If you want to view all pending changes for the entire project, then you should right-click the project root (e.g., Project: projectname.APP).

   The Pending Changes dialog box is displayed, and the project files with pending changes are listed at the top.

   ![Pending Changes dialog box]

2. If you do not want to do anything with the pending changes at this time, click Cancel at the bottom. The Pending Changes dialog box is closed.

3. To undo pending changes:
   a) Make sure only the changes that you want to undo are selected. (If you want to undo all changes, click Select All Changes.)

   b) Click Undo Changes at the bottom.

   The Pending Changes dialog box is closed, and the selected files are reverted to their most recently checked-in versions. In the Project Explorer, you will see that the files no longer have either the "add" badge or the "check" badge.

4. To check in pending changes:
   a) Make sure only the changes that you want to check in are selected. (If you want to check in all changes, click Select All Changes.)

   b) In the Comment box, type a comment that generally describes the changes you are checking in. This comment will be saved with the changeset in source control.
c) In the Work Items list, select the work item(s) with which the changes should be associated.

d) Click Check In.

The Pending Changes dialog box is closed, and the selected files are checked into source control. In the Project Explorer, you will see that the files no longer have either the "add" badge or the "check" badge. Please note that when you right-click files in the Project Explorer, there are two other choices on the shortcut menu: Check In and Undo Pending Change. If you click either of these, the same Pending Changes dialog box is displayed but with certain options pre-selected. Everything else is the same as described in the steps above.
Branch a project in source control

Branch a project in source control in order to create a copy of the project that can be developed separate from the original.

To perform this task, you must be connected to your collaboration server.

To branch a project in source control:

1. Save and close all open worksheets.
2. Either check in or undo any pending changes.
3. Right-click the project root (e.g., `Project: projectname.APP`) in the Project Explorer, and then click **Branch** on the shortcut menu.
4. On the **Branch** page, in the **Target** box, type a new name for the branch, and then click **OK**.

The branch is created in source control, but the new project is not yet saved on your computer.

To save the new branch on your computer, open it from source control.
Open a project from source control

Open a project from source control in order to download it to your computer.

To perform this task, you must be connected to your collaboration server.

Since the point of collaboration is to collaborate with the other developers in your team, you may occasionally need to get projects that they have added to source control. You may also need to get a new branch that you have created.

Note: This only applies to projects that you are opening for the first time; if you have opened a project before, then it is already on your computer and you can open it normally.

To open a project from source control:

1. Click the Application button (at the top-left of the development environment), and then on the Application menu, click Open Project > Open Project from Collaboration. The Select Project page is displayed, listing all of the IWS projects in the Team Foundation Server team project to which you are connected.

2. Select a project from the list, and then click OK.

The selected project is downloaded to your computer, added to your Team Explorer workspace, and then opened for editing.
Get the latest version of a project

Get the latest version of a project in order to download any changes that were made by other developers since the last time you worked on the project.

To perform this task, you must be connected to your collaboration server.

To get the latest version of a project:

1. Close all open worksheets.
2. Right-click the project root (e.g., Project: projectname.APP) in the Project Explorer, and then click Get Latest Version on the shortcut menu.

The latest version of the project is downloaded from source control and saved on your computer. It overwrites whatever you had previously saved, even if it was different from what is in source control.
Get a specific version of a project

Get a specific version of a project in order to roll back changes that you have made to the project files.

To perform this task, you must be connected to your collaboration server.

To get a specific version of a project:

1. Close all open worksheets.
2. Right-click the project root (e.g., Project: projectname.APP) in the Project Explorer, and then click Get Specific Version on the shortcut menu.
   The Get page is displayed.
3. In the list of project files, select the files for which you want to get a specific version.
4. In the Version area, do one of the following:
   • In the Type list, select Changeset to get a specific changeset, and then in the Changeset box, type the number of the changeset. (Detailed information about the changesets is available in Team Explorer.)
   • In the Type list, select Latest Version to get the latest version. This is the same as using the Get Latest Version command.
   • In the Type list, select Label to get a "snapshot" version that was previously created, and then in the Label list, select the name of the version.
5. Click Get.

The specified version of the project is downloaded from source control and saved on your computer.

Apply a label to a project

Apply a label to a project in source control in order to create a "snapshot" of the project that can be retrieved later.

To perform this task, you must be connected to your collaboration server.

Labels are used for version control within a project. When you attempt to get a specific version of a project, any labels that you have applied will be included in the list of available versions.

Applying a label to a project offers many of the same benefits as branching the project, but without having to create an entirely separate copy of the project.

To apply a label to a project:

1. Save and close all open worksheets.
2. Right-click the project root (e.g., Project: projectname.APP) in the Project Explorer, and then click Apply Label on the shortcut menu.
   The Apply Label page is displayed.
3. In the Name box, type a name for the label.
   This is what will appear in the list of available versions.
4. In the Comment box, type a description of the "snapshot".
   This is optional.
5. Click Create.

The "snapshot" is created in source control. To retrieve it later, get a specific version.
Project Security

InduSoft Web Studio includes a project security system that manages how users and user groups can access a project, during both development and runtime.
About security modes

In addition to managing users and groups locally, entirely within a single project, you can also get pre-defined users and groups from other IWS projects or from an LDAP-compliant domain server.

IWS supports four security modes:

**Local Only**
This is the standard mode for most projects: users and groups are created in the development application, and they apply only to the project for which they're created.

**Distributed – Server**
This is similar to Local Only, except that the project's security system configuration is also made available to other IWS projects (that are set to Distributed – Client) on the same network. Furthermore, if the project loses its security system configuration for some reason, then it can reimport the configuration from one of its client projects.

**Distributed – Client**
When this mode is selected, the project gets its entire security system configuration from another IWS project (that is set to Distributed – Server) on the same network. The project caches this configuration and can continue to run even if it loses communication with the server project.

**Domain (LDAP)**
The Lightweight Directory Access Protocol (LDAP) is a recognized standard for managing users and groups across many different applications on a network. When this mode is selected, the project gets its users and groups from an LDAP-compliant domain server, such as Microsoft Active Directory for Windows or OpenLDAP for Linux. However, only the user names, passwords, and group memberships are taken from the domain; specific rights must still be configured within the project.
About security access levels

Almost every item in a project — screen object, object animation, project screen, task worksheet — can be assigned a security access level. That access level determines which user groups can edit the item during development and/or use the item during runtime.

There are 255 possible access levels, allowing a large amount of granularity. Each user group is configured with ranges of levels for both development and runtime, and the groups’ ranges may overlap.

This means that for a user to be able to edit and/or use an item, the item’s access level must fall within the range specified for that user’s group.

For example, UserA of GroupA has a security access level range of 1–10 and UserB of GroupB has a security access level range of 5–15. To continue the example:

- Item #1 has Access Level = 1
- Item #2 has Access Level = 7
- Item #3 has Access Level = 12
- Item #4 has Access Level = 20

Consequently,

- Only UserA can access Item #1
- Both users can access Item #2
- Only UserB can access Item #3
- Neither user can access Item #4

**Note:** The default access level for all items is 0, and all users can use all items at that level.
Setting access levels for different types of items

For project screens, the access level can be set in the Screen Attributes dialog.

Security (access level) setting in Screen Attributes dialog

For screen objects and object animations, the access level can be set in the individual Object Properties dialog. (If the option is not available in the main dialog, then it should be available in one of the sub-dialogs.)

Security (access level) setting in Object Properties dialog

For task worksheets, do the following:

1. Open the worksheet for editing.
2. Click anywhere in the body of the worksheet. The Access Level control, on the Project tab of the ribbon, is enabled.
3. Click Access Level. The Security dialog is displayed.
4. In the Access Level box, type an access level for editing the worksheet.
5. Click OK.
Using the security system configuration wizard

The security system configuration wizard helps you through the steps of configuring the project security system.

1. Start the wizard.
   - If you are configuring the security system for the first time, then the wizard will start automatically when you do one of the following:
     - On the Project tab of the ribbon, in the Security tab, click **Configure**; or
     - In the Global tab of the Project Explorer, right-click Security and then click Settings on the shortcut menu.
   
   **Note:** After the first time, doing one of these will open the **Security System** dialog instead.
   
   - If you have already configured the security system, click **Configure** on the ribbon to open the **Security System** dialog and then click **Run Wizard**.

   The first page of the wizard is displayed.

   ![Security System Configuration Wizard](image)

   **Security System Configuration Wizard**

   This page always shows how the security system is currently configured.

2. Click **Next**.
   The second page of the wizard is displayed.

   ![Security System Configuration Wizard](image)

3. Select **Enable Security System**.

4. Click **Next**.
The third page of the wizard is displayed.

5. Select the **security mode**.
6. For **Distributed – Client** and **Domain (LDAP)**, click **Server Settings** and then configure the settings as needed.
7. Click **Next**.
   The fourth page of the wizard is displayed.

8. If you need to create or configure groups, click **Create/Configure Groups**.
   The **Group Account dialog** is displayed. When you’re done with that dialog, it will automatically return to the wizard.
9. Click **Next**.
   The fifth page of the wizard is displayed.

10. If you need to create or configure users, click **Create/Configure Users**.
    The **User Account dialog** is displayed. When you’re done with that dialog, it will automatically return to the wizard.
11. Click **Next**.
The sixth page of the wizard is displayed.

12. Review your configuration, and then click Finish to close the wizard.
Configuring server settings for security modes

If you set your project's security mode to either Distributed – Client or Domain (LDAP), then you must also configure server settings.

When you click Server Settings in either the security system configuration wizard or the main Security System dialog, the appropriate Server Settings dialog will be displayed.

Configuring the server settings for Security Mode: Distributed – Client

To configure the server settings:

1. In the Server IP and Server Port boxes, type the IP address of a runtime project that is set to Distributed – Server.
2. In the Connection timeout box, type the timeout (in seconds) after which the client will attempt to reconnect to the server. (A typical connection timeout is 3 seconds.)
3. In the Synchronization Period box, type the frequency (in seconds) at which the client will synchronize its security system configuration with the server's. (A typical synchronization period is 10 seconds.)
4. In the Force Cache Reload box, type the name of a project tag (Integer or Boolean type). If the tag value is TRUE (non-zero) and the specified server has a timestamp older than the client, then the local security system will be updated with outdated server information.
5. In the Status Tag box, type the name of a project tag (Integer type) that will receive a value indicating the status of the server connection.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No cache</td>
</tr>
<tr>
<td>1</td>
<td>Updated cache</td>
</tr>
<tr>
<td>2</td>
<td>Outdated local cache</td>
</tr>
<tr>
<td>3</td>
<td>Outdated server cache</td>
</tr>
<tr>
<td>4</td>
<td>Disconnected from server</td>
</tr>
</tbody>
</table>

While the project is running, the possible values are:

For more information, see GetSecuritySystemStatus.

6. Click OK.
Configuring the server settings for Security Mode: Domain (LDAP)

To configure the server settings:

1. In the **Domain** box, type the domain name of the LDAP server.

2. In the **User** and **Password** boxes, type your logon credentials for the LDAP server.

   **Note:** You must have sufficient privileges to get lists of groups and users. Please consult your LDAP administrator.

3. In the **Connection timeout** box, type the timeout (in seconds) after which the client will attempt to reconnect to the LDAP server. (A typical connection timeout is 4–5 seconds.)

4. In the **Retry interval** box, type the frequency (in seconds) at which the client will try to connect to the LDAP server if the connection could not be established.

5. In the **Status tag** box, type the name of a project tag (Integer type) that will receive a value indicating the status of the server connection.

   While the project is running, the possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Connection timeout</td>
</tr>
<tr>
<td>1</td>
<td>Bind timeout</td>
</tr>
<tr>
<td>2</td>
<td>Query timeout</td>
</tr>
<tr>
<td>3</td>
<td>Disconnected</td>
</tr>
<tr>
<td>4</td>
<td>Connected</td>
</tr>
<tr>
<td>5</td>
<td>No users or groups returned by query</td>
</tr>
<tr>
<td>6</td>
<td>Invalid user or group</td>
</tr>
</tbody>
</table>

   For more information, see **GetSecuritySystemStatus**.

6. Click **Check Connection** to confirm that the project can connect to the specified domain. If it cannot, review and correct your settings.

7. Click **OK**.
Note: The project security system will cache the most recent users in case the project loses its connection to the LDAP server. These users will be able to log onto the project regardless of LDAP server status. You can make the cache size unlimited by setting Cache size to 0, and you can make the cache never expire by setting Cache expiration to 0, but that is not recommended.

The users are cached in a FIFO (first in, first out) list, which means that when the most recent user is cached, the least recent user is removed. The exception to this is the Default User, who is always kept in the cache.

If you want to monitor the cache during project runtime, type the name of a project tag in the Hours until cache expiration box. This tag will receive a value equal to the number of hours until the cache expires.

If you want to allow users to log on with either local or server-defined accounts, select Mixed mode cache. Local users and groups can be created only when Security Mode is set to Local Only, but after they are created, they are kept in the cache when Security Mode is set to Domain (LDAP).

Finally, you can opt to receive trace messages (in the Output window) each time the cache is updated. To do this, manually edit your project file (projectname.APP) and add the following setting:

```
[Options]
EnableSecuritySystemTraceLdapOfflineCache=1
```

In the LDAP Advanced Settings tab of the dialog, there are additional settings that should be configured only by experienced LDAP administrators:

Server Address Manual Configuration

If for some reason the LDAP server cannot be accessed using its domain name, then you can manually configure the server’s IP address: select Enable, and then type the IP address and port number of the LDAP server.

Note: The default port for LDAP is 389. However, please verify the port number with the server administrator.

Allow simple bind (ADAM)
LDAP normally requires secure binding for authentication, but Active Directory Application Mode (ADAM) in Windows Server 2003 does not fully support secure binding. To allow simple binding with an ADAM server, select this option.

**Tip:** Simple binding means that user credentials are sent in clear text, so you should secure the connection by other means such as VPN, TLS/SSL, or proxies.

**Save Rights to Server**

By default, IWS security rights are saved entirely within your project. However, you can save those rights back to the LDAP server, either to make them available to other projects that use the same LDAP server or for simple redundancy.

To make this option work, you must first extend the server’s LDAP schema to contain additional information about the project security system. See [Extending the LDAP schema to allow saving of security rights](#).

Once that is done, click **Modify** to provide your LDAP server credentials and then select **Enable**.

In the *LDAP Query Customization* tab of the dialog, you can further customize how LDAP server queries are formed:

![LDAP Query Customization](image)

By default, the LDAP server provides a list of all registered users and groups, so in a large or complex network environment, that can result in an impractically long list to manage when you’re configuring your project security system. To restrict the list of users and groups, you may customize the LDAP query to eliminate anyone who should never have access to your project: click **Modify** to provide your LDAP server credentials, select **Enable**, and then configure the **Search Base** and **Filter Query** settings. For the proper syntax, consult the LDAP server documentation.

Also, some non-standard LDAP implementations — such as Linux-based LDAP servers and Active Directory Application Mode (ADAM) in Windows Server 2003 — use different entity identifiers and attributes. Those can be customized in this dialog, but again, it should only be done by an experienced LDAP administrator.

**Example of alternate attributes in ADAM**

<table>
<thead>
<tr>
<th>LDAP Server</th>
<th>User name attribute</th>
<th>Group name attribute</th>
<th>User lock attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Directory</td>
<td>sAMAccountName</td>
<td>sAMAccountName</td>
<td>userAccountControl</td>
</tr>
<tr>
<td>Active Directory Application Mode (ADAM)</td>
<td>Name</td>
<td>Name</td>
<td>userAccountControl</td>
</tr>
</tbody>
</table>
Extending the LDAP schema to allow saving of security rights

In order to save IWS project security rights back to a Domain (LDAP) server, the server's LDAP schema must be extended to contain the additional information.

The server must already be configured and running on your network, and you must have sufficient privileges to make changes to the server configuration.

In this procedure, you will create a new attribute called "proprietarySCADAInfo" to contain the IWS project security rights, and then you will add the attribute to the "person" and "group" classes in the server configuration. These classes correspond to users and groups in the project security system.

Please note this procedure only shows how to extend the schema in Microsoft Active Directory running on Windows Server 2003. The exact procedure is different for other LDAP servers and operating systems, but the basic steps should be essentially the same. Please consult your LDAP server documentation.

**Caution:** Extending a server's LDAP schema cannot be undone.

1. Register the schema management DLL.
   a) Click Start > All Programs > Accessories > Command Prompt.
      A Command Prompt window is displayed.
   b) At the prompt, type cd %SystemRoot%\System32 and then press Return.
      The working directory is changed.
   c) Type regsvr32 schmmgmt.dll and then press Return.
      If the DLL is successfully registered, then a confirmation message is displayed.
   d) Click OK to dismiss the message.
   e) Close the Command Prompt window.

2. Add the Active Directory Schema snap-in to the console root.
   a) Click Start > All Programs > Accessories > Run.
      A Run dialog is displayed.
   b) In the Open box, type mmc, and then click OK.

(If you have User Access Control (UAC) enabled, then you will be asked if you want to allow Microsoft Management Console to make changes. Click Yes.)

A console window is displayed.
c) In the console window, click **File > Add/Remove Snap-in.**

The *Add/Remove Snap-in* dialog is displayed.

![Console window showing Add/Remove Snap-in dialog](image)

**Figure:** Add/Remove Snap-in dialog

- The *Add/Remove Snap-in* dialog is displayed.
- In the *Snap-ins added to* list, select **Console Root**, and then click **Add**.
- The *Add Standalone Snap-in* dialog is displayed.

The *Add Standalone Snap-in* dialog is displayed.
e) In the list of available snap-ins, select **Active Directory Schema**, and then and click **Add**.

The snap-in is added to Console Root.

f) Click **OK** to close the *Add/Remove Snap-in* dialog.

3. Create the proprietarySCADAInfo attribute in the Active Directory Schema snap-in.
a) In the Console Root tree-view, expand Active Directory Schema.

b) Right-click Active Directory Schema > Attributes, and then click Create Attribute on the shortcut menu. A message is displayed explaining that your schema will be permanent changed.

c) Click Continue. A Create New Attribute dialog is displayed.

d) In the dialog, complete the fields as follows.
   • Common Name: proprietarySCADAInfo
   • LDAP Display Name: proprietarySCADAInfo
   • Unique X500 Object ID: 0.7.7777.77777777.777.7.7

   Note: An unique Object ID should be used.

   • Description: proprietarySCADAInfo
   • Syntax: Octet String
   • Minimum: 0
e) Click **OK** to close the dialog.

The **proprietarySCADAInfo** attribute is added to the list.

4. Add the **proprietarySCADAInfo** attribute to the **person** and **group** classes.
a) In the Console Root tree-view, select Active Directory Schema > Classes

![Console Screenshot]

b) In the list of classes, right-click person, and then click Properties on the shortcut menu. The Properties dialog is displayed.
c) In the dialog, click the **Attributes** tab.

![person Properties](attachment:personProperties.png)

The **Select Schema Object** dialog is displayed.

d) Click **Add**.

The **Select Schema Object** dialog is displayed.
e) In the list of schema objects, select **proprietarySCADAInfo**, and then click **OK**.

![Select Schema Object Dialog](image)

The attribute is added to the class properties.

f) Click **OK** to close the **Properties** dialog.

g) Repeat steps b through f for the **group** class.

5. In the **Console Root** tree-view, right-click **Active Directory Schema**, and then click **Refresh** on the shortcut menu.

6. Click **File > Exit** to close the console window.

7. Restart the server.
Group Account dialog

The Group Account dialog is used to create and delete user groups, as well as to configure the access privileges for a selected group.

Accessing the dialog

Assuming the project security system has already been enabled (i.e., you have used the security system configuration wizard at least once), then you can access this dialog by doing one of the following:

• Open the Security System dialog, and then click Groups; or
• In the Global tab of the Project Explorer, right-click Groups and then click Groups properties on the shortcut menu.

The dialog in detail

Please note that if a user is assigned to more than one group (see Creating and configuring users), then the groups’ settings may conflict with each other. How the settings are resolved depends on which settings they are:

• The settings in the Group Account dialog below are permissive — that is, the most permissive setting from all of a user’s groups applies to the user. For example, if any of the groups can create and modify tags, then the user can create and modify tags.
• The settings in the Group Account Advanced dialog (both tabs) are restrictive — that is, the most restrictive setting from all of a user’s groups applies to the user. For example, if one group has a minimum password size of 8 and another group has a minimum password size of 12, then the user’s minimum password size is 12. (For Auto Log Off in particular, Counting from logon overrides Counting from user’s last action.)

<table>
<thead>
<tr>
<th>Area / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Account</td>
<td>The user group that you are currently configuring.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> There are two default groups for all projects: Guest and (Default Rights).</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If the security mode is set to Domain (LDAP), then please note that the built-in groups in Microsoft Active Directory will not appear in this list of groups and cannot be added to the project.</td>
</tr>
<tr>
<td>New</td>
<td>Creates a new group. In the New Group Account dialog, type the name of the new group and then click OK.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the currently selected group.</td>
</tr>
<tr>
<td>Reset</td>
<td>Resets the privileges of the currently selected group to match the (Default Rights) group.</td>
</tr>
<tr>
<td>Area / Element</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Advanced</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Security Rights – Development</strong></td>
<td>Use Default Rights</td>
</tr>
<tr>
<td><strong>Security Level – Development</strong></td>
<td>Range</td>
</tr>
<tr>
<td><strong>Engineering Access</strong></td>
<td>Project Settings</td>
</tr>
<tr>
<td></td>
<td>Drivers, Data Sources</td>
</tr>
<tr>
<td></td>
<td>Network Configuration</td>
</tr>
<tr>
<td></td>
<td>Create, modify tags</td>
</tr>
<tr>
<td></td>
<td>Create, modify screens</td>
</tr>
<tr>
<td></td>
<td>Create, modify task sheets</td>
</tr>
<tr>
<td><strong>Security Rights – Runtime</strong></td>
<td>Use Default Rights</td>
</tr>
<tr>
<td><strong>Security Level – Runtime</strong></td>
<td>Range</td>
</tr>
<tr>
<td><strong>Runtime Access</strong></td>
<td>Start Project</td>
</tr>
<tr>
<td></td>
<td>Close Project</td>
</tr>
<tr>
<td></td>
<td>Database Spy (write)</td>
</tr>
<tr>
<td></td>
<td>Note: This only applies to projects running locally. For projects running remotely, see Enable Remote Debugging Tools below.</td>
</tr>
<tr>
<td></td>
<td>Task switch enabled</td>
</tr>
<tr>
<td></td>
<td>This option does not apply to Secure Viewer thin clients running on Windows XP 64-bit or any version of Windows Embedded; the user may always switch away from the client.</td>
</tr>
<tr>
<td></td>
<td>Edit Security System</td>
</tr>
<tr>
<td></td>
<td>Note: Be careful not to clear this option for your own group, or you may not be able to undo your own changes.</td>
</tr>
<tr>
<td></td>
<td>Enable Remote Debugging Tools</td>
</tr>
<tr>
<td></td>
<td>Windows Task Manager</td>
</tr>
<tr>
<td></td>
<td>This option does not apply to Secure Viewer thin clients running on Windows XP 64-bit or any version of Windows Embedded; the user may always open the Windows Task Manager.</td>
</tr>
<tr>
<td></td>
<td>Note: Clearing this option means disabling the Task Manager during run time, which requires Administrator privileges. You will need to run the client with elevated privileges.</td>
</tr>
<tr>
<td><strong>Runtime group</strong></td>
<td>If this option is selected, then the group can only include users that have the option Runtime user selected.</td>
</tr>
<tr>
<td><strong>Web Thin Client Access</strong></td>
<td>Members of the group may use a web thin client to connect to the project runtime server.</td>
</tr>
</tbody>
</table>
### Secure Viewer Access
Members of the group may use a Secure Viewer thin client to connect to the project runtime server.

### Advanced dialog – Password Options

<table>
<thead>
<tr>
<th>Area / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min password size</td>
<td>To make user passwords more complex and therefore more secure, you can require that they contain a certain number of alpha (A-Z, a-z), numeric (0-9), and special (punctuation) characters. When the user is prompted to change his password — for example, when his old password expires (see Password aging below) — the new password will not be accepted unless it meets these requirements.</td>
</tr>
<tr>
<td>Min number of special characters</td>
<td></td>
</tr>
<tr>
<td>Min number of numeric characters</td>
<td></td>
</tr>
<tr>
<td>Min number of alpha characters</td>
<td></td>
</tr>
<tr>
<td>Case-sensitive</td>
<td>If this option is selected, then passwords are case sensitive — that is, passwords created with both upper and lowercase characters must be entered the same way by the user.</td>
</tr>
</tbody>
</table>

**Note:** In projects created with InduSoft Web Studio v6.1+SP2 through v6.1+SP6, all passwords were case sensitive.

<table>
<thead>
<tr>
<th>Area / Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| Password aging               | Longevity (in days) of the password for all users in this group. After the specified number of days, the project will force the user to change his password: when the user tries to log in, the Change Password dialog is automatically displayed and the user cannot complete the logon process until he provides a new password. By default, the user must choose a new password that is different from the old password. To change this so that the user can re-use the same password, manually edit the project file (project_name.app) to include the following setting:  

```
[Security]
ChangePasswordMode=1
```
To make passwords never expire, set Password aging to 0.                                                                 |
| E-signature time-out         | Timeout period (in minutes) of the E-sign prompt for all users in this group. The user must enter his user name and password before the specified timeout to use project features that require an e-signature. |
| Disable e-signature          | When the value in this box is TRUE (non-zero), users in this group cannot use any project features that require an e-signature.                                                                           |
### Area / Element Description

- **You can configure a project tag in this box, so that e-signature is dynamically enabled/disabled during run time.**

---

**Advanced dialog – Auto LogOff/LockUp**

<table>
<thead>
<tr>
<th>Area / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto Log Off</strong></td>
<td><strong>Log Off after</strong> Number of minutes after which the current user must be logged off automatically. If this field is left in blank (or with the value 0), the current user is never logged off automatically.</td>
</tr>
<tr>
<td><strong>Counting from logon</strong></td>
<td>When this option is selected, the current user is automatically logged off after the period of time configured in the Log Off after field elapsed since when the current user was logged on the system.</td>
</tr>
<tr>
<td><strong>Counting from user's last action</strong></td>
<td>When this option is selected, the current user is automatically logged off after the period of time configured in the Log Off after field elapsed since the last action (mouse, touchscreen, or keyboard action) was performed by the current user.</td>
</tr>
<tr>
<td><strong>Disconnect Web Thin Client on auto logoff</strong></td>
<td>If the user logged on through a Web Thin Client, then when the user is logged off, the client is automatically disconnected from the data server.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Area / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto Lock-up</strong></td>
<td><strong>Enable</strong> Enables the auto lock-up features described below.</td>
</tr>
<tr>
<td><strong>Lock up account after</strong></td>
<td>Maximum number of times a user can try to log on to an account. If the user exceeds the specified maximum number of attempts (provides an invalid password) within the period of time specified in the Reset counter after field, the project will automatically block the user.</td>
</tr>
<tr>
<td><strong>Reset counter after</strong></td>
<td>Defines how long after an invalid log-on attempt the project will wait (in minutes) until it resets the log-on attempts counter.</td>
</tr>
</tbody>
</table>

**Note:** If VBScript debugging is enabled, then the **Auto Log Off** feature cannot be used; the normal execution cycle is suspended during debugging, so it is not possible to accurately measure the time elapsed without user input. For more information, see [Debugging VBScript](page 983) on page 983.
Creating and configuring users

To create and maintain accounts for project users, click the Users button on the Security System dialog. (Alternately, to configure a user, open the Users folder located in the Security folder.)

The User Account dialog displays.

After the project initializes, if no users log on (or when the current user logs off), then the project automatically logs on the default user (Guest). In addition to the default Guest user, there is a Guest group, which has default privileges that enable all tasks. We recommend that you evaluate and edit the Guest group’s privileges to specify a minimal amount of privileges for the start up procedure.

To create a new user, click New to open the New User Account dialog.

To delete a user, click the User combo-box button, select the user name from list, and then click Delete.

To configure a user, use the following procedure:

1. Click the User combo-box button and select a user from the list.
2. To block the user from logging onto the project at all, select User is blocked. This allows you to disable a user account without deleting it.

   **Tip:** To select and clear this option during run time, use the functions BlockUser and UnblockUser, respectively.

3. To restrict the user’s access so that they can log onto project only from project clients during run time, select Runtime user. Specifically, users that have this option selected can only be assigned to groups that have the option Runtime group selected. For more information, see Group Account dialog on page 482.

   Users that are created during run time, using the function CreateUser, have this option automatically selected.

4. Click the new Settings button to open the User Settings dialog:

5. Configure the parameters on this dialog as follows:
• **User Full Name** text box *(optional)*: Type the user's full name.
• **New Password** text box: Type the user’s password.
• **Confirm Password** text box: Re-type the user's password.

**Tip:** In most cases, user names and passwords can include spaces. However, if you plan to enable Mobile Access for your project, make sure the user names and passwords do not include spaces. For more information, see [Link directly to a project screen in the Mobile Access web interface](#) on page 594.

6. In the **Available Groups** list, select the group(s) to which the user should be assigned, and then click > to move those group(s) to the **Assigned Groups** list.

7. When you are finished, click **OK** to apply the changes and close the **Settings** dialog.
Security System dialog

The main Security System dialog is used to manage the project security system after it has been initially configured.

Accessing the dialog

Assuming the project security system has already been enabled (i.e., you have used the security system configuration wizard at least once), then you can access this dialog by doing one of the following:

• On the Project tab of the ribbon, in the Security group, click Configure; or
• In the Global tab of the Project Explorer, right-click Security and then click Settings on the shortcut menu.

If you do either of these before the security system has been enabled, then the security system configuration wizard will open automatically.

If you’ve already configured the security system and set a main password, then you’ll be prompted to enter it.

The dialog in detail

<table>
<thead>
<tr>
<th>Area / Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Security System</td>
<td>Indicates whether the project security system is currently enabled. If it is, then the users and groups’ specified access privileges are enforced.</td>
</tr>
<tr>
<td>Main Password</td>
<td>Opens a dialog where you can specify a main administrative password for the entire project.</td>
</tr>
<tr>
<td>Security Mode</td>
<td>Mode: The current security mode of the project.</td>
</tr>
<tr>
<td>Server Settings</td>
<td>Opens the Server Settings dialog, where you can configure the server settings for Distributed – Client or Domain (LDAP).</td>
</tr>
<tr>
<td>Run Wizard</td>
<td>Opens the security system configuration wizard.</td>
</tr>
<tr>
<td>Backup</td>
<td>Opens the Import/Export dialog, where you can export or import the security system configuration.</td>
</tr>
<tr>
<td>Accounts Management</td>
<td>Groups: Opens the Group Account dialog, where you can create and configure groups.</td>
</tr>
<tr>
<td></td>
<td>Users: Opens the User Account dialog, where you can create and configure users.</td>
</tr>
<tr>
<td>Display list of users at logon</td>
<td>Displays a list of available users (in the Log On dialog) when a user is prompted to log on. The user may select from this list rather than type his user name. If Security Mode is Domain (LDAP) and the offline cache is enabled, only the currently cached users will be displayed.</td>
</tr>
<tr>
<td>Log On on E-Signature</td>
<td>Forces a user to log on with their own user account when they're prompted to e-sign an event. If this is not selected, then the current user account remains logged on regardless of who e-signs the event.</td>
</tr>
<tr>
<td>Area / Element</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Default User</td>
<td>This user is automatically logged on when no other user is logged on, such as when the previous user times out or manually logs off.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This user's privileges should be heavily restricted, to prevent your project from being left vulnerable.</td>
</tr>
<tr>
<td>Virtual Keyboard</td>
<td>The type of virtual keyboard that is displayed on the client when the user is prompted to log on.</td>
</tr>
</tbody>
</table>
Backing up the security system configuration

You can back up your project's security system configuration by exporting it to a file. You can also import a configuration either from a file or from another runtime project.

Exporting the configuration to a file

To export the security system configuration:

1. In the main Security System dialog, click Backup. The Import/Export dialog is displayed.
2. Click Export to file. A standard Save As dialog is displayed.
3. Specify a file name and location for the file, and then click OK.

The exported file is encrypted, using the main password configured in the Security System dialog.

Tip: You can also export the configuration during runtime by calling the ExportSecuritySystem function.

Importing the configuration from a file

If your project's security mode is set to Local Only, then you can import a configuration from a previously exported file.

To import the security system configuration:

1. In the main Security System dialog, click Backup. The Import/Export dialog is displayed.
2. Click Import from file. A standard Open dialog is displayed.
3. Locate the configuration file (*.dat) that you want to import, and then click OK. You will be prompted for the configuration's main password.
4. Type the password, and then click OK. The Import from File dialog is displayed.
5. Select an import method:
   - Import only settings that do not conflict: Merge the imported settings with the current project settings. In the case of conflicts, keep the current settings.
   - Import all settings and replace conflicts: Merge the imported settings with the project settings. In the case of conflicts, use the imported settings.
   - Replace the current settings: Completely replace the current project settings with the settings imported from the file.
6. Click OK.

Tip: You can also import the configuration during runtime by calling the ImportSecuritySystem function.

Importing the configuration from another project

If your project's security mode is set to Distributed – Server, then you can import a configuration from another project if:

- The other project’s security mode is set to Distributed – Client, and its server settings are configured to use your project as the server; and
- The other project is currently running on the same network.

To import the security system configuration:

1. In the main Security System dialog, click Backup. The Import/Export dialog is displayed.
2. Click Import from client station. The Import Security from Client Station dialog is displayed.
The dialog shows a list of runtime projects that are using your project as their security system server. Each project/client listing includes a time stamp that shows when it last cached the security system configuration.

3. Select a client station, and then click **Import from client**. You will be prompted for the configuration's main password.

4. Type the password, and then click **OK**.
Logging on/off

If the project security system has been enabled and the default "Guest" user's privileges have been restricted, then you must log on to fully use the development application and/or the runtime project.

**Note:** The project security system must be enabled before you can use this feature.

To log on to the development application, click Log On on the Project tab of the ribbon.

To prompt a user to log on to the runtime project, do one of the following:

- Call the LogOn function somewhere that an expression can be configured — for example, draw a Button object in a screen and then apply the Command animation to it, so that pressing the button shows a logon prompt; or
- Select the Log On on E-Signature option (in the main Security System dialog), which forces the user to log on whenever he performs some action that requires an e-signature.

In either the development application or the runtime project, the Log On dialog is displayed:

![Log On dialog](image)

Use this dialog as follows:

- To log on as yourself, type your user name and password in the appropriate boxes and then click OK.
- To log on as the default "Guest" user, type guest in the User Name box and then click OK.

**Note:** By default, "Guest" has no password, so you can leave the Password box empty. However, if you've changed the password or you're getting your security settings from a server (either Distributed or Domain), then you will need to enter a password for "Guest."

- To log off, simply click Log Off. The default user (typically "Guest," but this may be changed in the main Security System dialog) is automatically logged on to replace you.

**Caution:** If the security mode is set to Domain (LDAP) and a user created on the LDAP server is required to change his password the first time he logs onto the domain, then he must do that before he will be able to log onto the IWS project.
Blocking or unblocking a user

An individual user in the project security system may be completely blocked from accessing the project, and a blocked user may subsequently be unblocked.

A user may be blocked in the following ways:

- By manually selecting the User is blocked option in the User Account dialog;
- By calling the BlockUser function during runtime; or
- Automatically if the user enters the wrong password too many times. (The number of attempts allowed is configured in the Group Account dialog.)

To check whether a user is blocked, do one of the following:

- Look at their user icon in the Project Explorer, which will be marked with a red circle; or
- Call the GetUserState function during runtime.

To unblock a blocked user, do one of the following:

- Clear the User is blocked option in the User Account dialog; or
- Call the UnblockUser function during runtime.
Password-protecting screens, symbols, and worksheets

Screens, symbols, and worksheets in the Project Explorer can be password-protected. You can assign individual passwords to each file, or you can assign a single password to all files in the project.

Almost all project files are encrypted as a matter of course, to prevent unauthorized analysis by third-party tools. (Screen files are not encrypted, because decrypting them during runtime would decrease performance.) However, you can take the extra step of password-protecting your files to prevent unauthorized changes or reuse by other IWS project developers.

Note: These passwords are always case sensitive.

Assigning a password to a single file

To assign a password to a single project file:

1. In the Project Explorer, find and right-click the desired file, and then click Password Protection on the shortcut menu. The Edit Protection dialog is displayed.
2. Type the new password, and then type it again to confirm.
3. Click OK to close the dialog.

The file is now protected. The next time you try to open it, you will be prompted for the password.

Clearing the password from a single file

To clear a password from a single project file:

1. In the Project Explorer, find and double-click the desired file to open it. You will be prompted for the password.
2. With the file open for editing, right-click the file in the Project Explorer and then click Password Protection on the shortcut menu. The Edit Protection dialog is displayed.
3. Leave the New password and Confirm password boxes empty.
4. Click OK to close the dialog.

The file is no longer protected. You can open the file without being prompted for the password.

Assigning a password to all files

To assign a single password to all files in your project:

1. On the Home tab of the ribbon, in the Tools group, click Verify. The Verify Project dialog is displayed.
2. Click Set password for all files. The Edit Protection dialog is displayed.
3. Type the current password for your project, if any.
4. Type the new password, and then type it again to confirm.
5. Click OK. The verification routine proceeds.
6. Click Close to close the Verify Project dialog.

All files in your project are now protected. The next time you try to open one, you will be prompted for the password.

Clearing the password from all files

To assign a single password to all files in your project:

1. On the Home tab of the ribbon, in the Tools group, click Verify. The Verify Project dialog is displayed.
2. Click Set password for all files. The Edit Protection dialog is displayed.
3. Type the current password for your project.
4. Leave the New password and Confirm password boxes empty.
5. Click **OK**. The verification routine proceeds.
6. Click **Close** to close the **Verify Project** dialog.
   Your project files are no longer protected.
You can quickly translate your project’s user interface to multiple languages, using either machine translation (e.g., Google Translate) or a human translator, and then you can switch your project’s language during runtime with a simple function call.

The Translation Table is a worksheet that you can use to create a multilingual user interface (MUI) for your project. (This is different from changing the language of the development environment itself; that is done with the Language command on the View tab of the ribbon.) The worksheet is divided into a Source column, which contains original pieces of text from your project screens, and a Target column, which contains the translated equivalents of the items in the Source column.

![Translation Table worksheet]

**Note:** Google has made the Google Translate API a paid, limited service, so as of InduSoft Web Studio v7.1, automatic translation of project texts is no longer available. You can still use the Google Translate website to translate project texts, but it requires additional steps.

It is our goal to ensure that the functionality of InduSoft Web Studio continues to evolve and grow around emerging technology. We are pursuing alternatives for automatic translation, and we hope to offer this feature again in the near future.
Add a target language to the Translation Table

The Translation Table is used to manage the languages into which you want to translate your project. Adding a language to the table can be as simple as selecting it from a list and then automatically translating your project texts.

Note: By default, the source language of a project is the language of the development environment itself. In other words, it is assumed that when you create a new project, it is for the same language that you work in. For more information about changing the language of the development environment, see Language.

In some cases, however, you might work in one language but develop your project for another — for example, you might work in Portuguese but develop your project for English. If that is the case, you must remember to set the source language in Step 3 below, to associate the correct language with the Source column of the worksheet.

To add a target language to the Translation Table:

1. Open the Translation Table worksheet by doing one of the following:
   - On the Insert tab of the ribbon, in the Global group, click Translation; or
   - In the Global tab of the Project Explorer, double-click Translation.

   The Translation Table worksheet is opened for editing.

2. Make sure the Enable Translation option is selected.

   If this option is cleared, then the entire Translation Table worksheet is disabled and the language cannot be changed during run time.

3. If the source language of your project is other than the language of the development environment itself, set the source language:
   a) To the right of the Source language box, click the browse button.

      The Languages dialog box is displayed.

   b) In the Languages dialog box, select the language for which your project was originally developed, and then click OK.

   The selected language is set in the Source language box.

4. Set the target language:
   a) In the Target languages group, click Add.

      The Languages dialog box is displayed.

   b) In the Languages dialog box, select the language to which you want to translate your project, and then click OK.

   The selected language is added to the Select list, and a new worksheet is created for the language. The Source column of the worksheet is automatically populated with all of the translatable text strings in your project, and the Target column is blank.

5. Configure Date order and Date separator as needed for the target language.

   For example, for English-United States (i.e., American English), Date order is typically MDY and Date separator is typically /, resulting in a date format of MM/DD/YYYY. In many European languages, however, Date order is typically YMD and Date separator is typically ., resulting in a date format of YYYY.MM.DD.

   The language is added to the Select list, and a new worksheet is created for the language. The Source column of the worksheet is automatically populated with all of the translatable text strings in your project, and the Target column is blank.

6. Use the Filters to search the worksheet for specific text items; as you type a few characters, the list is dynamically filtered to show only the items that match.

Note: The ampersand character (&) is ignored when filtering rows. This is to improve the handling of text in program dialogs, where ampersands are used to indicate keyboard accelerators.
7. For each text item in the **Source** column, enter the translation in the **Target** column.

You can manually translate the items one by one, or you can automatically translate multiple items using Google Translate.

   a) Highlight the cells in the **Source** column to select them, and then press **Ctrl-C** to copy those items to the clipboard.
      
      You can also click the column header to select the entire column.

   b) In your web browser, go to **translate.google.com**.

   c) Select the **From** and **To** languages.

   d) Click in the text box on the left, and then press **Ctrl-V** to paste the items from the clipboard.
      
      The items are pasted as separate lines.

   e) Click **Translate**.
      
      The translation appears on the right. Again, the items are translated as separate lines.

   f) Highlight the translated items to select them, and then press **Ctrl-C** to copy the items to the clipboard.

   g) In InduSoft Web Studio, highlight the empty cells in the **Target** column to select them.
      
      You can also click the column header to select the entire column.

   h) Press **Ctrl-V** to paste the translated items from the clipboard into the **Target** column.
      
      Because the translated items were copied as separate lines, they should be correctly pasted into the rows of the worksheet.

8. Review the translated items in the **Target** column, to confirm the translations and to make sure they match the untranslated items in the **Source** column.

   **Tip:** If you want to keep the original, untranslated text for a specific item, leave the **Source** column blank.

9. Save and close the worksheet.

Changes made to the translation table will not take effect until you either call the function **SetLanguage** or restart the project runtime.

   **Tip:** The translation table is saved as a tab-delimited text file in your project folder at:

   \[
   [...]\Documents\InduSoft Web Studio v7.1 Projects\project_name\Web\Translation.trn
   \]

   You can open and directly edit this file with Microsoft Excel, if you choose to do so.
Configure fonts for a target language

Configure the fonts for a target language so that translated text items are displayed in another font, size, and/or style that is more appropriate to the language.

Before you begin this task, you should have already added one target language to your project’s Translation Table.

By default, when you change your project’s runtime appearance to one of the target languages in the Translation Table, the translated text items are still displayed in their original font, size, and style. This can cause many different problems, depending on the target language. For example, if the target language uses a non-Roman character set, or it typically has longer or shorter words, or it is read right-to-left instead of left-to-right, then the screen objects with translated text items may not be displayed correctly.

You can use the Font Configuration tool on the Translation Table worksheet to configure new fonts, sizes, and styles for the translated text items. When you set your project’s interface to the target language, the translated text items will be changed to the new configuration. The configuration can be either very general or very specific — just add more rows for more specific changes.

If you have two or more rows that conflict with each other — for example, if one row applies to all objects and another row applies only to Button objects — then the row that most specifically matches a given text item is the one that is applied.

For more information, see “Examples” below.

To configure fonts:

1. Open the Translation Table worksheet by doing one of the following.
   - On the Insert tab of the ribbon, in the Global group, click Translation; or
   - In the Global tab of the Project Explorer, double-click Translation.

   The Translation Table worksheet is opened for editing.

2. In the Target languages area, select the language for which you want to configure fonts, and then click Fonts.

   This font configuration applies only to the selected target language.

   The Font Configuration dialog box is displayed.

   ![Font Configuration dialog box]

3. In the Object column, select the type of screen objects that the configuration will apply to. To have it apply to all types, select (All).

4. In the Source Font column, select the font that the configuration will apply to. To have it apply to all fonts, select (All).

   The list of selectable fonts includes all fonts that are installed on your computer, not just the fonts that are used in your project.

5. In the Source Size column, type the font size (in points) that the configuration will apply to. To have it apply to all sizes, type (All).

6. In the Source Style column, select the font styles that the configuration will apply to. To have it apply to all styles, select (All).

7. In the Source RTL column, select the reading direction (left-to-right or right-to-left) that the configuration will apply to. To have it apply to both, select (All).

8. In the Target Font column, select the font to which the Source Font should be changed. If you want to leave the Source Font unchanged, select *.

   The list of selectable fonts includes all fonts that are installed on your computer.
9. In the **Target Size** column, type the font size (in either points or percentage) to which the **Source Size** should be changed. If you want to leave the **Source Size** unchanged, type *.

10. In the **Target Style** column, select the font style to which the **Source Style** should be changed. If you want to leave the **Source Style** unchanged, select *.

11. In the **Target RTL** column, select the reading direction (left-to-right or right-to-left) to which the **Source RTL** should be changed. If you want to leave the **Source RTL** unchanged, select *.

12. Repeat this procedure for additional rows, if necessary.

13. Click **OK** to save the configuration and close the dialog box.

When you set your project’s interface to the target language, the translated text items will be changed to the new configuration.

### Examples of font configuration

The Font Configuration tool included in the Translation Table worksheet

#### Example #1

You are using ten different fonts in your project and they all have different font sizes. You have added Chinese as a target language in your Translation Table, so when you set it as the language for your project interface, you also want to convert all text items to a font that supports the Chinese character set.

You would create the following configuration using the Font Configuration tool:

<table>
<thead>
<tr>
<th>Object</th>
<th>Source Font</th>
<th>Source Size</th>
<th>Source Style</th>
<th>Source RTL</th>
<th>Target Font</th>
<th>Target Size</th>
<th>Target Style</th>
<th>Target RTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>Arial Unicode MS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

The * configured in the **Target** columns means that the original settings should be kept. In other words, if you have two texts with different sizes, then the font type will be changed to Arial Unicode MS (a general-purpose font that includes all Unicode character sets, including Chinese) but the font sizes will not be changed.

#### Example #2

You are using a single font type in your project, but it has different font sizes depending on where and how it is used. You have added German as a target language in your Translation Table; words tend to be much longer in German, so you when you set it as the language for your project interface, you also want to decrease the font sizes to make sure the translated text items fit within their screen objects.

You would create the following configuration using the Font Configuration tool:

<table>
<thead>
<tr>
<th>Object</th>
<th>Source Font</th>
<th>Source Size</th>
<th>Source Style</th>
<th>Source RTL</th>
<th>Target Font</th>
<th>Target Size</th>
<th>Target Style</th>
<th>Target RTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>*</td>
<td>80%</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Under this configuration, it does not matter how many fonts you are using or what their specific sizes are. They will all be proportionately decreased to 80% of their original sizes.

#### Example #3

You have added Arabic as a target language in your Translation Table, but you have also decided not to translate your button labels. Therefore, you want to convert all objects except buttons to a font that supports the Arabic character set, and you also want to change the reading direction from left-to-right (default) to right-to-left.

You would create the following configuration using the Font Configuration tool:

<table>
<thead>
<tr>
<th>Object</th>
<th>Source Font</th>
<th>Source Size</th>
<th>Source Style</th>
<th>Source RTL</th>
<th>Target Font</th>
<th>Target Size</th>
<th>Target Style</th>
<th>Target RTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td>Source Font</td>
<td>Source Size</td>
<td>Source Style</td>
<td>Source RTL</td>
<td>Target Font</td>
<td>Target Size</td>
<td>Target Style</td>
<td>Target RTL</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------</td>
<td>------------</td>
<td>------------------</td>
<td>-------------</td>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>Arial Unicode MS</td>
<td>*</td>
<td>*</td>
<td>Enabled</td>
</tr>
<tr>
<td>Button</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>(All)</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Even though it would seem like both rows of the configuration would apply to Button objects, only the second row is actually applied because it most specifically matches.
Set the project’s language at startup

Even when you have multiple languages configured for your project, you must still specify which language you want your project to start in at runtime.

This procedure assumes that you have already added at least one target language to the Translation Table. For more information, see Add a target language to the Translation Table on page 497.

To set the language at startup:

1. Open the Translation Table worksheet by doing one of the following.
   • On the Insert tab of the ribbon, in the Global group, click Translation; or
   • In the Global tab of the Project Explorer, double-click Translation.
     The Translation Table worksheet is opened for editing.

2. In the Startup target language list, select the language in which you want your project to start.
   The list of available languages includes the source language in which you developed the project and any target languages that you have added.

3. Save and close the worksheet.

   **Note:** Setting your project’s language will also automatically set the language of the Virtual Keyboard, as long as there is a keyboard configuration file for the selected language. Keyboard configuration files are stored at C:\Program Files\InduSoft Web Studio v7.1\Bin\VK**.ini, where ** is the two-letter language code.

   You can directly set the language of the Virtual Keyboard, separate from the project’s language, by calling the SetKeyboardLanguage function during run time.
Set the project’s language during run time

You can set your project’s language during run time by using the function SetLanguage anywhere that an expression can be configured.

Before you begin this task, you must have already added at least one target language to the Translation Table worksheet. For more information, see Add a target language to the Translation Table on page 497.

The function SetLanguage takes one parameter: the Locale ID for the target language. The Locale ID for a language is shown in parentheses next to the language in the Translation Table worksheet — for example, “English—United States” is 1033.

Note: Locale IDs are the proprietary language/region codes used in Microsoft Windows. For more information, go to: msdn.microsoft.com/library/ms912047

The following example shows how to draw two Button objects that switch the project’s language between English and French:

1. In the Graphics tab of the Project Explorer, double-click a project screen to open it for editing.
2. On the Graphics tab of the ribbon, in the Active Objects group, click Button.
3. Draw a Button object in the project screen.
4. Double-click the Button object. The Object Properties dialog is displayed.
5. In the Caption box, type English.
6. Click Command. The Command animation properties are displayed in the dialog.
7. In the first row of the On Down tab, in the Expression field, type SetLanguage(1033).
9. Duplicate the Button object, either by copy-and-paste or by Ctrl+Click.
10. Repeat steps 4 through 8, replacing the caption with French and the expression with SetLanguage(1036).
11. Save and close the project screen.

During project run time, clicking each button will set the language of the entire project to that language, using the translated text from the Translation Table.

Note: Setting your project’s language will also automatically set the language of the Virtual Keyboard, as long as there is a keyboard configuration file for the selected language. Keyboard configuration files are stored at C:\Program Files\InduSoft Web Studio v7.1\Bin\VK**.ini, where ** is the two-letter language code.
You can directly set the language of the Virtual Keyboard, separate from the project's language, by calling the function `SetKeyboardLanguage` during run time.
Disable translation of selected screen objects

By default, translation is enabled for all screen objects that have text to be translated. However, you can disable translation of selected objects if you need to preserve their original text.

1. Double-click a screen object to open its Object Properties dialog.
2. In the dialog, look for the Enable translation option.
   If the option is not available in the object's basic properties, then click Advanced to access the advanced properties.
3. Clear the Enable translation option.

Once the option is cleared on an object, its text will no longer be translated during project runtime. The text is still added to the translation table and may be processed though the automatic translation engine, along with all other project texts, but the resulting translation will not actually be applied to the object during runtime.
Configure the advanced translation settings

Configure the advanced translation settings as needed.

Before you begin this task, you should have already enabled translation and added at least one target language to your project’s Translation Table.

To configure the advanced translation settings:

1. Open the Translation Table worksheet by doing one of the following:
   • On the Insert tab of the ribbon, in the Global group, click Translation; or
   • In the Global tab of the Project Explorer, double-click Translation.
   The Translation Table worksheet is opened for editing.
2. Click Advanced.
   The Advanced dialog is displayed.
3. Select or clear the options as needed.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore edging spaces</td>
<td>Ignore any leading or trailing spaces in strings to be translated. If this option is not selected, then strings must match exactly.</td>
</tr>
<tr>
<td>Keep original text when translation is blank</td>
<td>Display the string in the <strong>Source</strong> column when the <strong>Target</strong> column is empty. If this option is not selected, then the original text may be replaced with blank space.</td>
</tr>
<tr>
<td>Translate before parsing strings in curly brackets</td>
<td>Translate the string before evaluating any tag/ expression configured in curly brackets. For example, if the original text is &quot;The current user is {UserName}&quot; and the Portuguese translation is &quot;O usuário atual é {UserName}&quot;, then the string is correctly translated before getting the value of {UserName}. If this option is not selected, then the tag/ expression is evaluated before the string is translated. Therefore, if you have not configured a translation that includes the exact value of the tag/ expression, then the string will not be translated.</td>
</tr>
</tbody>
</table>
| Enable alarm/event delimiters               | Enable the use of delimiters in alarm/event messages, so that when the messages are generated and saved, they are formatted in such a way that they can be correctly translated when retrieved. If this option is not selected, then any tag/ expression configured in curly brackets is evaluated when the message is generated and saved, before translation. In other words, the message is saved as a literal string, and if you do not configure a
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>translation that includes the exact value of the tag/expression, then the message will not be translated. Selecting this option, however, will cause the message to be saved with both the tag/expression and its value at the time the message was generated. For example, if you configure an event message to be &quot;The current user is {UserName}&quot;; then it will be saved as &quot;The current user is $$UserName:Michael$$&quot;; which means that the value of UserName was Michael at the time the message was generated. Then, when the message is retrieved, it is translated as described in Translate before parsing strings in curly brackets above, with the saved value inserted as needed.</td>
</tr>
</tbody>
</table>

4. Click **OK** to close the dialog, and then save and close the Translation Table.
Output (LogWin)

You can use the Output window to view debugging messages generated during project runtime. The window displays OPC, DDE, and TCP/IP transactions, module activation, trace tags, and so on.

The window contains the following elements:

- **XRef tab**: Use the Cross Reference tool to get a tag, and to find every place in the project where the tag is being used. Results appear on this tab, providing path and filename, column, row in the spreadsheet. So, if something changes in the tag, and produces unexpected or unsuccessful results, you can locate all instances of the tag for debugging purposes.

  **Note:** The XRef tab does not work for functions, only tags, but it does allow you to look for array indices.

- **Hide Docked Window button (x)**: Click to open or close the window. Alternatively, to hide the window, you can deselect (uncheck) the Output Window option on the View tab of the ribbon.

- **Contract/Expand button (x)**: Click to contract and expand the Output window.

- **Scroll Bars**: Click and drag to view areas of the Output window that are obscured from view because of the window size or the length of your data.

The Output window is dockable, which means you can move it to another location in the development environment. Click on the title bar and drag the window to a new location. Release the mouse button to attach or dock the window to its new location.

**Configuring the Log to Show Additional Information**

By default, the log shows only debugging and error messages — that is, messages indicating that your project is not running properly. If the log showed all messages generated by IWS, it would quickly overflow with information, making it unusable.

To configure the log to show specific additional information:

1. Right-click anywhere in the Output window, and then click **Settings** on the shortcut menu. The Log Settings dialog is displayed.

2. In the Log Options tab of the dialog, select the specific types of messages that you want the log to show.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Read Commands and Field Write Commands</td>
<td>Show any read and/or write commands that are sent to connected devices.</td>
</tr>
<tr>
<td>Protocol Analyzer</td>
<td>Show messages generated by configured device drivers.</td>
</tr>
<tr>
<td>OPC DA 2.05</td>
<td>Show messages generated by the OPC DA 2.05 Client Runtime task.</td>
</tr>
<tr>
<td>OPC XML/DA</td>
<td>Show messages generated by the OPC XML/DA Client Runtime task.</td>
</tr>
<tr>
<td>OPC UA</td>
<td>Show messages generated by the OPC UA Client Runtime task.</td>
</tr>
<tr>
<td>OPC .NET</td>
<td>Show messages generated by the OPC .Net Client Runtime task.</td>
</tr>
<tr>
<td>Logon/Logout</td>
<td>Display a message whenever a user logs on or logs out. (For more information, see Security.)</td>
</tr>
<tr>
<td>Trace Messages</td>
<td>Show messages generated by the Trace() function. This function is used to generate customized messages from within your project.</td>
</tr>
<tr>
<td>Database Messages</td>
<td>Show messages generated by the ODBC and ADO.NET database interfaces.</td>
</tr>
<tr>
<td>Recipe/Report</td>
<td>Show messages generated by the Recipe and Report tasks.</td>
</tr>
<tr>
<td>Display Open/Close</td>
<td>Show detailed information whenever a screen is opened or closed:</td>
</tr>
<tr>
<td>TCP/IP Messages</td>
<td>Show messages generated by TCP/IP communications.</td>
</tr>
<tr>
<td>DDE Messages</td>
<td>Show messages generated by DDE communications.</td>
</tr>
<tr>
<td>Insert date/time</td>
<td>Timestamp each message.</td>
</tr>
</tbody>
</table>

3. If you selected **OPC XML/DA, OPC UA, OPC .NET**, you must specify exactly which messages are displayed in the Output window. To do this:

   a. Click the browse button to the right of the option. An additional Log Settings dialog is displayed.

   ![](opc_ua_log_settings.png)

   **OPC UA Log Settings dialog**

   Each task has its own settings, but the descriptions of the settings are the same for all three tasks.
b. Select the type(s) of messages to display.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Connection| Enable trace messages on changes in connection status between the OPC Client (i.e., your project) and the OPC Server. Examples of messages:  
  OPC UA: Connection established with server “Connection1”  
  OPC UA: Error to connect to server “Connection2” – Time out waiting for server response |
| Subscription| Enable trace messages on subscriptions to server items, such as the creation of new subscriptions and changes in data type. Examples of messages:  
  OPC UA: Group 1 => Subscription created |
| Read      | Enable trace messages on read operations. Examples of messages:  
  OPC UA: Read Group 1 Started – OK  
  OPC UA: Read Group 1 Completed – OK  
  OPC UA: Read Group 25 Started – Error, asynchronous reading pending for the current group |
| Write     | Enable trace messages on write operations. Examples of messages:  
  OPC UA: Write Group 1 Started [Line 1 = 10.25, Line 42 = 20.45] – Status OK  
  OPC UA: Write Group 1 Completed [All Items] – Status OK |


c. In the Verbosity list, select the level of verbosity.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Critical issues that have caused the communication to fail. These issues must be resolved before you can resume communication.</td>
</tr>
<tr>
<td>Warning</td>
<td>Non-critical issues that may cause the communication to fail under other conditions. These issues should be resolved as soon as possible. Includes Error above.</td>
</tr>
<tr>
<td>Information</td>
<td>All trace messages on the communication. This is the default. Please note that this can be extremely verbose, depending on which type(s) of messages you have selected to display and how many item subscriptions you have created. Includes Error and Warning above.</td>
</tr>
</tbody>
</table>
d. Click **OK** to close the dialog.

4. In the *Log Tags* tab of the dialog, click **Add** to browse for project tags.

![Log Settings — Tags Tab](image)

The *Output* window will display a message whenever the value of a selected tag changes.

5. Click **OK** to save your settings and close the *Log Settings* dialog.
Database Spy

The *Database Spy* window is a debugging tool that allows you to: monitor and force values to project tags; execute and test functions; and execute and test math expressions.

![Sample Database Spy window](image)

The window contains the following elements:

- For each item that you want to monitor during runtime:
  - **Tag/Expression**: Specify a project tag, system tag, or expression that you want to monitor.
  - **Value**: Displays the value returned by the tag/expression.
  - **Quality**: Displays the quality (GOOD or BAD) of the value returned by the tag/expression.
  - **Continuous**: Select this option to have the project continuously evaluate the tag/expression.
- **DB tabs**: The window is divided into multiple sheets, so that you can keep your items organized.
- **Scroll bars**: Use to view areas of the *Database Spy* that are obscured from view because of the window size or the size of the current sheet.

**Tip**: The Database Spy is dockable, which means you can move it to another location in the development environment. Click on the titlebar and drag it to a new location. Release the mouse button to attach or dock the window to its new location.
Using the LogWin Module

This module provides a continuous record of activities and tags for debugging over long periods of time. It creates a file into which you can dump the data collection results, and this file continues to grow in size until you stop the logging (data collection) process. Use the LogWin module (local and remote) to record communication, activate modules, trace tags, and so on.

To enable the LogWin module as a part of your project:

1. On the Home tab of the ribbon, in either the Local Management group or the Remote Management group (depending on where the project is being served from), click **Tasks**. The **Execution Tasks** dialog is displayed:

![Execution Tasks dialog]

2. Select LogWin from the **Task** list.

3. With the LogWin task selected, click **Startup**. The **Startup** dialog is displayed:

![Startup dialog]

4. Click the **Automatic** radio button.

5. Click **OK** to apply your choice and close the **Startup** dialog.

6. Click **OK** again to close the **Execution Tasks** dialog.

**Note:**

- To start the LogWin module directly on your Windows Embedded device, choose **Tools > LogWin** from the EmbeddedView or CEView menu bar.
- To enable logging in the **Thin Client** version of a project, use the **Log** feature in the project settings (**Thin Client** on the Project tab of the ribbon).
Using Remote Tools

**Remote Database Spy**

You can use the **Remote Database Spy** tool to monitor the **Database Spy** of a project running on a remote computer. The project must have the **Database Spy execution task enabled**, and the remote computer must be in **runtime**.

To use the Remote Database Spy tool:

1. On the Home tab of the ribbon, in the Remote Management group, click **Database Spy**. The **Remote Computer** dialog is displayed.

2. Enter the IP address of the remote computer, as shown below.

   ![Entering a Remote IP](image)

   **Note:** The IP address **192.168.1.52** is only an example. Please verify the IP address of the computer to which you want to connect.

3. Click **OK** to connect to the specified address. If the connection is good, then the **Remote Database Spy** window is displayed.

   ![Remote Database Spy](image)

   **Note:** You cannot add or remove tags remotely; the Database Spy tag list must be configured on the remote computer itself.

When you are done, click **Close** to disconnect from the remote computer.

**Remote LogWin**

You can use the **Remote LogWin** tool to monitor the **Output log (LogWin)** of a project running on a remote computer. The project must have the **LogWin execution task enabled**, and the remote computer must be in **runtime**.

To use the Remote LogWin tool:

1. On the Home tab of the ribbon, in the Remote Management group, click **LogWin**. The **Remote Computer** dialog is displayed.
2. Enter the IP address of the remote computer, as shown below.

![Image of Remote Computer]

**Entering a Remote IP**

**Note:** The IP address **192.168.1.52** is only an example. Please verify the IP address of the computer to which you want to connect.

3. Click **OK** to connect to the specified address. If the connection is good, then the *Remote LogWin* window is displayed.

![Image of Remote LogWin]

**Remote LogWin**

When you are done, just close the window to disconnect.
About remote management

Use the Remote Management tool to download project files to a target device and then run/stop the project on that device.

In this case, a "target device" is any computer that has the project runtime server software installed and running. For more information, see Installation on page 34.

The actual connection is handled by a small program on the target device called Remote Agent (CEServer.exe).

**Note:** For the purposes of this section, "Windows Embedded" includes Windows Embedded Standard (formerly Windows XP Embedded) and Windows Embedded Compact (formerly Windows CE, Windows Mobile, Windows Phone).
Install EmbeddedView or CEView on a target device

Install EmbeddedView on a Windows Embedded Standard computer, or install CEView on a Windows Embedded Compact device, to use it as a project runtime server and/or client.

You must have already installed the full InduSoft Web Studio software on your computer, either from the installation disc or from the downloadable installer, because the redistributable EmbeddedView and CEView software is included in the InduSoft Web Studio program folder.

To install and run EmbeddedView or CEView, you must have:

- A Windows Embedded-compatible computer or device (a.k.a. the target device);
- A Windows Embedded Standard or Windows Embedded Compact operating system that is currently supported by Microsoft, which at this time includes:
  - Microsoft Windows XP Embedded Service Pack 3
  - Microsoft Windows Embedded Standard 7
  - Microsoft Windows Mobile or Windows CE 5.x/6.x
  - Microsoft Windows Embedded Compact 7
- 100 MB of free hard drive space or non-volatile memory (flash card or hard disk) for the runtime. More space might be required depending on your project size.
- 32 MB of free RAM. More memory might be required depending on your project size.
- An Ethernet or Wi-Fi network adapter, for TCP/IP networking.

The following items are optional but recommended:

- A USB port; and
- Serial COM ports and adapters, to be used for direct communication with PLCs and other devices.

This is optional because many newer device protocols use TCP/IP communication instead of serial communication.

Installing EmbeddedView or CEView on a target device is actually a two-part procedure. First, you will copy the Remote Agent utility to the target device and then run it. Remote Agent allows you to connect from the project development application to the target device. And second, through this connection, you will install the rest of the EmbeddedView or CEView software.

Note: If EmbeddedView or CEView is pre-installed on the target device, you may skip this entire task. Many hardware manufacturers pre-install Remote Agent on their devices, as part of a larger InduSoft Web Studio package.

To install EmbeddedView or CEView:

1. Turn on the target device and make sure it is connected to your TCP/IP network.
   If Remote Agent is pre-installed on the target device, it will automatically run at start up, and you may skip the next step. Many hardware manufacturers pre-install Remote Agent on their devices, as part of a larger InduSoft Web Studio package.

2. Copy the Remote Agent utility to the target device and then run it:
   a) Locate the correct version of the Remote Agent utility [CEServer.exe] for the target device. All versions are stored in your InduSoft Web Studio program folder.

   Remote Agent for Windows Embedded Standard is located at:

   C:\Program Files\InduSoft Web Studio v7.1\Redist\WinEmbedded\Bin\CEServer.exe

   Remote Agent for Windows Embedded Compact is located at:

   C:\Program Files\InduSoft Web Studio v7.1\Redist\WinCE 5.0\processor\Bin\CEServer.exe
...where processor is the specific processor used by the target device. For more information, consult the manufacturer’s documentation.

b) Copy CEServer.exe to the target device by either downloading it over the network, transferring it on a USB flash drive, or syncing it with Microsoft ActiveSync. (ActiveSync is also known as Windows Mobile Device Center in Windows Vista or Zune Software in Windows 7.) You may save the file anywhere you want on the target device, as long as it is in permanent (i.e., non-volatile) memory and it is not in the root folder (i.e., C: \ or \).

c) Set Remote Agent to automatically run at start up.
   You can do this on most Windows Embedded Compact devices by creating a link to it in \Windows \Startup. If this does not work on your device, please refer to the manufacturer’s documentation.

d) Run Remote Agent.
   The Remote Agent window is displayed.

3. Configure the communication settings in Remote Agent:
   a) Click Setup.
      The Setup dialog box is displayed.

   b) Select the appropriate connection type: Serial Port or TCP/IP. If you select Serial Port, also select the COM port and review the advanced settings.
      If you are already connected to the target device via ActiveSync, you do not need to select another connection at this time. However, keep in mind how the target device will actually be used during project run time.

   c) Click OK.
   d) If you selected TCP/IP for the device connection, note the IP address.
   e) Make sure that you leave Remote Agent running on the target device.

4. Use the Remote Management tool to connect to the target device:
   a) Run the project development application on your computer.
   b) On the Home tab of the ribbon, in the Remote Management group, click Connect.
The *Remote Management* dialog box is displayed.

c) Select the appropriate connection type for the target device: **Network IP**, **Serial Port**, or **Microsoft ActiveSync**. If you select **Network IP**, also type the IP address of the target device. If you select **Serial Port**, also select the COM port and verify the advanced serial communication settings.

d) Click **Connect**.

If you are successfully connected, the connection status is shown in the **Status** box and the target device’s specifications are shown in the **Platform** box.

If you are not connected, check both the connection settings and the physical connections. In particular, if you have selected **Network IP** as the connection type, check to make sure that port 4322 is open on any firewalls between your computer and the target station, including on the target station itself.

**Note:** In some cases, the Remote Management tool may not be able to connect via Microsoft ActiveSync to a device running Windows CE 6.0 or later. This is because of a problem in the default configuration of Windows CE 6.0. You can fix the problem by using a small utility that is included with InduSoft Web Studio. The utility is located at:

*C:\Program Files\InduSoft Web Studio v7.1\Redist\ActiveSyncUnlock.exe*

Copy this file to the device using the stand-alone version of Microsoft ActiveSync and then execute the file on the device. It does not matter where on the device the file is located. When this is done, try again to use the Remote Management tool to connect to the device.

If you still cannot connect via Microsoft ActiveSync, empty the device’s \Temp directory and try again.

5. Install the rest of the EmbeddedView or CEView software on the target device:

a) In *Remote Management*, click **Install system files**.

When the installation is finished, the target device's updated status is displayed in the **Status** box.

With EmbeddedView or CEView installed on the target device, you can now use it as a project runtime server and/or client.

**Note:** The following run-time features are supported by the full InduSoft Web Studio software but not by EmbeddedView and CEView:

- DDE task worksheets.
- ODBC task worksheets and built-in functions. (ADO interface is supported.)
- Third-party .NET controls.
- OPC HDA Server.
- OPC UA and OPC .NET clients
• Some minor graphic features, such as fill effects for shapes other than rectangle, picture formats other than BMP or JPG, and Ellipse style.
• Some built-in functions. For more information, see the individual function descriptions.

Please keep these limitations in mind as you develop your projects.
Enable security in Remote Agent and add users

Enable security in Remote Agent on a target device in order to allow only certain users to connect to the device and to encrypt communications between the device and the project development application.

Before you begin this task, the project runtime software (InduSoft Web Studio, EmbeddedView, or CEView) must be installed on the target device and Remote Agent must be running.

To enable security and add users:

1. In Remote Agent on the target device, click **Setup**.
   The **Setup** dialog box is displayed.

2. Click **Users**.
   The **Users** dialog box is displayed.

3. Select **Enable security system**.
4. Click **Add**.
The *Add User* dialog box is displayed.

![Add User dialog box](image)

5. Type the **User Name** and **Password** for the user, and then select which rights that user should have. The available rights correspond to commands in the Remote Management tool in the project development application.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send To Target</td>
<td>Send an entire project to the device.</td>
</tr>
<tr>
<td>Send File</td>
<td>Send a specific file to the device.</td>
</tr>
<tr>
<td>Run/Stop</td>
<td>Run or stop the project runtime server on the device.</td>
</tr>
<tr>
<td>Get From Target</td>
<td>Get an entire project from the device.</td>
</tr>
<tr>
<td>Install System Files</td>
<td>Install the EmbeddedView or CEView runtime software on the device.</td>
</tr>
</tbody>
</table>

6. Click **OK**. The *Add User* dialog box is closed and the user is added to the **User List**.

7. Repeat for all of the users that should be able to connect to the device.

8. Click **Close** to close the **Users** dialog box.

9. Click **OK** to close the **Setup** dialog box, but leave Remote Agent running on the device.

**Customize Remote Agent’s encryption key**

Customize Remote Agent’s encryption key in order to increase the security of remote management.

Before you begin this task, Remote Agent must be installed and running on the target device and security must be enabled.

When you enable security in Remote Agent, connections between the project development application and the target device are automatically encrypted. This prevents third-party programs from intercepting projects or sending unauthorized Run/Stop commands to the target device.

By default, Remote Agent uses a built-in encryption key that should be secure enough for most situations. You may want to customize the key, however, because the more unique the key is, the more secure the connection will be.

To customize the key:

1. Determine what you want the key to be.
   - It does not matter what the key actually is, because it is automatically shared between Remote Agent and the project development application. As such, you can use an online GUID generator to generate a suitable key.

2. On the target device, exit Remote Agent.

• For Remote Agent running on Windows, Windows Server, or Windows Embedded Standard, create a new text file named `RemoteAPI.ini` in the same directory as Remote Agent (`CEServer.exe`).

• For Remote Agent running on Windows Embedded Compact, edit the file named `CEServer.ini` that is located in the same directory as Remote Agent (`CEServer.exe`).

4. Insert the following lines in the initialization file.

```
[Protection]
InternalKey=<your custom key>
```

5. Save and close the initialization file.

6. Run Remote Agent.

Remote Agent will automatically share the new key with the project development application, when you use the Remote Management tool to connect to the target device and log on as an authorized user.
Install or upgrade a softkey license for EmbeddedView or CEView (Remote)

Use the Remote Management tool in InduSoft Web Studio to remotely install or upgrade the softkey license for the EmbeddedView or CEView runtime software.

This task assumes that you have both a Windows computer with the full InduSoft Web Studio software installed and a Windows Embedded device with the EmbeddedView or CEView runtime software installed, that both stations are connected to your network, and that Remote Agent is running on the Windows Embedded device.

Also, you should have already purchased a valid license or upgrade. You may purchase it when you send in the site code (see below), but doing so might increase your downtime. For more information, contact your InduSoft Web Studio software vendor. You can update any license setting (e.g., product type, number of thin clients) or upgrade the software to a new version. The cost of the upgrade depends on the difference between the current and upgraded license settings.

Tip: You can purchase some Windows Embedded devices with an EmbeddedView or CEView softkey license already installed. Ask your software vendor about this possibility.

These instructions apply both to installing a new softkey license and to upgrading an existing softkey license; whatever license you apply will overwrite the previous one, if any.

To install or upgrade a softkey license remotely:

1. In InduSoft Web Studio, on the Home tab of the ribbon, in the Remote Management group, click Connect. The Remote Management dialog box is displayed:

2. In the Target Station group, do one of the following to configure the connection to the target device running EmbeddedView or CEView:
   - Click Network IP, and then type the IP address of the target device;
   - Click Serial Port, and then select a port from the list provided; or
   - Click Microsoft ActiveSync.

   Tip: TCP/IP links provide better communication performance than serial links.

3. When the Connect button becomes active, click it to connect to the target device. The Status and Platform boxes show whether you have successfully connected to the target device.

4. When you have connected, click the Embedded License tab.
The current license settings are displayed, if any, as well as a randomly generated site code for the target device.

![Embedded License tab of the Remote Management dialog box](image)

5. Send the site code to your software vendor.
   Typically, you will paste the site code into an email to your software vendor.

6. When you receive the corresponding site key from your software vendor, type or paste it in the **Site Key** box, and then click **Send**. (You will be prompted to confirm.)
   The new license settings are saved on the target device, and then a confirmation message is displayed.

If the new site key is not validated, an error message is displayed. If this happens, double-check that you typed the site key correctly. If you typed it correctly and still get an error message, contact your software vendor for further assistance.

Also, after the new license settings are saved on the Windows Embedded device, be sure to save its registry settings. If you do not save these settings, you will lose the license after rebooting the device. For more information about how to save the registry settings, see the manufacturer's documentation.
Download your project to the target device

This task describes how to download your project files to the target device that will host your project.

To complete this task, you must be connected to the target device and the project runtime software (InduSoft Web Studio, EmbeddedView, or CEView) must be installed on it.

1. On the Home tab of the ribbon, in the Remote Management group, click Connect.
   The Remote Management dialog is displayed with the Target tab selected.

2. On the Target tab, check to see if you are already connected to the target device. If you are not, then review the connection settings and click Connect.
   The Remote Management tool connects to the target device and its status is displayed.

   **Tip:** If the connection settings are correct but you cannot connect to the target device, then make sure Remote Agent (CEServer.exe) is running on the device, empty the device's Temp directory, and try again.

3. In the list of tabs on the left, click Project.
   The Project tab of the dialog is displayed.

4. In the Local box, you should see location/filepath of the project that is currently open.
   This is the project that will be downloaded to the target device.

5. In the Target box, verify the location/filepath for the project files on the target device. If the path is incorrect, then click the browse button to the right of the Target box to select a new path. Assuming you are properly connected to the target device, you should be able to browse it like any network drive.

   **Note:** By default, you can download the project files to any location on the target device, even to another location outside the directory that contains the runtime software. You might consider this a security vulnerability, however, so if you want to restrict downloads only to the directory that contains the runtime software, then go to Remote Agent on the target device and select Lock project download.

6. Click Download to download the entire project to the target device, or click Send File to select an individual file to send.

   When you download your project to the target device, the new project replaces the old one automatically and immediately, even if the old project is still running. Also, if you clear the Only newer files option, then the development application will delete all of the old files in the project folder before downloading the new files. As such, you may want to manually stop the old project (by clicking Stop) before you download the new one. That way, you can make sure it stops gracefully and does not disrupt any other processes.

   It is not strictly necessary to stop the project, however, so if your project is robust enough to handle the switch then you can send new files whenever you need to.

   **Note:**
• Once you've initially configured the Remote Management settings, you can use **Download** (on the Home tab of the ribbon) to send updated project files at any time without opening this dialog.

• You can compress the project files to make them download more quickly over a slow network connection. To do this, select the **Enable File Compression** check box in the **Communication tab** of the **Project Settings** dialog.

• If the download is interrupted, then the development application will request confirmation to continue and advise you that the project may not run properly.
Run or stop your project on the target device

Once you have downloaded your project files to the target device, you may run or stop the project at will. There are three ways to run or stop a project on a target device.

**Ribbon**

Assuming you are currently connected to the target device and you have downloaded your project files to it, the easiest way to run and stop the project is to use the appropriate commands on the ribbon: on the **Home** tab of the ribbon, in the **Remote Management** group, click **Run** or **Stop**.

**Remote Management**

If you are not currently connected to the target device:

1. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Connect**. The **Remote Management** window is displayed.
2. Click the **Target** tab of the **Remote Management** window, check the connection settings, and then click **Connect**.
3. Click the **Project** tab of the **Remote Management** window, check the project settings, and then click **Run** or **Stop**.

**Remote Agent**

If you cannot connect to the target device:

1. Go to the target device itself and make sure **Remote Agent** (**CEServer.exe**) is running on it.
2. In **Remote Agent**, click **Run** or **Stop**.
Configure Remote Agent to autorun a project

By default, you must manually run your finished project on the target device, either from your PC by using the Project tab of the Remote Management dialog (see above) or on the target device itself by clicking the Start button in the Remote Agent dialog.

However, you can configure the target device to automatically run a specified project. To do this, edit the file CEServer.ini on the target device to include the following setting:

```
[Setup]
AppName=Applicaion Path
```

Where `project_name` is the location of the IWS project files on the target device. For example:

```
[Setup]
AppName=\Harddisk\Test\CEserverTest
```

The next time the target device boots up and opens the Remote Agent dialog (CEServer.exe), it will read this setting and automatically run the specified project.

There are three ways to edit the CEServer.ini file:

- Edit the file directly on the target device using an attached keyboard or the touchscreen keypad. The file should be located in the same directory as the CEServer.exe file, which was installed earlier.
- Mount the target device as a shared volume on your PC and edit the file there.
- Edit the file in the `[...]\InduSoft Web Studio v7.1\Redist` directory before you install the system files on the target device.

⚠️ **Caution:** This last method changes the default copy of CEServer.ini that is included with IWS. Use this method only if:

- You back up the file before editing it;
- You are installing the same system files on multiple, identical devices; and
- You already know the location (file path) of the IWS project files on the target device (perhaps by using the normal installation method on a test station).
About thin clients and mobile access

This section describes how to make your project accessible to web thin clients and mobile devices.

InduSoft Web Studio is built on a server/client architecture that can support both thick clients and thin clients. The choice of the type of client depends upon your system requirements:

- A thick client is a computer that performs most, if not all, of the processing activity during project run time. It has sufficient processing power, memory, and graphics to run the complete project files, and it only exchanges data (i.e., function synchronization and tag value changes) with the project runtime server as needed. (A project runtime server can also be a thick client to another server.)

In IWS, thick clients are handled through Remote Management.

- A thin client is a computer that depends primarily on the project runtime server for processing. It only needs to have a network connection to get screens and data from the server and a web browser to display the screens to the user.

Thin Clients in IWS

InduSoft Web Studio allows you to create screens that can be viewed on a remote station in a regular web browser. The station where the user can view the screens is called the thin client.

Typical thin client architecture

The actual InduSoft Web Studio software is installed only on the server station. All project files — the tags database, screen files, and task worksheets — are stored on the server, and all background and communication tasks are executed on the server.

The thin client simply loads your project's graphical interface (i.e., the screens containing objects and animations) as needed and then uses that interface to represent data (i.e., tag values) on the server. You do not need to install the IWS development application or any of the project files on the thin client.

This solution provides a high level of flexibility because any computer that has a network connection to the server station (via TCP/IP) can access the project during run time.

Note: Since screens and screen objects may contain scripting, using VBScript and/or the IWS Scripting Language, these scripts are executed on the thin client.

Competitive Advantages of Thin Clients

InduSoft Web Studio is built on a server/client architecture that supports true thin clients. This capability is built into IWS and is not an add-on. This means that:

- The project runtime server can support a large number of simultaneous thin client connections. Each thin client can view the same or different screens as another thin client.

- The server knows which screen each thin client is viewing and automatically "pushes" any tag value changes to the thin client, thereby eliminating the need for browser refreshes.

- The server can support run-time language switching for each thin client, which means that one thin client can display a screen in English while another thin client can display the same screen in Spanish.
• The project can be configured to support redundant data and web servers with automatic switch-over.

**Thin Client Licensing**

The maximum number of simultaneous thin client connections depends on settings of the license installed on the project runtime server. The user does not need to install any license on the thin client.
Building a Simple Thin client program

As we will see shortly, there is no one way to build a Thin client program. The following procedures allow you to develop a simple, unsecured Thin client program.

Procedure A: Thin Client using NTWebServer, Local Loopback

An easy way to initially develop and test a Thin client program is to use the light-weight Web server NTWebServer on your development PC, initiate the runtime project and use Microsoft Internet Explorer on the same PC to display the Web pages.

Step 1: Configure IWS Settings

1. On the Project tab of the ribbon, in the Web group, click Thin Client.

   - In the Data Server IP Address field, enter **127.0.0.1**
   - Check the Auto Screen Scaling, Enable File Compression, and Enable Tooltips checkboxes are checked
   - Click on the IP Security Button. Be sure the Enable checkbox is unchecked. Click OK.
   - Click on the Advanced Button. Be sure the Web Tunneling Gateway Enabled checkbox is unchecked. Click OK.

2. Open the Execution Tasks dialog (Tasks on the Home tab of the ribbon) and be sure the TCP/IP Server task is set to Automatic.

Step 2: Configure Database
1. In the Development Environment, select the Global tab of the Project Explorer. Open the Project Tags folder.

2. Be sure the project tags are properly set to either Server or Local. If the tags are to be exposed to the Web Client, then set them to Server, otherwise set to Local.

**Step 3: Develop your Screens and create HTML screens**

1. Develop your project screen. Depending on your Thin Client screen size, you may want to develop a separate set of screens with a different resolution. Auto Screen scaling is supported (enabled in a prior step), but auto screen scaling naturally has limitations.

2. Save and close all project screens.

3. Be sure you have defined a Startup Screen (Viewer on the Project tab of the ribbon).

4. You can save individual screens as HTML by selecting the Save as HTML or Save Screen Group as HTML options in the Application menu, or save all Screens and Screen Groups as HTML by selecting the Save All as HTML option.

5. If you are just updating a Screen, and especially when you make any configuration changes to the Web settings, you should run the Verify Project tool (i.e., on the Home tab of the ribbon, in the Tools group, click Verify).

6. The HTML (web) pages will be stored in the Web sub-folder of your project folder.

**Step 4: Install NTWebServer**

1. NTWebServer is InduSoft’s lightweight Web Server for Windows NT/XP/2000/Server 2003/Vista environments. NTWebServer.exe is found in [...]\InduSoft Web Studio v7.1\Bin

2. Copy NTWebServer.exe from the \Bin folder and paste it into the Web sub-folder of your project folder.

3. From the Web sub-folder, double click on NTWebServer to start it.

4. A new window should pop-up. At the bottom, there should be a message indicating that NTWebServer is listening. If a message appears that NTWebServer failed to open a socket, it is most likely caused by Microsoft IIS (Web Server) running in a background mode. If this is the case, you will need to stop IIS, and then restart NTWebServer.

   **Tip:** A web server typically operates on, or "listens to," a computer’s TCP/IP port 80. Only one running process can listen to a given port, so if another process on your computer — for example, some third-party SCADA software — is already listening to port 80, then it and the web server process may conflict with each other. You must either configure one of the processes to listen to a different port or use Task Manager to end the conflicting process. If you cannot identify the conflicting process, then in Windows, open Command Prompt and enter the following command to get a list of all networking processes:

   ```
   netstat -a -o
   ```

**Step 5: Start the runtime project**

1. On the Home tab of the ribbon, click Run.

**Step 6: Launch Microsoft Internet Explorer and connect to the Web Server**

1. Click on the Start button (or Alt Tab) to access the Microsoft Internet Explorer program.

2. Start Internet Explorer, and type the address of the starting (home webpage). E.g. http://127.0.0.1/startup.html

3. Note that the startup.sg (or whatever your startup display or screen group name is) will have a HTML file extension on it when accessed from Internet Explorer.

4. Sign on as Guest with no password, assuming no security has been enabled.

**Procedure B: Thin Client using NTWebServer, Network IP**

Once Procedure A has been completed, the next step is to enable network connected Thin Clients, instead of using the local loopback. You need to know the IP address of your PC, as seen from the network. You can get
this information from the Network Settings in the Control Panel, or putting the function `GetComputerIP()` in a Rectangle Object on a screen and running the project.

**Step 1: Configure IWS Settings**

1. Stop any active runtime project.
2. On the Project tab of the ribbon, in the Web group, click **Thin Client**. In the Data Server IP Address field, enter the IP address of your PC (as viewed from the network). E.g. 192.168.1.100
3. Run the **Verify Project** tool (Verify on the Home tab of the ribbon). This will set change the Data Server IP address in the web pages, so that the Thin Client will automatically exchange data with the correct Data Server.

**Step 2: Start the runtime project**

1. On the Home tab of the ribbon, click **Run**.
2. Be sure NTWebServer is still running.

**Step 3: Launch Microsoft Internet Explorer and connect to the Web Server**

1. Click on the **Start** button (or Alt Tab) to access the Microsoft Internet Explorer program.
2. Start Internet Explorer, and type the address of the starting (home webpage). E.g. http://192.168.1.100/startup.html
3. Sign on as **Guest** with no password, assuming no security has been enabled.

**Procedure C: Thin Client using IIS, Network IP**

Once Procedure B has been completed, the next step is to enable IIS to become the Web Server for your Thin Clients instead of NTWebServer. The following are the basic steps to configuring IIS, although depending on your network and Operating System environment, additional settings may need to be configured such as user security.

**Step 1: Configure IWS Settings**

1. Be sure NTWebServer is terminated.
2. Click the Start button, then **Control Panel > Administrative Tools**.
3. Select (click on) **Internet Information Services**.
4. Expand the **Web Sites** tree structure to see the **Default Web Site**.

5. Right click on the Default Web Site and select **Properties** to get the Default Web Site Properties dialog.

6. Click on the **Home Directory** tab. Click on the **Local Path Browse** button and point to your project's **Web** subfolder.

7. Click on the **Web Site** tab. Make sure the TCP port is set to 80. You can click the **Advanced** button to enable the IIS Web Server to respond to specific IP addresses and IP Port numbers.
8. If your Web Server is behind a Proxy, be sure to check the HTTP Keep-Alives Enabled checkbox. It does not hurt anything if this is always checked.

9. Click on the **HTTP Headers** tab. Click on the **MIME Map File Types** button to open the **File Types** dialog. Next, click on the **New Type** button to add a new MIME type. Put the file extension in the **associated extension** field. In the **content type (MIME)** field, put MIME type followed by a `/` character, followed by the file extension name (application/studio). E.g.

   **Associated Extension:** .scc
   **Content Type:** application/studio
10. Click on the Directory Security tab. From this tab, you can change the settings for Anonymous User Access and Authentication Control as well as Secure Communications (i.e., using SSL 3.0).

11. Click on the Anonymous User Access and Authentication Control Edit button to get the Authentication Methods dialog. Normally, you do not have to do anything in this dialog but depending on the Security system your network administrator has installed, you may need to adjust settings in this dialog.

12. The Secure Communications Server Certificate button opens a wizard that lets you define a Certificate for support of secure communications using SSL.

### About MIMEs

- MIME, or Multipurpose Internet Mail Extensions, types instruct a Web browser how to handle files received from a server. For example, when a Web browser requests an item on a server, it also requests the MIME type of the object. Some MIME types, like graphics, can be displayed inside the browser. Others, such as word processing documents, require an external helper program to be displayed.

When IIS delivers a Web page to a client Web browser, it also sends the MIME type of the data it is sending. If there is an attached or embedded file in a specific format, IIS also tells the client program the MIME type of the embedded or attached file. The client program then knows how to process or display the data being received from IIS.

IIS serves only files with known extensions registered in the MIME types list or with the operating system. IIS allows you to configure additional MIME types and change or remove MIME types. Removing a MIME type in IIS does not block access to that MIME type by other programs if it is also registered with the operating system.

IIS is preconfigured to recognize a default set of global MIME types. These MIME types are recognized by all Web sites you create in IIS. MIME types can also be defined at the Web site and directory levels, independent of one another or the types defined globally. When you view MIME types at the Web site or directory level, only the types unique to that level are displayed, not all types inherited from the next level up.

- IIS returns a 404.3 error if a client request refers to a file name extension that is not defined in the MIME types
- MIME configuration is usually only required for Windows Server 2003, not Windows XP or Vista due to Windows 2003 default settings.
- MIME types should include all file extensions found in the Web directory. These include:
  - .app
  - .bin
  - .csv
  - .gis
  - .html
• .ico
• .ini
• .lst
• .rtgis
• .scc
• .scr
• .sg
• .stmp
• .tra
• .txt

Step 2: Start IIS and start the IWS Project

1. Be sure NTWebServer is terminated.
2. Click on the IIS Start button.
3. Run the project.

Notes:
• If IIS is not installed on your PC, you can add it (to Windows XP Pro/Windows 2000 or Windows Server 2003) by opening the Control Panel, then selecting Add/Remove Programs > Add/Remove Windows Components and checking the Internet Information Services (IIS) checkbox in the Windows Components Wizard. You can click on the Details button to select various components of IIS to install. Note that you may need to have your Windows installation disk.

• You can get more information on IIS by opening a browser and entering http://localhost/iishelp in the browser navigation address bar. Note that IIS must be running.

• NTWebServer is a Windows program, while IIS is a Windows service.

• It is STRONGLY SUGGESTED that you use IIS instead of NTWebServer in a runtime mode. It is more reliable and higher performance than NTWebServer.

Troubleshooting:
• If you get a Cannot find Server error message,
  • From the browser, ping the server IP address (primary and/or backup). You can ping using the IP address or use the NetBios name to ping the server (e.g., from a command prompt enter Ping 152.57.100.25 or Ping ServerName.
- Be sure IIS is running
- Be sure your project is running
- Make sure the TCP/IP Server (in the project’s Tasks) is running
- Be sure IIS is set to the correct Home Page (root directory).
- Be sure the Port addresses are correct (HTTP – Port 80, HTTPS (SSL) – Port 443, Data – Port 1234)
- Make sure your firewall has these ports open
- If you are using a Windows Embedded device, be sure ISSymbol is properly loaded and registered
- Be sure your runtime license supports the Web Client configuration
- If you get a **Page cannot be displayed** error message,
  - From the browser, ping the server IP address
  - Stop and restart IIS
  - Be sure the MIME types are properly set
  - Make sure you updated your web pages (i.e., Save as HTML) and use the **Verify Project** tool if you change any Web settings.
  - Verify the Windows security settings are properly set
  - Be sure that the Screen name (and Web Page name) do not have any spaces in the name
- If the web pages are incorrect:
  - Be sure you are pointing to the correct primary URL
  - Be sure your backup URL (if you use it) has the correct (updated) web pages

**The Underlying Technology**

In a IWS project, there are several components used to implement the Thin Client capability.

These components are:

**Data Server**

The Data Server is built-in to the IWS runtime. The Data Server has direct access to the IWS Project Tags Database (runtime) and is responsible for working with ISSymbol to make sure any Tag data being displayed on a Web page at any Thin Client is updated with the latest value(s).

IWS can support a backup or secondary Data Server that will be used should the Primary Data Server become unavailable. The Thin Client will automatically switch over to the Secondary without user intervention required.

**Web Server**

The Primary Web Server is responsible for providing Web pages on demand (i.e., when requested by the Client) through navigation to various project screens by the Thin Client. The Web Server communicates with the Thin Client via HTTP protocol over TCP/IP. SSL (Secure Socket Layer) encrypted communications can be enabled. The Web Server does not need to reside on the same PC as the IWS runtime project. In fact, the Web Server could be a non-Windows corporate Web Server. However, the Web Server needs to have access to the HTML files that are the project Web pages.

IWS supports a Secondary Web Server that will be automatically switched to (by the Client) in case the Primary Web Server becomes unavailable.

**Web Browser**

The Web Browser is located on the Thin Client PC and provides the graphical interface function with the user. Web pages (HTML) is passed to the browser via demand (“pull”) and data is “pushed” to the browser by the Data Server whenever a Tag or Tags referenced on the Screen displayed on the Web Client is updated in the Tag Database.

**ISSymbol**
ISSymbol is a InduSoft-provided ActiveX Control that facilitates the interaction between the browser on the Web Client and the Web Server as well as the Data Server. There are ISSymbol ActiveX Control versions for Windows XP/Vista/7 and all Windows Embedded platforms.

The ISSymbol ActiveX Control is used for both the Internet Explorer-based and Secure Viewer-based browsers.

**Web Tunneling Gateway**

The (Primary) Web Tunneling Gateway is a bridge between the Web Server and the Data Server that is used in one of two situations. The first is whenever data security is required (e.g., IWS data exchanged with the Thin Client needs to be encrypted). The second situation is when the Data Server is "hid" behind a corporate firewall, and only the Web Server IP address (or URL) is exposed.

IWS supports a backup (Secondary) Web Tunneling Gateway to be used if the Primary Web Tunneling Gateway becomes unavailable. The Thin Client will automatically switch over to the secondary Web Tunneling Gateway.

The Web Tunneling Gateway is automatically installed when InduSoft Web Studio is installed on your PC if the installation program detects that IIS is present.

**Note:**

- The Web Tunneling Gateway is automatically installed if IIS is detected during the installation process. Otherwise, it must be manually installed.
- The main function of the Web Tunneling gateway is to encapsulate data packets in HTTP or HTTPS for communication through a firewall.

**ISSymbol Control Layer**

ISSymbol is a component designed by InduSoft that is able to display the screens created with IWS in the Web browser and exchange data (tag values and history data) with the TCP/IP server module of the project. On the Thin Client station, the Web browser (e.g., Internet Explorer) is the container that hosts the ISSymbol control.

ISSymbol works as a control layer between the project and the Web browser — equivalent to the Java Virtual Machine for Java-based applications. This approach provides a high level of security because ISSymbol does not allow the project to access the operating system directly.

When the Web browser downloads the HTML page specified by the user, it checks for ISSymbol control registration on the current computer. If it does not find it, the browser attempts to download registration from the URL specified in the project settings (Thin Client on the Project tab of the ribbon). The Web browser is not able to display the screens from the project if the ISSymbol control is not properly registered in the Thin Client station.

**Caution:** Make sure your Web browser is enabled to download signed ActiveX controls, in order to download ISSymbol automatically. Otherwise, you will need to register ISSymbol manually in the Thin Client station. Check your Web browser's documentation about security settings if you have questions about how to configure these settings.

**Manually Installing the ISSymbol Control**

You can also install the ISSymbol control manually in the Thin Client station. The procedure to install ISSymbol in each operating system is described below:

**Windows PC**

1. Copy the following files...
   - […]\InduSoft Web Studio v7.1\Bin\ISSymbolReg.exe
   - […]\InduSoft Web Studio v7.1\Bin\ISSymbolVM.cab
   ...into any directory of the Thin Client station. Make sure that both files are stored in the same directory.
2. Run ISSymbolReg.exe to register the ISSymbol control on the PC.

**Windows Embedded**
1. Determine the OS version and processor type of the Windows Embedded device, and then find the corresponding redistribution folder in the IWS program directory:

   
   For example, for a MIPS IV processor running Windows CE 5.0 or later:

2. From that directory, copy the following files...

   • IndHTTP.dll
   • IndVkStd.dll
   • ISSymbolCE.ocx

   ...and one of the following Virtual Keyboard layouts...

   • VKEN.ini (for English)
   • VKGE.ini (for German)

   ...into any directory in non-volatile memory on the Windows Embedded device. Make sure that all files are stored in the same directory.

3. At the Windows command prompt, execute the following command:

   `regsvrce.exe "\ISSymbol_directory\ISSymbolCE.ocx"

   For example:

   `regsvrce.exe "\Storage Card\ISSymbolCE.ocx"

4. Save the registry settings to keep ISSymbolCE.ocx registered when you reboot the Windows Embedded device.

Windows Mobile / Pocket PC

1. Follow steps 1 and 2 of the instructions for Windows Embedded (above).

2. Run the program RegSvrCE.exe on the Windows Mobile device. While in the program, do the following:

   a. Locate the file ISSymbolCE.ocx in the directory to which you copied it.
   b. Select the Register option.
   c. Click OK.

   Tip: Your Windows Mobile device may not come with the program RegSvrCE.exe, because such consumer devices (i.e., smartphones and PDAs) are typically not meant to be used like this. If you need to acquire this program — or VBScript.dll, which is also left off most Windows Mobile devices — then please contact Customer Support.

How It Works

After you open the Web browser, you must type the URL for one web page available in the Web Server station (e.g., `http://127.0.0.1/main.html`) into the Address field. At this point, the Thin Client executes the following process:

1. The Web browser downloads the HTML page of the screen you specified.

2. The Web browser checks for ISSymbol control registration in the local computer. If it does not find it, the Web browser attempts to download the ISSymbol component from the URL configured in the project (settings saved in the HTML page). Since the ISSymbol control is properly registered in the Thin Client station, the Web browser loads it.

   From this point on, ISSymbol takes over the communication with the server station, and the Web browser is used only as a host for ISSymbol.
3. ISSymbol connects to the data server. You configure the data server IP Address in the project settings (Thin Client on the Project tab of the ribbon). This setting is embedded in the HTML page.

4. ISSymbol prompts a window on the Thin Client, asking for the User Name and Password. The data you enter is codified by Binary Control and sent to the server. The server station checks the validity of the data and whether you have the rights to open the startup screen. If so, the process continues. If not, you are prompted with an error message indicating that the User Name and Password are invalid. In this case, the process will not continue.

   **Note:** Step 4 is skipped if the Security System is disabled during the configuration of the project.

5. ISSymbol downloads the necessary files to display the screen specified by the user (screen files, tags database, translation files and so forth).

6. ISSymbol connects to the data server and reads the value of the tags that are displayed in the screen you specified.

7. ISSymbol displays the screen on the Web browser and keeps updating the objects according to the values read from the server. Whenever the value of any tag displayed on the open screen(s) changes on the server, the new value is sent to the Thin Client (and vice-versa). Therefore, there is no pooling between the Thin Client and the server station. This method increases the communication performance and optimizes the traffic in the network.

Notice that there are two servers in this process:

- **Web server** (HTTP Server): Provides the files from the server to the Thin Client via HTTP protocol over TCP/IP.

- **Data server** (TCP/IP Server module from IWS): Provides tag values and/or history data from the project running on the server to the Thin Client station(s).

Although both servers are usually running in the same computer, IWS provides the flexibility to run each server in a different station, if necessary. See Web-based application typical architectures for further information.

### Examples of Client/Server Architecture

This section describes some example architectures applied for web-based solutions and provides information on how to configure the project for each architecture. This section does not describe all possible architectures, but it provides the concepts necessary to design and configure different scenarios based on the basic architectures illustrated below

The Web Settings are configured by the Web tab of the Project Settings dialog. To open this dialog: on the Project tab of the ribbon, in the Web group, click Thin Client. By pressing the Advanced button, you access additional settings. The following pictures illustrate these dialogues:
The following table describes the meaning of the main Web settings illustrated in the above dialogues:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Server IP Address</td>
<td>When the Web Tunneling Gateway is disabled: The Thin Client Control (ISSymbol) uses the Data Server IP Address to connect to the IWS TCP/IP Server Task. When the Web Tunneling Gateway is enabled: The Web Tunneling Gateway uses the Data Server IP Address to connect to the IWS TCP/IP Server Task.</td>
</tr>
<tr>
<td>Secondary Data Server IP Address</td>
<td>Same as the Data Server IP Address. However, the Secondary IP Address is used only when the connection with the Data Server IP Address fails.</td>
</tr>
<tr>
<td>Web Tunneling Gateway IP Address</td>
<td>The Thin Client Control (ISSymbol) uses the Web Tunneling Gateway IP Address to connect to the Web Tunneling Gateway.</td>
</tr>
<tr>
<td>Web Tunneling Gateway Secondary IP Address</td>
<td>Same as the Web Tunneling Gateway IP Address. However, the Web Tunneling Gateway Secondary IP Address is used only when the connection with the Web Tunneling Gateway IP Address fails.</td>
</tr>
</tbody>
</table>

The Secondary addresses can be used in the following scenarios:
- When the Thin Clients can connect to either one of two redundant Servers (Web or Data); or
- When the Thin Clients can connect to the Server through the Intranet (LAN – Local Area Network) or through the Internet (WAN – Wide Area Network). In this case, the Primary addresses should be configured based on the network used more often by the Thin Clients. In the following examples, the LAN addresses are used as Primary and the WAN addresses are used as Secondary.

The following table describes the meaning from some terms used in the next examples:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>Local Area Network (for example, Intranet)</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network (for example, Internet)</td>
</tr>
<tr>
<td>Server</td>
<td>Station where the following components are running:</td>
</tr>
<tr>
<td></td>
<td>• IWS (TCP/IP Server task)</td>
</tr>
<tr>
<td></td>
<td>• Web Server (for example, Internet Information Services from Microsoft – IIS)</td>
</tr>
<tr>
<td></td>
<td>• Web Tunneling Gateway for IIS (if enabled)</td>
</tr>
<tr>
<td></td>
<td>Although IWS does not need to run in the same station where the other components are running, the following examples assume that it is.</td>
</tr>
<tr>
<td>Thin Client LAN</td>
<td>Thin client station (Web Browser + ISSymbol control) that connects the Server via the LAN.</td>
</tr>
<tr>
<td>Thin Client WAN</td>
<td>Thin client station (Web Browser + ISSymbol control) that connects the Server via the WAN.</td>
</tr>
<tr>
<td>IP_SERVER_LAN</td>
<td>IP Address of the Server on the LAN.</td>
</tr>
<tr>
<td>IP_SERVER_WAN</td>
<td>IP Address of the Server on the WAN.</td>
</tr>
</tbody>
</table>
### Term and Description

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP_ROUTER_LAN</td>
<td>IP Address of the Router on the LAN.</td>
</tr>
<tr>
<td>IP_ROUTER_WAN</td>
<td>IP Address of the Router on the WAN.</td>
</tr>
<tr>
<td>ScreenName</td>
<td>Name of the project screen, saved as HTML, that is open on the Thin Client station.</td>
</tr>
</tbody>
</table>

### Example 1: Web Server and Thin Client in the same Intranet (LAN)

This is the very common architecture, as well as the simplest to configure. In this architecture, both the Web Server (e.g., Microsoft IIS) and the Data Server (i.e., the IWS TCP/IP Server module) are running on the same PC. The Thin Client connects to the Web Server to download the HTML screen file(s). Then it connects to the Data Server to exchange data with the IWS runtime project. Since both the Thin Client and the Server station are connected to the same network, the Thin Client can access the Server station directly through its IP address (or host name).

**Configuration:**

<table>
<thead>
<tr>
<th>Setting</th>
<th>WTG Enabled</th>
<th>Web Gateway Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Server IP Address</td>
<td>IP_SERVER_LAN</td>
<td>IP_SERVER_LAN</td>
</tr>
<tr>
<td>Secondary Data Server IP Address</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Web Tunneling Gateway IP Address</td>
<td>IP_SERVER_LAN</td>
<td>-</td>
</tr>
<tr>
<td>Web Tunneling Gateway Secondary IP Address</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:**

- **URL From Thin Client LAN:** http://IP_SERVER_LAN/ScreenName.html
- **Home directory of the web server (HTTP server) on the server station:** Web sub-folder of the project

### Example 2: Web Server with Intranet (LAN) and Internet (WAN) Connections

This architecture has both the Web Server (e.g., Microsoft IIS) and the Data Server (i.e., the IWS TCP/IP Server module) running on the same PC. Thin Clients can connect to the Server through either an Intranet (LAN) connection to the Server or an Internet (WAN) connection to the Server (e.g., two different Ethernet ports).

**Configuration:**
### About thin clients and mobile access

<table>
<thead>
<tr>
<th>Setting</th>
<th>Web Gateway Enabled</th>
<th>Web Gateway Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Server IP Address</td>
<td>IP_SERVER_LAN</td>
<td>IP_SERVER_LAN</td>
</tr>
<tr>
<td>Secondary Data Server IP Address</td>
<td>IP_SERVER_LAN</td>
<td>IP_SERVER_WAN</td>
</tr>
<tr>
<td>Web Tunneling Gateway IP Address</td>
<td>IP_SERVER_LAN</td>
<td>-</td>
</tr>
<tr>
<td>Web Tunneling Gateway Secondary IP Address</td>
<td>IP_SERVER_WAN</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:**
- **URL From Thin Client LAN:** [http://IP_SERVER_LAN/ScreenName.html](http://IP_SERVER_LAN/ScreenName.html)
- **URL From Thin Client WAN:** [http://IP_ROUTER_WAN/ScreenName.html](http://IP_ROUTER_WAN/ScreenName.html)
- **Home directory of the Web Server (HTTP server) on the Server station:** Web sub-folder of your project folder
- **You must assign a Fixed IP address to the Web Server on the Internet (WAN), and the project must be running in this Server. Consult your ISP provider or IT department for further information about how to get a Fixed IP address for your Server.**

### Example 3: Web Server with Intranet (LAN) and Router Internet (WAN) Connections

This architecture has both the Web Server (e.g., Microsoft IIS) and the Data Server (i.e., the IWS TCP/IP Server module) running in the same PC. Thin Clients can connect to the Server through either an Intranet (LAN) connection or an Internet (WAN) connection. There is a Router between the Intranet (LAN) and the Internet (WAN).

**Configuration:**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Web Gateway Enabled</th>
<th>Web Gateway Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Server IP Address</td>
<td>IP_SERVER_LAN</td>
<td>IP_ROUTER_WAN</td>
</tr>
<tr>
<td>Secondary Data Server IP Address</td>
<td>IP_SERVER_LAN</td>
<td>-</td>
</tr>
<tr>
<td>Web Tunneling Gateway IP Address</td>
<td>IP_SERVER_LAN</td>
<td>-</td>
</tr>
<tr>
<td>Web Tunneling Gateway Secondary IP Address</td>
<td>IP_ROUTER_WAN</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:**
- **URL From Thin Client LAN:** [http://IP_SERVER_LAN/ScreenName.html](http://IP_SERVER_LAN/ScreenName.html)
- **URL From Thin Client WAN:** [http://IP_ROUTER_WAN/ScreenName.html](http://IP_ROUTER_WAN/ScreenName.html)
- **The Router must be configured to forward the TCP Port(s) from its public IP (IP_ROUTER_WAN) to the Server private IP (IP_SERVER_LAN).**

If the Web Gateway is **enabled**, only the HTTP Port (80, by default) or the HTTPS Port (SSL Port 443, by default) must be forwarded from IP_ROUTER_WAN to the IP_SERVER_LAN.
If the Web Gateway is **disabled**, both the HTTP Port (80, by default) and the Studio TCP/IP Server Port (1234, by default) must be forwarded from IP_ROUTER_WAN to the IP_SERVER_LAN. Consult the Router documentation for further information about how to configure Port Forwarding on it.

- Home directory of the Web Server (HTTP server) on the Server station: Web sub-folder of your project folder

- You must assign a Fixed IP address to the Router on the Internet (WAN), and the project must be running in this Server. Consult your ISP provider or IT department for further information about how to get a Fixed IP address for your Server.

**Configuring the Data Server**

IWS has a couple dialogues that are used for configuration of the Data Server and the Web Server configuration to be used. This information is embedded in the HTML web pages that correspond the screens. The Data Server is part of the IWS runtime project and uses the TCP/IP Server module. There are three (3) basic steps to this configuration:

**Step 1: Web Settings dialog configuration**

1. On the Project tab of the ribbon, in the Web group, click **Thin Client**.

2. In the **Data Server IP Address** field, type the IP address of the PC where the runtime project will reside.

3. Check the checkboxes for **Auto Screen Scaling**, **Enable File Compression** and **Enable Tooltips**. If you want to disable the Web Client from issuing commands to the Server, check the **Disable Remote Client Commands** checkbox.

4. If the Web Client is to use a Virtual Keyboard, check the **Virtual Keyboard** checkbox and any additional settings for the Virtual Keyboard at the Thin Client.

5. By clicking on the **Advanced** button, you can define a backup URL (i.e., backup Website for web pages) and a Secondary Data Server IP address. This is for a redundant Web Server and/or a redundant Data
Server, respectively. Web Tunneling can also be enabled in this dialog. The URL for the ISSymbol OCX can also be defined if it is not found on the Web Server.

![Web Tunneling Configuration Dialog](image)

6. By clicking on the **IP Security** button, you can define a range of IP addresses that will be accepted by the Data Server. If a request for data comes from an IP address outside of the defined ranges, the request will not be acknowledged. This is an Embedded Firewall function.

![IP Security Configuration Dialog](image)

### Step 2: Communication Settings dialog configuration

1. On the Project tab of the ribbon, in the settings group, click **Communication**.

![Project Settings Dialog](image)

2. Enter the Port number (1234 is the default) for the Data Server. You can also define the Data Send Period (i.e., time period for updated communication of data values to the Web Client).

3. Optionally enable **Binary Control** of the data. It is more secure, but is slower. The default is disabled.
Step 3: Enable the TCP/IP task

1. On the Home tab of the ribbon, click **Tasks** (local or remote, depending on the project’s target system).

2. Be sure the TCP/IP Server is set to Automatic. This should be the default state, but can be manually configured by selecting the **Startup** button.

3. Be sure the TCP Port number is properly set (see Communication Settings above), otherwise the TCP/IP Server will start then stop.

   **Note:**
   - This configuration information (from Step 1 & 2) gets embedded in the Web Pages generated from the IWS development environment when you **Save As HTML**.
   - If you change any of these settings, you MUST run the **Verify Project** tool.

Using the SetWebConfig Function

The **SetWebConfig** built-in function allows the developer to programmatically configure the Data Server and Web Client configuration, and the resulting configuration settings are automatically updated in your project’s HTML files (located in the **Web** sub-folder). This function always runs on the IWS Server, and works in both a Windows and Windows Embedded environment.

For more information, see **SetWebConfig** function.

Configuring the Web Server to Support IE and Secure Viewer

IWS is capable of supporting both Interned Explorer (IE)-based browsers and Secure Viewer-based browser at the same time. To do this however, you need to take some care in how you configure your system.

**From the Web-server side**
- All HTML files go into the project **Web** sub-folder.
- Configure your Web Server (e.g., IIS) to have its root folder point to the project folder.
- Put ISSymbolVM.cab in the project folder if an Internet connection to InduSoft’s Web site is not available and the ISSymbol ActiveX control is not installed on the IE-based Web Client.

**From the Web-Client side**
- The Secure Viewer-based browser should be configured to point to the .APP file in the project folder.
• The IE-based browser should have as an initial URL something like as follows:

   http://192.168.1.100/Web/startup.html

   ...where startup.html is your startup web page.

### Configuring a web server to host your project pages

As part of deploying your project over the Web, you must configure a web server to host your project pages.

You are not required to use a Windows PC or a Windows Embedded device to host your project pages. The pages are essentially static files waiting to be downloaded; all runtime processing is handled by the project viewer (i.e., Internet Explorer with ISSymbol installed, or Secure Viewer) on the Thin Client. As such, you can use any standards-compliant web server on any computer platform to host your pages.

For example, if you already have a Unix-based intranet server, then you can copy your project's Web sub-folder (or whatever folder in which you’ve saved your project pages) to the server and have your Thin Clients point to that server's address.

Please note, however, that the web server you choose may not be robust enough to serve your project in a production environment and/or it may not support all features of InduSoft Web Studio. If you want to use these features, then in most cases you should use Microsoft IIS as described below. Specifically:

• To support Mobile Access, the server must be able to process Active Server Pages (ASP) and ASP.NET; and
• To support Web Tunneling Gateway (WTG), the server must allow Internet Service API (ISAPI) extensions.

Before you install and configure any software, please review its documentation thoroughly.

---

**Tip:** A web server typically operates on, or "listens to," a computer’s TCP/IP port 80. Only one running process can listen to a given port, so if another process on your computer — for example, some third-party SCADA software — is already listening to port 80, then it and the web server process may conflict with each other. You must either configure one of the processes to listen to a different port or use Task Manager to end the conflicting process. If you cannot identify the conflicting process, then in Windows, open Command Prompt and enter the following command to get a list of all networking processes:

   netstat -a -o

---

**NTWebServer and CEWebServer**

NTWebServer and CEWebServer are lightweight, zero-configuration web servers included free with InduSoft Web Studio. You can use them to demonstrate your project and run basic tests without making the financial and technical investment in a full-featured web server. However, they are not robust and they do not support SMA or WTG. For real-world applications, we recommend that you use Microsoft IIS as described below.

NTWebServer can be run on any supported Windows PC, and it can be found in the IWS program directory at `\[...\]\Indusoft Web Studio v7.1\Bin\NTWebServer.exe`. Copy it to your project's Web sub-folder before you run it; it must be located in the same folder as the pages it will serve. After you run it, that folder becomes the top level (or "home" directory) of the website.

CEWebServer can be run on any supported Windows Embedded device, and it can be found in the IWS program directory at `\[...\]\Indusoft Web Studio v7.1\Redist\CE_version\CE_processor\CEWebServer.exe`. Assuming you've already copied your project's Web sub-folder to the device's non-volatile memory, copy `CEWebServer.exe` to the same folder and then run it. Again, that folder becomes the top level of the website.

---

**Note:** Both NTWebServer and CEWebServer must run as normal programs; they cannot run as Windows services.

**Microsoft IIS**

Internet Information Services (IIS) is the full-featured server software that is bundled with Windows Server and "professional" versions of Windows:
## Versions of IIS

<table>
<thead>
<tr>
<th>Version of Windows</th>
<th>Version of IIS</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP Professional</td>
<td>IIS 5.1</td>
<td>Maximum of 10 simultaneous connections.</td>
</tr>
<tr>
<td>Windows Server 2003</td>
<td>IIS 6.0</td>
<td>No limit on connections.</td>
</tr>
<tr>
<td>Windows XP Professional x64 Edition</td>
<td>IIS 6.0</td>
<td>No limit on connections.</td>
</tr>
<tr>
<td>Windows Server 2008</td>
<td>IIS 7.0</td>
<td>No limit on connections.</td>
</tr>
<tr>
<td>Windows Vista (most versions)</td>
<td>IIS 7.0</td>
<td>No limit on connections.</td>
</tr>
<tr>
<td>Windows Server 2008 R2</td>
<td>IIS 7.5</td>
<td>No limit on connections.</td>
</tr>
<tr>
<td>Windows 7 (all versions)</td>
<td>IIS 7.5</td>
<td>No limit on connections.</td>
</tr>
<tr>
<td>Windows CE and Windows Mobile 5.0 and later</td>
<td>IIS for CE .NET</td>
<td>Must be included in Platform Builder. Default is maximum of 10 simultaneous connections.</td>
</tr>
</tbody>
</table>

IIS supports all features of InduSoft Web Studio and it is robust enough to serve almost any IWS project in a production environment. It’s the web server that we recommend for most users. However, to properly install and configure it, you should be experienced with administering Windows on a network.

For the sake of system security, IIS is turned off by default when the operating system is installed. To turn it on, use the Windows Features dialog (in Windows 7, Control Panel > Programs > Programs and Features > Turn Windows features on or off). If you want to use Mobile Access and/or WTG in your project, make sure ASP, ASP.NET, and ISAPI Extensions are also turned on.

![Turning on IIS in the Windows Features dialog](image)

Once IIS is turned on, you can use Administrative Tools (in Windows 7, Control Panel > System and Security > Administrative Tools) to configure it. For more information, please refer to Microsoft’s extensive documentation.

### Apache for Windows

If IIS is not available to you or if you choose not to use it, then the second most popular web server for Windows is the open-source Apache. However, it requires even more expertise than IIS to properly install and configure, so please review the documentation thoroughly before you attempt it.
Installing the web tunneling gateway

The Web Tunneling Gateway (WTG, a.k.a. IndWebSvc) enables an Internet-connected computer — typically your web server — to route project data between remote thin clients and a data server that is not connected to the Internet. This section describes how to install WTG.

Before you proceed with this task, you should first set up Microsoft IIS as a web server to host your project pages; WTG can only run on Microsoft IIS. Also, your web server must be connected to the Internet and have a static IP address, rather than a dynamically assigned IP address. Consult your IT department or ISP about how to get a static IP address. Finally, your web server and data server both must be on the same internal TCP/IP network, even if only your web server is connected to the Internet.

In order for thin clients to access your project, they must be able to communicate with both the web server that is hosting your project pages and the data server that is actually running your IWS project. This is not a problem if your servers and clients are all on the same internal network, nor if you connect both of your servers to the Internet so that remote thin clients can communicate with them.

However, if you choose not to connect your data server to the Internet (for network security or topology reasons), then you must route the project data through another computer that is connected.

WTG provides these routing capabilities. It is an ISAPI extension for Microsoft IIS, and it is typically installed on the same web server that is hosting your project pages. It encapsulates the packets sent between the data server and the thin clients, thereby ensuring project security, and it can route data for multiple data servers at the same time.

These instructions are for:
- Microsoft IIS 7.0 running on Windows Vista and Windows Server 2008; and
- Microsoft IIS 7.5 running on Windows 7 and Windows Server 2008 R2.

1. On your web server (or whichever Internet-connected computer that will act as the gateway), make sure Microsoft IIS is running and ISAPI Extensions are enabled.
   For more information, see Configuring a web server to host your project pages.

2. Copy the WTG installer from the IWS application directory to the "root" directory of your website (e.g., C:\inetpub\wwwroot\).
   The installer is located at C:\Program Files\InduSoft Web Studio v7.1\Bin\WebTunnelingGateway.exe

3. Run the installer, and then proceed through the installation wizard.
   There are no options to select.
   The installer will create a new directory in the website’s "root" directory and install the WTG extension files there. It will also create a new application pool.

4. Open IIS Manager.
   a) Click Start, and then click Control Panel.
   b) In the Control Panel window, do one of the following.
      - If you are using Windows Server 2008, click Administrative Tools.
      - If you are using Windows Vista, click System and Maintenance, and then click Administrative Tools.
      - If you are using Windows 7 or Windows Server 2008 R2, click System and Security, and then click Administrative Tools.
   c) In the Administrative Tools window, do one of the following.
      - If you are using Windows Vista or Windows 7, double-click Internet Information Services (IIS) Manager.
      - If you are using Windows Server 2008 or Windows Server 2008 R2, double-click Server Manager, and then in the Server Manager window, click Internet Information Services (IIS) Manager.
   The Internet Information Services (IIS) Manager window is displayed.

5. Make sure the application pool was created by the installer.
   a) In the Connections pane, expand the server node and click Application Pools.
   b) On the Application Pools page, in the list of pools, look for IndWebSvcPool.
If `IndWebSvcPool` is not in the list, then the WTG installer did not finish correctly. Return to the beginning of this task.

6. Add an ISAPI restriction for IndWebSvc.
   a) In the Connections pane, click the server node.
   b) On the server node’s Home page, double-click ISAPI and CGI Restrictions.
   c) On the ISAPI and CGI Restrictions page, in the Actions pane, click Add.
      The Add ISAPI or CGI Restriction dialog is displayed.
   d) In the ISAPI or CGI path box, type the path to the file `IndWebSvc.dll` (e.g., `C:\inetpub\wwwroot\indwebsvc\IndWebSvc.dll`), or click the browse button (...) and navigate to the file.
   e) In the Description box, type IndWebSvc.
   f) Select Allow extension path to execute.
   g) Click OK.

7. Configure the website’s handler mappings to make ISAPI extension files executable.
   a) In the Connections pane, expand Sites, and then click the website that is acting as the gateway (i.e., typically Default Web Site, or whichever site is hosting your IWS project pages).
   b) On the website’s Home page, double-click Handler Mappings.
   c) On the Handler Mappings page, in the list of mappings, select ISAPI-dll.
      This item is disabled by default.
   d) In the Actions pane, click Edit Feature Permissions.
      The Edit Feature Permissions dialog is displayed.
   e) Select Execute.
   f) Click OK.

8. Make sure the website and the gateway extension are running — on the server, start Internet Explorer, and then go to `http://127.0.0.1/indwebsvc/indwebsvc.dll`
   127.0.0.1 is the "localhost" or "loopback" address, so the browser is looking at the website that is running on the same computer.
   The resulting webpage should look like this:

   Studio Web Gateway
   Status: Ok
   Version: 1.3
   Active Connections: 0

   If you do not see this, then either the website is not running or the gateway extension was not correctly configured. Return to the beginning of this task.

9. Close IIS Manager.

10. Configure your IWS project to include the address of the gateway.
    For more information, see Project Settings: Web.

**Configuring the Thin Client**

IWS supports two types of Web Browsers for use with Thin Clients; Microsoft Internet Explorer, which supports ActiveX Controls, and the InduSoft-developed Secure Viewer. Both browsers can use used concurrently with the Web Server, and the choice of browser depends on the project requirements. The Secure Viewer is targeted at plant floor use, while the Internet Explorer browser is generally more appropriate for desktop applications or where the Thin Client needs to connect to different project servers.
Configuring Microsoft Internet Explorer

There is really very little that needs to be done to configure Microsoft Internet Explorer to be used as a Web Browser for a Thin Client. The primary items are:

• Make sure ActiveX Controls are enabled.
  
  The settings to control this are generally found in Tools > Internet Options > Security. Click on the Custom Level button and be sure to allow the ISSymbol ActiveX Control to be installed, then you can return to a higher security level. (Otherwise, be sure to install the ISSymbol ActiveX Control manually).

• Be sure VBScript is supported on your PC. This is generally not an issue with desktop PCs, but it can be an issue on Windows Embedded devices. You will need to be sure the VBRun.dll is installed and registered on your PC.

If you are going to use Microsoft Internet Explorer as your Web browser for Thin Client projects, you may want to remove some navigation bars and buttons to reduce the amount of space taken up by these items.

You can create an icon on the desktop that is a shortcut to your startup web page on the Web Server. This will take you to the web page when you click on the icon. Additionally, you can put this shortcut in the Startup folder so that the Web page is automatically started whenever the browser is started.

Installing the Secure Viewer

Installing on Windows PC

You will need to install the Secure Viewer from your CD (or download from the InduSoft Web site). Follow the following steps:

1. Request the Secure Viewer installer from your software vendor, save it to the designated computer, and start it.

2. Follow the instructions of the installation wizard. There are only two settings that must be configured during installation:
   • URL: Enter the URL or filepath of the project file (project_name.app) on the Web Server.
   • Server IP: Enter the IP address or hostname of the Data Server (a.k.a. TCP/IP Server).

3. Finish the installation and click Finish to close the installer.

The installation wizard automatically installs and registers the Thin Client ActiveX component (ISSymbol.ocx), so if you have correctly configured the URL and Server IP settings, then the Secure Viewer should be ready to go.

**Note:** For additional security of the runtime environment, add a shortcut to the Secure Viewer (Viewer.exe) to the Startup directory in Windows. Viewer.exe will install into the \Bin sub-folder where the IWS program installs.

Installing on Windows Embedded

You can also install the Secure Viewer on a Windows Embedded device, by copying the necessary files to the device’s non-volatile memory. To install Secure Viewer:

1. Determine the OS version and processor type of the Windows Embedded device, and then find the corresponding sub-folder in the IWS program directory. For example, for Windows CE 5.0 running on a MIPS processor, find the [...]\InduSoft Web Studio v7.1\Redist\WinCE 5.0\MIPS\Bin sub-folder.

2. Select the following files in the sub-folder:
   • IndVkStd.dll
   • ISSymbolCE.ocx
   • Tagi.bin
   • Viewer.exe
   • ViewerCfg.exe

3. Copy the files to non-volatile memory on the Windows Embedded device.
4. Register the ISSymbol control on the Windows Embedded device by executing the following command from the Prompt window:

    regsvr32.exe "\ISSymbolPath\ISSymbolCE.ocx"

Example:

    regsvr32.exe "\Storage Card\ISSymbolCE.ocx"

5. Save the registry settings to keep ISSymbolCE.ocx registered when you reboot the Windows Embedded device.

   Tip: Check the device manufacturer's documentation for how to save the registry settings.


Changing the Secure Viewer Configuration

After initial installation, the configuration of the Secure Viewer is saved in the viewer.ini file, which should be in the same directory as Viewer.exe. (This should be in the \Bin sub-folder where the Secure Viewer program is installed). There are two ways to change the configuration of the Secure Viewer after it has been installed. First, you can run the Viewer configuration utility (ViewerCfg.exe) that was also installed.

![Secure Viewer Configuration](image)

   Note: On Windows Vista and Windows 7, the Secure Viewer configuration utility must have Administrator privileges to run properly. It should be installed with those privileges by default, but if you still have problems, then check the file properties for ViewerCfg.exe and make sure Run as Administrator is selected.

The configuration utility provides the following options:

- **Save** button: Immediately save the current settings to a viewer.ini in the same directory as the ViewerCfg.exe utility.
- **Import** button: Load a viewer.ini file from another directory.
- **Export** button: Save the viewer.ini file to another directory.
- **Data Server IP** field: Enter the IP address (or host name) of your data server station. The data server station is the computer or device where the TCP/IP Server module is running.
- **TCP Port** field: Enter the port number of the Data Server, if it is different than the default port of 1234.
- **URL** field: Enter the URL or filepath of the project file (*.app) on the Web Server.
- **Enable Splash Window** option: Check (enable) this option to see a splash window when you start the Secure Viewer.
- **Enable Progress Bar** option: Check (enable) this option to see a progress bar while the Secure Viewer loads the project file.
• **Advanced** button: Click to access additional configuration options:

```
[Options]
nosplash= // Enable (0) or disable (1) the splash window
noprogressbar= // Enable (0) or disable (1) the progress bar
dsl= // Data Server Primary
ds2= // Data Server Secondary
dsp= // Data Server Port
wtg1= // Web Tunneling Gateway Primary
wtg2= // Web Tunneling Gateway Secondary
url= // URL from project file (*.app)
proxyip= // Proxy Address
proxyp= // Proxy Port
ceemul= // Enable (1) or disable (0) Windows Embedded emulation
```

**Example**

PC with IWS runtime project (has both Web Server (NTWebServer or IIS) and Data Server):

- IP Address is 192.168.1.106
- Project name is SecureViewerTest
- Project file is SecureViewerTest.app

Secure Viewer configuration dialog:

- Data Server IP = 192.168.1.106
- TCP Port = 1234
• URL = http://192.168.1.106//SecureViewerTest.app

Alternatively, you could edit the viewer.ini file:

[Options]
url=http://192.168.1.106//SecureViewerTest.app
noprogresbar=1
ds1=192.168.1.106
nosplash=1
ds2=
dsp=1234
wtg1=
wtg2=
user=Guest

[OEM]
Splash=Splash.bmp //this is a splash screen

Note:
• Your InduSoft Web Studio runtime license must support a Secure Viewer (Thin Client) otherwise the connection to the Data Server will be refused.
• Be sure to put NTWebServer in your project folder (not Web sub-folder) or point IIS to your project folder.
• If you have configured your web server to support secure connections via TLS/SSL (i.e., the URL begins with "https" instead of "http") and you have created a self-signed certificate, that certificate must be manually installed as a Trusted Root Certification Authority on any computer that uses Secure Viewer to access the project. This is because Secure Viewer runs in a restricted mode, and it does not provide a way for the user to accept a self-signed certificate offered by a web server. For more information, go to: technet.microsoft.com/library/cc725849.aspx

Implementing Security for Web-based Applications

There are various methods for implementing security of Web-based applications. The approach that you require can depend on a number of factors, and may involve one or more methods of implementing Security.

Method 1: Password Protection

IWS provides the ability to create Groups of Users and individual Users within a Group. Each Group (e.g., Operators, Supervisors, Maintenance) can have different security levels and access different levels of functionality. Individual passwords can be configured for each User.

In addition, Groups can have advanced settings, allowing features like minimum password size, password aging, e-signature on Objects with Command animations, Account Auto-lockup (e.g., lock up after a number
of invalid attempts to access), and User Account blocking (temporarily disable – e.g., when employee is on vacation).

If System Security is enabled, these Password Protection features are also available at the Thin Client station. When a User at a Thin Client station attempts to connect to the Web Server, they will be prompted for a User Name and a Password. If either is invalid, the User will not be let on to the system.

Within a project, the various screen objects and their animations, and Screen access can have a security level assigned to it. The current User logged on must have a access level range which matches the desired Object or Screen. The following is a representative method of assigning security access levels by Group.

For more information, see Security.
Method 2: Disabling Thin Client Commands

IWS allows bi-directional data exchange between the Thin Client and the Data Server. However, for security reasons it may be advantageous to only allow the Thin Client to view the process or machine data, and not send any data back to the Data Server.

Selecting (checking) the **Disable Remote Client Commands** option in the project settings (Thin Client on the Project tab of the ribbon) ensures that all commands coming from a Thin Client station are blocked. The communication becomes unidirectional (from the Server to the Thin Clients):

![Project Settings — Web tab](image)

Method 3: Embedded Firewall

This feature allows the user to filter access to the project based on the Thin Clients IP Address. When a Thin Client attempts to connect to the Server station, the Server checks if the IP Address of the Thin Client station is authorized to access the project. The ranges of authorized IP Addresses can be configured in the Server station by clicking **IP Security** in the project settings (Thin Client on the Project tab of the ribbon):

![IP Security dialog](image)

**Access allowed by IP address**
Method 4: Encrypted Communications (SSL)

By enabling the Web Tunneling Gateway (WTG), you can enable all communications between the Data Server + Web Server and the Thin Client to be encrypted using RC6, a highly-secure 128-bit encryption standard. To use SSL, you must do the following:

1. Click Advanced in the project settings (Thin Client on the Project tab of the ribbon). Select (check) the Web Tunneling Gateway Enabled option. Click on the SSL radio button and be sure the SSL port is set to 443. Click OK.

![Project Settings — Web — Advanced dialog]

2. In your Web Server, be sure SSL capabilities are enabled and that a SSL Certificate of Authentication is present.

3. Be sure SSL is enabled in the Web Client

4. Set up all other Web configurations to support the WTG.

Method 5: VPN

A VPN is a Virtual Private Network. It is called virtual since it really uses the public Internet to transport data from one computer to another. But since this network is encrypted and uses other security mechanisms enabled by the ISP, is it a very secure Private Network. While VPN’s are inherently secure, they are more costly that a simple public Internet connection.

Port Usage

There are various ports that are used by IWS and/or related software. These are:

<table>
<thead>
<tr>
<th>Port #</th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>FTP Server (Data)</td>
</tr>
<tr>
<td>21</td>
<td>FTP Server (Command)</td>
</tr>
<tr>
<td>25</td>
<td>SMTP Server</td>
</tr>
<tr>
<td>80</td>
<td>Microsoft IIS Server for HTTP packets</td>
</tr>
<tr>
<td>110</td>
<td>POP3</td>
</tr>
<tr>
<td>118</td>
<td>Microsoft SQL Server Services</td>
</tr>
<tr>
<td>161</td>
<td>SNMP</td>
</tr>
<tr>
<td>162</td>
<td>SNMP Trap</td>
</tr>
<tr>
<td>389</td>
<td>LDAP</td>
</tr>
<tr>
<td>443</td>
<td>Microsoft IIS Server for HTTPS packets (SSL)</td>
</tr>
<tr>
<td>502</td>
<td>Modbus TCP/IP protocol</td>
</tr>
<tr>
<td>663</td>
<td>LDAP over SSL</td>
</tr>
</tbody>
</table>
### About thin clients and mobile access

<table>
<thead>
<tr>
<th>Port #</th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>1028</td>
<td>FTP Client (Command)</td>
</tr>
<tr>
<td>1029</td>
<td>FTP Client (Data)</td>
</tr>
<tr>
<td>1234</td>
<td>Project TCPI/IP Server</td>
</tr>
<tr>
<td>1443</td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td>1444</td>
<td>Microsoft SQL Server default port (Monitor)</td>
</tr>
<tr>
<td>1521</td>
<td>Oracle</td>
</tr>
<tr>
<td>1526</td>
<td>Oracle</td>
</tr>
<tr>
<td>2030</td>
<td>Oracle</td>
</tr>
<tr>
<td>3001</td>
<td>A-B Ethernet TCP/IP Protocol (default)</td>
</tr>
<tr>
<td>3306</td>
<td>MySQL (can be configured to use 3306-3309)</td>
</tr>
<tr>
<td>3872</td>
<td>Oracle Management Remote Agent</td>
</tr>
<tr>
<td>3997</td>
<td>Studio ADO Gateway</td>
</tr>
<tr>
<td>4322</td>
<td>Remote Agent (CEServer)</td>
</tr>
<tr>
<td>5432</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>47808</td>
<td>BACNet UDP Protocol (default)</td>
</tr>
</tbody>
</table>

You may need to accommodate one or more of these port’s usage in your Firewall settings.

**Exercise: Viewing Your Project on the Web**

To view your project, use the following steps:

1. Expand the Screens folder and double-click on your Main.scr screen.
2. To save the screen in HTML format, click **Save as HTML** on the Application menu.
   - Web files are stored in the Web folder, so open the folder and verify that you saved the Main screen successfully. You should see main.html.
3. On the Project tab of the ribbon, in the Web group, click **Thin Client**.
   - ![Open the Project Settings dialog](image)

4. Configure the **Data Server IP Address** to use the IP Address of the Server station (computer on which you are running) at runtime.
   - The Thin Client station exchanges on-line data (tag values) with the station specified in this field.
5. Type the URL path to your main.html file (in the Web folder) into the **URL** field.
   - The URL depends on the Home directory configured in the Web Server of your Server station.
6. After configuring the Web settings, click **OK** to close the **Project Settings** dialog.

7. Save and close all screens and worksheets, and then verify the project (**Verify** on the Home tab of the ribbon) to update the Web settings to your Web page.

---

### Caution: You must verify the project again any time you change the project settings.

To test your Web-based application, use the following steps:

1. Click **Run** (on the Home tab of the ribbon, in the Local Management group) to run your project locally on the Server station.

2. Open an Internet Browser (Microsoft Internet Explorer or Netscape) and type the URL address to open your main.html screen from the Server station.

3. When the **Log On** dialog displays in the Browser, type **guest** into the **User Name** field, then click **OK** to open the main.html screen in the Browser.

   ![Log On dialog](image)

Notice that you can modify the level of any tank locally (Server station) using the Viewer run-time module or remotely (Thin Client) using the Browser.

---

### Note: A Thin Client requires an ActiveX component (**[ISSymbol.ocx]**) to handle screens on the Browser. If you connect the Thin Client to the Internet, this component is downloaded and registered automatically.
About Mobile Access

You can use Mobile Access (also sometimes called Studio Mobile Access or SMA) to deploy an HTML5-enhanced web interface that presents alarms, trends, process values, and even project screens in a unified, easy-to-use "dashboard".

The web interface is specifically designed for smartphones and tablets, including Windows Phone, iOS, and Android devices, but it can be accessed from almost any computer using a modern web browser.

It is important to remember that although Mobile Access is part of the same project runtime and may be hosted on the same server that hosts the screens published for Secure Viewers and Web Thin Clients, it is a distinct interface based on platform-agnostic technology. Secure Viewer and Web Thin Client are based on ActiveX technology, and as such, they can run only on Windows computers. In contrast, the Mobile Access web interface is based on HTML5, so it can run in most web browsers on most computers and devices.

To use Mobile Access, you must have the Mobile Access Runtime software installed on your project runtime server. Either you selected the Mobile Access Runtime feature when you installed the full InduSoft Web Studio software, or you ran the separate, included Mobile Access Runtime installer (MobileAccessSetup.exe) afterward. For more information, see Install the Mobile Access Runtime software on page 576.

Also, your software license must include enough SMA Clients to accommodate all of the users that you expect to access the Mobile Access web interface at the same time. Please contact your vendor to review your software license. For more information, see License Settings on page 47.

Note: The Mobile Access Runtime software cannot be installed on Windows Embedded, and Mobile Access cannot be configured in projects that are set to run on Windows Embedded target systems.

The rest of this section describes how to configure the Mobile Access web interface during project development, as well as how to log on to and navigate the web interface during project run time. You should already be familiar with how to locate and open worksheets in the Project Explorer, how to edit a worksheet, and how to save and close a worksheet.

List of features supported in project screens in Mobile Access

This is a list of the features that are currently supported in project screens that you select to include in the Mobile Access web interface.

Screen objects and animations

The following table shows exactly which screen objects and animations are supported in project screens, as well as the specific properties that are supported on each one:

<table>
<thead>
<tr>
<th>Group</th>
<th>Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapes</td>
<td>Line</td>
<td>Solid Line, Dashed Line, No Line, Color, Weight</td>
</tr>
<tr>
<td></td>
<td>Open Polygon</td>
<td>Solid Line, Dashed Line, No Line, Color, Weight</td>
</tr>
<tr>
<td></td>
<td>Closed Polygon</td>
<td>All Border Types, Border Color, Border Weight, Fill Color, Fill Effects (Horizontal, Vertical, Diagonal Up, Diagonal Down)</td>
</tr>
<tr>
<td></td>
<td>Rectangle</td>
<td>All Border Types, Border Weight, Border Color, Fill Color, Fill Effects (Horizontal, Vertical, Diagonal Up, Diagonal Down), Caption, Fonts, Multiline, Wrap Text</td>
</tr>
<tr>
<td></td>
<td>Rounded Rectangle</td>
<td>All Border Types, Border Weight, Border Color, Fill Color, Fill Effects (Horizontal, Vertical, Diagonal Up, Diagonal Down)</td>
</tr>
<tr>
<td></td>
<td>Ellipse</td>
<td>Type - Ellipse only, Border Weight, Border Color, Fill Color, Fill Effects (Horizontal, Vertical, Diagonal Up, Diagonal Down)</td>
</tr>
<tr>
<td>Active Objects</td>
<td>Text</td>
<td>Caption, Align, Fonts, Background</td>
</tr>
</tbody>
</table>

Note: Horizontal scaling of text — which you can normally achieve by horizontally resizing the Text object — is not supported. Text will always appear
<table>
<thead>
<tr>
<th>Group</th>
<th>Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>at its full width for the specified font size. If you need &quot;narrow&quot; text, use an appropriate font like Arial Narrow.</td>
</tr>
<tr>
<td></td>
<td>Text Box</td>
<td>Input Enable, Fonts, Mask/Count, Minimum Value, Maximum Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: The character limit specified by the mask (e.g., ######) is not enforced. This might cause unexpected behavior during run time, depending on how the input is used.</td>
</tr>
<tr>
<td></td>
<td>Button</td>
<td>Style - OS Like, Fonts, Align, Multiline, Wraptext</td>
</tr>
<tr>
<td></td>
<td>Check Box</td>
<td>Tag, True Value, Caption, Fonts, 2 states</td>
</tr>
<tr>
<td></td>
<td>Radio Button</td>
<td>Tag, True Value, Caption, Fonts, 2 states</td>
</tr>
<tr>
<td></td>
<td>Combo Box</td>
<td>Label, Position, Disable, Security, Sort, Data Sources (Static Labels or Database; see note), Advanced (Color only), Fonts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: If you select Database as the data source for a Combo Box object, the object must use the default database (primary or secondary) that is configured in the project settings. However, you can select a different table and/or field for each instance of the object.</td>
</tr>
<tr>
<td></td>
<td>Libraries</td>
<td>Linked Symbol -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linked Picture Link File, Transparent (Color Code, Tracker)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Types: VBScript, Open Screen, Close Screen, Set Tag, Reset Tag, Toggle Tag</td>
</tr>
<tr>
<td></td>
<td>Hyperlink</td>
<td>Type, URL</td>
</tr>
<tr>
<td></td>
<td>Bar Graph</td>
<td>Minimum Value, Maximum Value, Foreground Color, Direction (Vertical, Horizontal), Orientation (Up, Down)</td>
</tr>
<tr>
<td></td>
<td>Text Data Link</td>
<td>Minimum Value, Maximum Value, Input Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: The character limit specified by the mask (e.g., ######) is not enforced. This might cause unexpected behavior during run time, depending on how the input is used.</td>
</tr>
<tr>
<td></td>
<td>Color</td>
<td>Type (By Limit, By Color), Change Limit, Color, Blink (Slow, Fast)</td>
</tr>
<tr>
<td></td>
<td>Visibility/Position</td>
<td>Visibility, Horizontal (Tag/Expression, Value Range, Position, Reference), Vertical (Tag/Expression, Value Range, Position, Reference), Slider</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: The tags configured for Horizontal and Vertical Tag/Expression are not updated until the user releases the object.</td>
</tr>
<tr>
<td></td>
<td>Resize</td>
<td>Height (Tag/Expression, Value Range, Size Range, Reference), Width (Tag/Expression, Value Range, Size Range, Reference)</td>
</tr>
<tr>
<td></td>
<td>Rotation</td>
<td>Range Minimum, Range Maximum, Degrees Start, Degrees End, Reference, Offset (x, y), Counter Clockwise</td>
</tr>
</tbody>
</table>
Hints, e-sign, and translation are not supported on any screen objects or animations. Also, the virtual keyboard is not used; if input is required, then the client’s own keyboard — on-screen for smartphones and tablets, physical for other computers — is used.

Examples of supported screen objects and animations

**Built-in functions**

Many but not all of InduSoft Web Studio’s built-in functions are supported in project screens. To see if a specific function is supported, please refer to the documentation for that function. The function will be marked either “Supported”, “Not Supported”, or “Executed on Server”, and there might be additional notes describing how the function is executed in Mobile Access.

Functions that are marked “Executed on Server” are executed via remote procedure call (RPC) on the project runtime server. While these functions are supported, you should avoid using a large number of them in any project screens that you include in Mobile Access. The extra communication required between server and client can affect run-time performance.

Also, please note that because Database/ERP functions are executed on the server, they affect server tags (i.e., tags with Server scope) rather than local tags (i.e., tags with Local scope), and that might result in unexpected behavior when multiple clients try to execute the functions at the same time. As such, you should avoid specifying optional parameters that take tag names — for example, if you call the function DBCursorOpen, do not specify the parameters optStrTags and optStrTagError.

For more information, see Appendix: Built-in Scripting Language on page 648. More functions will be supported in future releases of InduSoft Web Studio.

**VBScript**

Most VBScript interfaces — including the Graphics Script, the Screen Script, and Command animations on screen objects — are supported in project screens in Mobile Access. All regular VBScript functions, commands, and syntax are fully supported. However, InduSoft Web Studio’s built-in functions — which can be called in VBScript by prefacing them with $ — and other features have the following limitations:

**Built-in functions**

Many but not all of InduSoft Web Studio’s built-in functions are supported at this time; see “Built-in Functions” above.

**Dialog boxes in the Graphics Script**

Functions that display dialog boxes in the Mobile Access web interface cannot be called from the Graphics Script. They can be called from the Screen Script or Command animations. Also, displaying a dialog box will cause the web interface to stop updating, until the user closes the dialog box.

Functions that display dialog boxes include the VBScript function MsgBox and the IWS built-in function Logon.

**Calling procedures declared in the Graphics Script**
Normally, procedures that have been declared in the Graphics Script can be called from other VBScript interfaces using the syntax `Graphics.procedurename`. This feature is not supported in Mobile Access. Any procedures that you want to call like this should be declared in Global Procedures.

**System tags GroupCNFLoLevel and GroupCNFHiLevel**

The pre-defined system tags `GroupCNFLoLevel` and `GroupCNFHiLevel` are not supported in Mobile Access. In fact, these tags have been deprecated; if you want to check the security levels to which the user has access, use the function `CheckSecurityLevel` instead.

**Tag fields**

The following tag fields are supported on each type of project tag in Mobile Access:

<table>
<thead>
<tr>
<th>Tag Field</th>
<th>Supported on Type...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boolean</td>
</tr>
<tr>
<td>Name</td>
<td>#</td>
</tr>
<tr>
<td>MemberName</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>#</td>
</tr>
<tr>
<td>Index</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>#</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>#</td>
</tr>
<tr>
<td>Quality</td>
<td>#</td>
</tr>
<tr>
<td>Blocked</td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>UnitDiv</td>
<td></td>
</tr>
<tr>
<td>UnitAdd</td>
<td></td>
</tr>
<tr>
<td>DisplayValue</td>
<td></td>
</tr>
<tr>
<td>DisplayMin</td>
<td></td>
</tr>
<tr>
<td>DisplayMax</td>
<td></td>
</tr>
<tr>
<td>DisplayUnit</td>
<td></td>
</tr>
<tr>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>Lo</td>
<td>#</td>
</tr>
<tr>
<td>HiHi</td>
<td></td>
</tr>
<tr>
<td>LoLo</td>
<td>#</td>
</tr>
<tr>
<td>Rate</td>
<td></td>
</tr>
<tr>
<td>DevP</td>
<td>#</td>
</tr>
<tr>
<td>DevM</td>
<td>#</td>
</tr>
<tr>
<td>HiLimit</td>
<td>#</td>
</tr>
<tr>
<td>LoLimit</td>
<td></td>
</tr>
<tr>
<td>HIHILimit</td>
<td>#</td>
</tr>
<tr>
<td>LoLoLimit</td>
<td></td>
</tr>
<tr>
<td>RateLimit</td>
<td>#</td>
</tr>
</tbody>
</table>
About thin clients and mobile access

<table>
<thead>
<tr>
<th>Tag Field</th>
<th>Supported on Type...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boolean</td>
</tr>
<tr>
<td>DevPLimit</td>
<td>✓</td>
</tr>
<tr>
<td>DevMLimit</td>
<td>✓</td>
</tr>
<tr>
<td>DevSetPoint</td>
<td>✓</td>
</tr>
<tr>
<td>AlrStatus</td>
<td>✓</td>
</tr>
<tr>
<td>AlrDisable</td>
<td>✓</td>
</tr>
<tr>
<td>Ack</td>
<td>✓</td>
</tr>
<tr>
<td>UnAck</td>
<td>✓</td>
</tr>
<tr>
<td>AlrOffValue</td>
<td>✓</td>
</tr>
<tr>
<td>AlrOnValue</td>
<td>✓</td>
</tr>
<tr>
<td>AlrAckValue</td>
<td>✓</td>
</tr>
<tr>
<td>B0 ... B31</td>
<td>✓</td>
</tr>
</tbody>
</table>

Note: If you reference a Class tag without specifying a member (e.g., `$MyClass->fieldname`), the only supported fields are Name, Size, and Description.

The VBScript compiler for Mobile Access is stricter than the one used elsewhere in InduSoft Web Studio. It will not accept references to unsupported tag fields. For example, if you try to use `$MyString->B0` in VBScript anywhere else in InduSoft Web Studio, the compiler will accept the reference and simply return 0 or some other invalid value. On Mobile Access, however, you will receive a run-time error because B0 is not supported on String tags. You can check for such errors in the activity log.

For more information about tag fields, see List of tag properties on page 140.

Tag changes in the Event Logger

When a project tag is changed using VBScript in Mobile Access, that change will be logged in the Event Logger with the client’s IP address instead of its host name.

Date formats and time zones

While the server and clients may have their respective system times, Mobile Access always uses the server’s date format and time zone settings when it opens project screens. In the current release, it is not possible for the server and clients to have different settings, so you should not try to view project screens on clients with different settings as that might result in unexpected behavior during run time. (It is okay to use the Alarm, Process Values, and Trend controls in the web interface, because they do not include any VBScript that might be affected by this limitation.)

You can change the time zone setting on your client to match the server, but if that is not practical and you must view your project screens while in a different time zone, you should use Secure Viewer or Web Thin Client instead of Mobile Access.

Boolean tags

Mobile Access does not support the legacy method for handling Boolean tags (i.e., project tags of Boolean type) in VBScript. Boolean tags are always handled as if they have a numerical value of -1 for TRUE, to ensure compatibility with Boolean variables in VBScript. Editing your project file to change the property `VBBoolean` will not override this. For more information, see Boolean tags and Boolean variables on page 975.

Statement continuation with comments

In VBScript, you can use an underscore character to indicate that a statement is continued to the next line. Programmers often do this to make a long statement easier to read. For example:

```vbnet
MyArray = Array("FIRST_NAME", _
               "LAST_NAME", _
               "ADDRESS")
```
Without the underscore character, the end of the line would also be the end of the statement. The VBScript compiler in Mobile Access supports continuing a statement like this, except for the following limitation: in MsgBox statements and function calls, you cannot insert a comment after an underscore character. For example:

```
MyVar = MsgBox("Hello World!", _ 'This is a comment
65, _ 'This is another comment
"MsgBox Example")
```

This code would not be accepted by the VBScript compiler in Mobile Access, and the resulting compiler error could prevent a project screen from opening at all.

For more information about using VBScript in your project, see Overview of VBScript on page 943.

**Important features not supported**

Mobile Access supports only the features listed above and with the limitations mentioned. Among the features not supported, the following ones are most commonly used:

**Windows Embedded**

The Mobile Access Runtime software cannot be installed on Windows Embedded, and Mobile Access cannot be configured in projects that are set to run on Windows Embedded target systems.

**Selecting from a list of users to log on**

The Mobile Access Logon screen does not support selecting from a list of users. The user must know and type in their user name. For more information, see Log on to the Mobile Access web interface on page 586.

**Opening multiple screens or screen groups**

You cannot open multiple screens or screen groups in the Mobile Access web interface. If you need to display a supplemental screen of some kind, try one of the following:

- Configure a group of objects to act as a supplemental screen, and then apply the Visibility animation to that group in order to show/hide it; or
- Call a function to show an appropriate dialog box. Such functions include the VBScript functions `MsgBox` and `InputBox` and the IWS built-in function `ShowMessageBox`.

**Tip:** Avoid creating a script that both shows a dialog box and opens or closes a project screen. The Mobile Access web interface can display only one screen at a time, and opening a new screen will automatically close the previous one. Depending on the order of execution, this can lead to unexpected behavior such as a dialog box being orphaned or not being shown at all.

**Data objects**

The Alarm/Event Control object, the Trend Control object, and the Grid object are not supported in project screens in Mobile Access. For alarms and trends, you can use the Alarm and Trend controls that are native to Mobile Access. For grids, you can use the database interface to populate project tags with the appropriate data, and then display the tag values in your own, custom-made grid.

**Embedded bitmaps**

Embedded bitmaps are not supported in project screens in Mobile Access. If you paste bitmaps into your screens, make sure that they are saved in separate files. For more information, see Paste a bitmap image into a screen on page 188.

**Image formats**

Support for certain image formats varies from browser to browser. If you use an unsupported image in your project screen, the browser will not be able to render that image when you view the screen in Mobile Access. For example, Internet Explorer can render Windows Metafiles, but Google Chrome cannot.

For the best performance across all browsers, try to use "web-compatible" image formats such as GIF, JPG, and PNG wherever possible.

**Additional options for Driver and OPC communication**
Some additional options for Driver and OPC communication are not supported in Mobile Access. First, in the project settings, the option **Send last state** is not supported. Mobile Access automatically uses the option **Send every state**, with a fixed buffer size of 5. For more information, see Communication tab on page 106.

Second, in OPC communication worksheets, for the **Scan** setting on each tag configured in a worksheet, the option **Screen** (to scan the tag only while a screen that uses the tag is open) is not supported. Projects viewed in Mobile Access should use the option **Always** (to always scan the tag while the worksheet is enabled). For more information, see Configuring an OPC Client connection to an OPC Server on page 413. As an alternative, you can configure an entire communication worksheet to be enabled only while a screen is open.

**Compressing files for faster downloads**

In projects that are accessed by Secure Viewer and Web Thin Client, you can choose to compress the screen files to make them faster to download over slow connections. (The option **Enable File Compression** is located on the Web tab of the project settings. For more information, see Web tab.) This feature is not supported in Mobile Access. If you try to view a project screen that has been compressed like this, the screen might behave unexpectedly and you might see messages in the activity log that say tags or objects do not exist.

Tip: If your project requires a feature that is currently not supported in Mobile Access, consider using Secure Viewer or Web Thin Client instead. For more information, see About web thin clients on page 533.

More features will be supported in future releases of InduSoft Web Studio.

**Tips for Mobile Access development and run time**

These are general tips for developing projects for Mobile Access, as well as for using the Mobile Access web interface during run time.

**Make sure that you do not use unsupported features**

Mobile Access currently supports many but not all features of InduSoft Web Studio. If you use an unsupported feature in a project screen, you might see unexpected behavior when you view that screen in the Mobile Access web interface. Such behavior can range from incorrect tag changes and function calls to objects, animations, or scripts that do not work at all.

Make sure that all of the screen objects, animations, background tasks, VBScript, and built-in functions that you use are included in the list of supported features. For more information, see List of features supported in project screens in Mobile Access on page 563.

If you do use unsupported features, they will be reported in the Output Window in the development environment when you either verify your project or publish your screens as HTML.

**Make sure you are using an HTML5-compatible browser**

To use the Mobile Access web interface, you must have a modern, HTML5-compatible web browser. Different browsers have different levels of support for HTML5, however, and many older browsers do not support it at all. This can cause a wide variety of issues, from project screens not behaving as expected to the entire web interface not opening at all.

Therefore, the first thing you should do whenever you have issues with Mobile Access is make sure that you are using an HTML5-compatible browser, preferably one that we have validated for use with Mobile Access:

**List of validated browsers**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Phone 8</td>
<td>Internet Explorer (default browser)</td>
</tr>
<tr>
<td>Android Gingerbread (2.3.x)</td>
<td>Android Browser (default browser)</td>
</tr>
</tbody>
</table>

Note: Alarm, Process Values, and Trend controls only; Screens are not supported on this browser.
## Make sure your computer or device has enough resources

Viewing large or complex project screens in the Mobile Access web interface can be resource-intensive, and your computer or device might not have enough resources (e.g., processor, memory, bandwidth) to do the job regardless of how new it is, which operating system it runs, or which browser you use. If either the web interface in general or a specific project screen seems to perform unsatisfactorily, try viewing it on another computer or device. Also, particularly on mobile devices like smartphones and tablets, check to see if your device’s battery is low or if you have other apps open. Any or all of these factors can affect the performance of the web interface and give you a false impression of Mobile Access itself.

If, after checking these things, you still see unsatisfactory performance, you may need to “lighten” your project screens — that is, modify your screens to decrease the amount of resources that they require. Here are a few ways to do that:

- Try not to use large, high-resolution pictures in a screen, especially if you resize them in the screen editor after you place them. Replace them wherever possible with pictures that are properly scaled and resampled. Also, try decreasing the image resolution (e.g., from 300 DPI to 72 DPI) if full resolution is not necessary.

- Do not paste a picture or use a group of objects more than once in a screen, because each instance requires its own resources. Replace these pictures and groups with Linked Pictures and Linked Symbols, respectively.

- Make sure the project screen itself is properly sized for the computers or devices on which you plan to view it. It is a waste of resources to create, for example, a 2560-by-1440 screen for a 1136-by-640 smartphone. The screen is automatically scaled to fit the browser in which it is viewed, so nothing will be inadvertently cropped or hidden, and the user can zoom into a scaled screen to see smaller details, if necessary. (Please note that zooming works somewhat differently in mobile browsers versus desktop browsers.) An oversized screen, however, always takes more resources and is more difficult to use.

You can change the screen size in the Screen Attributes.

In the end, however, please keep in mind that the performance of the Mobile Access web interface is not an indicator of the performance of the InduSoft Web Studio project runtime itself. Regardless of what you see in your browser, your project should be running well everywhere else.

## Other tips for developing for Mobile Access

Here are some other tips for project and screen development:

### Do not enable file compression

In projects that are accessed by Secure Viewer and Web Thin Client, you can choose to compress the screen files to make them faster to download over slow connections. (The option Enable File Compression is located on the Web tab of the project settings. For more information, see Web tab.) This feature is not supported in Mobile Access. If you try to view a project screen that has been compressed like this, the screen might behave unexpectedly and you might see messages in the activity log that say tags or objects do not exist.

### Avoid tag synchronization when opening screens

If you use a large number of project tags in the VBScript sub-routines Screen_OnOpen or Graphics_OnOpen, your screens might take a long time to open or update. This is because the tag values — even for project tags with Local scope — must be synchronized between server and...
client when the scripts are executed, and that can take a long time if you have a slow client/server connection.

The solution is to use VBScript variables instead of project tags wherever possible. Variables exist only within the scripts where they are declared and used, so no synchronization is required. Otherwise, you can check the activity log to see which tags are being synchronized and when.

Do not use "Executed on Server" functions in FOR loops

Similar to the issue of tag synchronization that is described above, if you use a large number of functions marked "Executed on Server", your project screens might take a long time to update. This is because the function calls must be sent from the client to the server, and then the returned values must be sent from the server to the client.

This is especially true of functions called from within a FOR loop. The loop itself can be executed relatively quickly on the client, but the function calls might "stack up" as the client waits for the server to execute them.

If you experience any of the issues described above, you can use the activity log to troubleshoot your project screens. For more information, see Troubleshooting project screens in Mobile Access on page 595.

**Turning on IIS and installing Mobile Access Runtime**

To use Mobile Access with your IWS project, you must turn on Microsoft Internet Information Services (IIS) and then install the Mobile Access Runtime software.

You should also enable Secure Socket Layer (SSL) encryption in IIS, especially if you plan to access your project over the public Internet.

**TURN ON MICROSOFT IIS FOR THIN CLIENT ACCESS**

Turn on Microsoft Internet Information Services (IIS) and configure it with the correct settings to make your project accessible to thin clients over the network.

Please note that these instructions apply only to the following versions of IIS:

<table>
<thead>
<tr>
<th>Version</th>
<th>...on Operating System</th>
</tr>
</thead>
</table>
| IIS 7   | • Microsoft Windows Vista  
|         | • Microsoft Windows 7  
|         | • Microsoft Windows Server 2008  
|         | • Microsoft Windows Server 2008 R2 |

| IIS 8   | • Microsoft Windows 8  
|         | • Microsoft Windows 8.1  
|         | • Microsoft Windows Server 2012  
|         | • Microsoft Windows Server 2012 R2 |

IIS supports all features of InduSoft Web Studio, and it is robust enough to serve almost any IWS project in a production environment. It is the web server software that we recommend for most users, and to achieve the best performance during project run time, we strongly recommend that you use one of the versions listed above.

For information about installing and configuring earlier versions of IIS, see:

- IIS 6.0 on Windows XP Professional 64-bit or Windows Server 2003 ([technet.microsoft.com/library/cc785089](technet.microsoft.com/library/cc785089))

You need to turn on IIS only on the computer that will be your project runtime server. This may be the same computer that you are using to develop your project, particularly if you plan to test your project locally, but it does not have to be.
Note: You must have Administrator privileges on the computer in order to turn on and configure IIS, and you should be familiar with administering Microsoft Windows on a network.

For the sake of system security, IIS is turned off by default when the operating system is installed. To turn on and configure IIS in Windows:

1. Do one of the following:
   - Click the Start button, and then on the Start menu, click Control Panel; or
   - In Windows 8, swipe in from the right edge of the screen, click/tap Search, and then in the search box, type Control Panel. When Control Panel is displayed in the search results, click/tap it.

   The Control Panel window is displayed.

2. In the Control Panel window, click Programs, and then under Programs and Features, click Turn Windows features on or off.

   The Windows Features dialog box is displayed.

3. In the Windows Features dialog box, select Internet Information Services.

   IIS is selected with its default features, but you need to make sure that all of the features required by InduSoft Web Studio are also selected.

4. Expand Internet Information Services > World Wide Web Services > Application Development Features, and then do one of the following:
   - For IIS 7, make sure that all of the following features are selected: .NET Extensibility, ASP, ASP.NET, ISAPI Extensions, and ISAPI Filters.
   - For IIS 8, make sure that all of the following features are selected: .NET Extensibility 3.5, .NET Extensibility 4.5, ASP, ASP.NET 3.5, ASP.NET 4.5, ISAPI Extensions, and ISAPI Filters.
5. Expand Internet Information Services > World Wide Web Services > Common HTTP Features, and then make sure that Static Content is selected.
6. Click **OK**.
   IIS is turned on with the selected features, but you might be prompted to restart Windows to apply the changes.

7. After you have turned on IIS, you can use Internet Information Services (IIS) Manager to configure it. To open IIS Manager:
   a) In the **Control Panel** window, click **System and Security**, and then click **Administrative Tools**.
      The **Administrative Tools** window is displayed.
   b) In the **Administrative Tools** window, double-click **Internet Information Services (IIS) Manager**.

For more information about configuring IIS 7, go to: technet.microsoft.com/library/cc753433
For more information about configuring IIS 8, go to: technet.microsoft.com/library/hh831725

---

**Tip:** A web server typically operates on, or "listens to," a computer's TCP/IP port 80. Only one running process can listen to a given port, so if another process on your computer — for example, third-party SCADA software — is already listening to port 80, then it and the web server process may conflict with each other. You must either configure one of the processes to listen to a different port or use Task Manager to end the conflicting process. If you cannot identify the conflicting process, then in Windows, open Command Prompt and enter the following command to get a list of all networking processes:

```
netstat -a -o
```

---

**ENABLE SSL ENCRYPTION IN MICROSOFT IIS**

Enable Secure Socket Layer (SSL) encryption in Microsoft Internet Information Services (IIS) in order to secure communications between the web server and your thin clients.

Please note that these instructions apply only to the following versions of IIS:

<table>
<thead>
<tr>
<th>Version</th>
<th>...on Operating System</th>
</tr>
</thead>
</table>
| IIS 7   | • Microsoft Windows Vista
         | • Microsoft Windows 7
         | • Microsoft Windows Server 2008
         | • Microsoft Windows Server 2008 R2 |
| IIS 8   | • Microsoft Windows 8
         | • Microsoft Windows 8.1
         | • Microsoft Windows Server 2012
         | • Microsoft Windows Server 2012 R2 |

For information about enabling SSL on earlier versions of IIS, go to: support.microsoft.com/kb/299875

Also, before you begin this task, you should know whether you are going to use a signed or a self-signed certificate. Both types of certificates are explained below, but since this is not intended to be a complete discussion of Windows server administration, instructions are provided only for creating a self-signed certificate so that you can continue developing and testing your IWS project. For information about requesting a signed certificate, go to: technet.microsoft.com/library/cc732230

Your IWS project has a built-in **security system** that you can use to control who logs on and what access they have. It does nothing to secure the connection between the server and the client, however, so if your local network is insecure and/or you connect to your server over the Internet, then your communications can be intercepted and possibly compromised.

One way to secure the connection is to use Secure Socket Layer (SSL) encryption to encrypt the packets that are sent between the server and the client. When SSL is enabled on the server, the server offers a certificate that includes proof of the identity of the server and an encryption key. The client — in this case, your web browser — can either accept or reject the certificate, depending on whether it trusts the certificate. If the
certificate is trusted, then it is automatically accepted and SSL is turned on; in many web browsers, this is indicated by a padlock icon. If the certificate is not trusted, then an alert message is displayed and the user must choose whether to accept it or reject it.

The criteria for trusting a certificate is typically whether the certificate is signed by a known certificate authority (CA) and is unexpired. However, a signed certificate must be requested and purchased from a CA, so there is also an option to create a free, self-signed certificate. A self-signed certificate is a certificate signed by the server that is offering it, and as long as it is used only on a secure local network where you know and trust all of the other computers, it is sufficient for project development. (Again, for information about requesting a signed certificate, go to: technet.microsoft.com/library/cc732230)

**Note:** You should not use a self-signed certificate in a production environment.

To create a self-signed certificate and enable SSL encryption in IIS:

1. Do one of the following:
   - Click the **Start** button, and then on the **Start** menu, click **Control Panel**; or
   - In Windows 8, swipe in from the right edge of the screen, click/tap **Search**, and then in the search box, type **Control Panel**. When **Control Panel** is displayed in the search results, click/tap it.
   
   The **Control Panel** window is displayed.

2. In the **Control Panel** window, click **System and Security**, and then click **Administrative Tools**.
   
   The **Administrative Tools** window is displayed.

3. In the **Administrative Tools** window, double-click **Internet Information Services (IIS) Manager**.
   
   The **IIS Manager** window is displayed.

4. Create the self-signed certificate:
   a) In the **IIS Manager** window, in the **Connections** list on the left, select your server (typically your own computer).
   b) In **Features** view, double-click **Server Certificates**.
   c) In the **Actions** pane, click **Create Self-Signed Certificate**.
   d) On the **Create Self-Signed Certificate** page, in the **Specify a friendly name for the certificate** box, type a friendly name for the certificate (e.g., IWS), and then click **OK**.
   
   Your self-signed certificate is added to the list of server certificates.

5. Enable SSL for your web site:
   a) In the **Connections** list on the left, open your server, open **Sites**, and then select **Default Web Site**.
   b) In the **Actions** pane, click **Bindings**.
   c) On the **Site Bindings** page, click **Add**.
   d) On the **Add Site Binding** page, in the **Type** list, select **https**.
   e) In the **SSL certificate** list, select the self-signed certificate that you created.
   f) Click **OK** to close the **Add Site Binding** page, and then click **Close** to close the **Site Bindings** page.

6. Require clients to connect with SSL:
   a) In the **IIS Manager** window, in **Features** view, double-click **SSL Settings**.
   b) Select **Require SSL**.
   
   This step is optional. If you have problems connecting to the web site, then you may clear this option and try connecting without SSL.

7. Restart your web site with the new settings:
   a) In the **Connections** list on the left, select **Default Web Site** again.
   b) In the **Actions** pane, click **Restart**.

8. Close IIS Manager.
When you want to deploy your IWS project in a production environment, you should request a signed certificate and reconfigure IIS to use that certificate.

**INSTALL THE MOBILE ACCESS RUNTIME SOFTWARE**

Install the Mobile Access Runtime software on the same computer that hosts your IWS project runtime server.

Before you begin this task, Microsoft Internet Information Services (IIS) must be turned on on your server and the required ASP.NET features should be selected. The Mobile Access Runtime software installer will try to confirm that Microsoft IIS is turned on, and if it is not, it will abort the installation. The installer cannot confirm that the required features are selected, however, so you should confirm that yourself before you begin this task. For more information, see Turn on Microsoft IIS for thin client access on page 571.

The Mobile Access Runtime software is an add-on for Microsoft IIS that makes it possible for other computers and mobile devices to access the Mobile Access web interface. You do not need to install the software on your own computer in order to develop IWS projects that include Mobile Access; the necessary tools are always available in the InduSoft Web Studio development environment. However, you do need to install the software on the computer that hosts the InduSoft Web Studio project runtime server, whichever computer that happens to be.

**Note:** Mobile Access Runtime cannot be installed on Windows Embedded, and the Mobile Access feature is not supported in projects that are developed to run on Windows Embedded target systems.

There are two ways to install Mobile Access Runtime. First, you might have selected it as an installable feature when you installed the full InduSoft Web Studio software. If that is the case, you can skip the rest of this task. For more information, see Install the full InduSoft Web Studio software on page 37.

Second, you can use the separate Mobile Access Runtime software installer (MobileAccessSetup.exe) to install it after the fact. To do this:

1. On the computer where you want to install the software, make sure that InduSoft Web Studio is also installed and Microsoft IIS is turned on.

2. Locate the Mobile Access Runtime software installer at in the InduSoft Web Studio folder at C:\Program Files\InduSoft Web Studio v7.1\Bin\MobileAccessSetup.exe

3. Run the installer and follow its instructions.

To confirm that the software was successfully installed, run your IWS project and then try to log on to the Mobile Access web interface. For more information, see Log on to the Mobile Access web interface on page 586.

If the web interface fails to load — that is, if the Mobile Access Logon screen is not displayed at all — use IIS Manager to confirm that your web server has started and the Mobile Access application pool (StudioMobileAccessPool) has been configured to use .NET Framework version 4.0. For more information, see Turn on Microsoft IIS for thin client access on page 571.

**SPECIFY THE ADDRESS OF THE NETWORK ROUTER FOR MOBILE ACCESS**

In some cases, you need to edit the Mobile Access Runtime software configuration file in order to specify the address of the network router that manages traffic on your network.

Before you begin this task, you must have already installed Mobile Access Runtime on your IWS project server. The file that you need to edit is included with that software. For more information, see Install the Mobile Access Runtime software on page 576. You should also be familiar with XML and how to edit an XML-formatted configuration file.

If the project server is located on a private network behind a network router that uses network address translation (NAT) to manage traffic between the private network and the Internet, you need to do two things.

First, you need to configure the router to forward connections from the Internet to the project server. This is typically done by opening a port on the router and then mapping that port to the correct server on the network. If you cannot do this yourself, please contact your network administrator. When it is done, however, note the public IP address of the router and the number of the port that was mapped.

Second, you need to configure Mobile Access Runtime so that it knows the address of the router. It uses this information to set up the Mobile Access web interface in the browser on the client, after the user logs on. Without this information, the client cannot maintain its connection to the server.

To specify the address of the network router:
1. Find the Mobile Access Runtime software configuration file (Web.config) on your project server. Assuming that both Microsoft IIS and Mobile Access Runtime have been installed in their default locations, the configuration file should be located at C:\inetpub\wwwroot\MA\Web.config

2. Open the configuration file with a text editor (e.g., Notepad).

3. In the configuration file, find the node named **appSettings**. For example, in the default version of Web.config that is installed with the Mobile Access Runtime software:

   ```xml
   <appSettings>
   <add key="MinMaxMessage" value="Value out of range Min:{0} Max:{1}"/>
   <add key="InvalidFloatingPointValue" value="Invalid floating point value"/>
   <add key="DataServerIPAddress" value="127.0.0.1"/>
   <add key="TCPPort" value="1234"/>
   <!--<add key="RouterIP" value="200.128.128.0"/>-->
   <add key="RouterIP" value="200.128.128.0"/>
   <add key="RouterPort" value="3040"/>
   </appSettings>
   ```

4. Uncomment the keys named **RouterIP** and **RouterPort**. In other words, delete the brackets (<!-- -->) around the keys that mark them as comments. For example:

   ```xml
   <appSettings>
   <add key="MinMaxMessage" value="Value out of range Min:{0} Max:{1}"/>
   <add key="InvalidFloatingPointValue" value="Invalid floating point value"/>
   <add key="DataServerIPAddress" value="127.0.0.1"/>
   <add key="TCPPort" value="1234"/>
   <add key="RouterIP" value="200.128.128.0"/>
   <add key="RouterPort" value="3040"/>
   </appSettings>
   ```

5. Replace the key values: for **RouterIP**, type the public IP address of the network router, and for **RouterPort**, type the number of the port that was mapped to the project server. For example:

   ```xml
   <appSettings>
   <add key="MinMaxMessage" value="Value out of range Min:{0} Max:{1}"/>
   <add key="InvalidFloatingPointValue" value="Invalid floating point value"/>
   <add key="DataServerIPAddress" value="127.0.0.1"/>
   <add key="TCPPort" value="1234"/>
   <add key="RouterIP" value="66.94.230.43"/>
   <add key="RouterPort" value="3506"/>
   </appSettings>
   ```

6. Save and close the configuration file. Remember that the user will also need the IP address and port number in order to log on. For more information, see Log on to the Mobile Access web interface on page 586.

### Configuring the Mobile Access web interface

Use the Mobile Access Configuration worksheet to configure the Mobile Access web interface for your IWS project. Your project’s web interface consists of one or more "areas", which are typically organized by location, system, or machine. Each area has an Alarm control, a Process Values control, a Trend control, and a Screens
control. These controls appear in the web interface as green tiles, and when you click/tap one of these tiles, it opens a new page for that control.

An example of the Mobile Access web interface

Each area can also have one or more “sub-areas”, which appear in the web interface as orange tiles. When you click/tap one of these tiles, it opens a new page for that sub-area.

You can insert as many areas and sub-areas as you want; the web interface is automatically expanded to accommodate them. You use the Mobile Access Configuration worksheet to insert and configure areas.

Configuring the Mobile Access web interface for your project

The Alarm, Process Values, Trend, and Screens controls are functionally similar to their corresponding screen objects:

**Alarm**

The Alarm control is similar to the Alarm/Event Control screen object. It displays active alarms and allows the user to acknowledge them.

**Process Values**
The Process Values control is similar to the Symbols library, in that it uses various pre-made widgets (i.e., gauges and switches) to graphically represent project tag values. It can also allow the user to change the values during run time, depending on how you configure the widgets.

**Trend**

The Trend control is similar to the Trend Control screen object. It graphs the changes in process values over a specified duration, and it also saves a history of those changes.

**Screens**

The Screens control presents project screens that you have selected to include in the web interface. In most cases, selected screens function the same as screens published for Secure Viewer and Web Thin Client, but there are some limitations. For more information, see List of features supported in project screens in Mobile Access on page 563.

**INSERT A NEW AREA IN THE MOBILE ACCESS TREE VIEW**

Insert a new node, or "area", in the Mobile Access tree view in order to create a new screen in the Mobile Access web interface.

The site map, so to speak, of the Mobile Access web interface is determined by the tree view on the left side of the Mobile Access Configuration worksheet. The root node of the tree, named Main, corresponds to the home screen of the web interface. If you wish, you can configure the settings for Main so that it includes all of the project tags that you want to display in the web interface. You are not required to add to the tree, and if you choose not to, you can skip this task and proceed to configuring the settings for Main.

However, if you want organize your project tags in some way — for example, by machine, by process, or by facility — you can insert additional nodes, or "areas", in the tree view. Each area has its own settings and is represented by its own screen in the Mobile Access web interface. Areas can have as many sub-areas as you want, as many levels deep as you want. The structure of the tree view, and therefore the site map of the web interface, is entirely up to you.

Keep in mind, however, that the structure of the tree view determines how the user must navigate the Mobile Access web interface during run time. You may choose to have all of the areas together on the same level, but if you do, then the user must pan/scroll a lot to move between the areas. Alternatively, you may choose to create many levels of areas and sub-areas, but if you do, then the user must click/tap down through those levels and then back up again. In short, it is important to keep your web interface logically organized and easy to navigate, and as such you should manually outline your tree view before you begin inserting areas.

To insert an area in the Mobile Access tree view:

1. Do one of the following:
   - On the Project tab of the ribbon, in the Thin Clients group, click Mobile Access; or
   - On the Graphics tab of the Project Explorer, double-click Thin Clients > Mobile Access.
The *Mobile Access Configuration* worksheet is opened for editing, with the tree view displayed on the left.

![An example of the Mobile Access tree](image)

2. Carefully determine where in the tree view you want to insert the new area, because once you have inserted it, it is not possible to move it using the Mobile Access Configuration worksheet.

3. Right-click on the existing area in which you want to insert the new area, and then on the shortcut menu, click **Insert Area**. A *New Area* dialog box is displayed.

4. In the **Area name** box, type the name of the new area, and then click **Add**. The new area is inserted in the tree view.

After you have inserted the new area, you can configure the settings for that area.

To delete an area that you have inserted, right-click the area, and then on the shortcut menu, click **Delete Area**. Please note that when you delete an area, you also delete all of the sub-areas that it contains. You cannot delete **Main**.

**Note:** Your Mobile Access configuration is saved as an XML file in your project folder at:

`[...]\Documents\InduSoft Web Studio v7.1 Projects\projectname\Web\MobileAccess.sma`

If necessary, you can manually edit this file to make changes that cannot be made in the Mobile Access Configuration tool. For more information, please contact Technical Support.

**CONFIGURE THE MOBILE ACCESS AREA SETTINGS**

Each area in the Mobile Access tree view has its own area settings. Configure these settings to determine which project tags and screens to display in the Mobile Access web interface.

For more information about the Mobile Access tree view and how it determines the site map of the Mobile Access web interface, see #unique_489.

To configure the settings for a selected area:

1. If the *Mobile Access Configuration* worksheet is not already open, do one of the following:
   - On the **Project** tab of the ribbon, in the **Thin Clients** group, click **Mobile Access**.
   - On the **Graphics** tab of the Project Explorer, double-click **Thin Clients > Mobile Access**.

   The *Mobile Access Configuration* worksheet is opened for editing, with the tree view displayed on the left.

2. In the tree view, select the area for which you want to configure settings.
The *Area Settings* worksheet for that area is displayed.

<table>
<thead>
<tr>
<th>General</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Main</td>
<td>Label:</td>
</tr>
<tr>
<td>Access Level</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alarms</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Group:</td>
<td></td>
<td>Priority From: 0</td>
</tr>
<tr>
<td>Selection:</td>
<td></td>
<td>Priority To: 255</td>
</tr>
</tbody>
</table>

- **Trend and Process Values**
  - **Tag Name**: AlarmDig[1], AlarmDig[2], AlarmAnalog[1], TrendLin, TrendRampUp, TrendWaveUp, TrendOnOff, TrendCox, TrendRampDown
  - **Label**: Digital #1, Digital #2, Analog #1, Level, Temperature, Demand, Speed, Pressure, Pump
  - **Write**: ✓
  - **Trending**: ✓
  - **Min**: 0
  - **Max**: 1
  - **Min**: 0
  - **Max**: 1
  - **Widget**: Switch
  - **Color**: 

- **Screens**
  - **Screen**: Animations
  - **Label**: Animations

Some objects and functions are not supported in screens; please see the documentation for more details. You must also save your screens as HTML.

3. In the **General** section of the worksheet, configure the general settings for the area.
   a) In the **Name** box, type the name of the area as it should appear in the tree view.
   b) In the **Label** box, type the area label that should be displayed in the Mobile Access web interface during run time.
   c) In the **Access Level** box, type the minimum security level that the user must have in order to access the area during run time.

4. In the **Alarms** section of the worksheet, specify which alarms should be displayed in this area’s Alarm control during run time.
   By default, all areas of the Mobile Access web interface will display all alarms that are configured in your project’s Alarm worksheets. However, you can filter the alarms by group, by selection, or by priority in order to display only the alarms that are relevant to this area. The easiest way to do this is to configure a separate Alarm worksheet/group for each area and then filter by those group numbers, but if you configured your project’s alarms long before you configured the areas of your Mobile Access web interface, then they may not correspond. If that is the case, then you can use the other settings. For more information about group, selection, and priority, see **Alarm worksheet** on page 280.
   a) In the **Group** box, type the number of the Alarm group(s) that you want to display in this area.
You can specify more than one group by using commas and hyphens. For example, if you type...

5-10, 60, 80–90

...you will display groups 5 through 10, group 60, and groups 80 through 90 in this control.
The **Group** box will validate as you type; if you type an invalid group or groups, then the box will be bordered in red.
If you leave this box empty, no filtering will be done by Alarm group.

b) In the **Selection** box, type the selection alias(es) of the specific alarms that you want to display in this area.
You can specify more than one selection alias by using commas. For example...

**AliasA, AliasB, AliasD**

If you leave this box empty, no filtering will be done by selection alias.

c) In the **Priority From** and **Priority To** box, type the priority range of the specific alarms that you want to display in this area.
If you leave the default values of 0 and 255 (i.e., the maximum range), no filtering will be done by priority.

d) Select **Show process values alarms only** if you only want to show alarms for the process values that are actually configured for this area (see below).
If you select this option, it will override all of the other alarm filter settings.

5. In the **Trend and Process Values** section of the worksheet, configure the process values that you want to display in this area and how they should be displayed. For each row, do the following.

a) In the **Tag Name** box, type the name of a project tag that you want to display as a process value.
You can also double-click in the box to open the Object Finder.

b) In the **Label** box, type a simple label for the project tag.

c) Select **Write** if you want to be able to write new values to the tag by manipulating its widget (e.g., slide its gauge, toggle it switch) during run time.

d) Select **Trending** if you want the values to be graphed in the Trend control.

e) In the **Min** and **Max** boxes, type the minimum and maximum for the process value.

Please note that these are not hard limits on the process value. The actual value of the project tag can exceed both the minimum and maximum, depending on how the value is calculated during run time. It simply will not be displayed, because it is literally off the scale. Instead, when the actual value is less than the minimum, the minimum is displayed, and when the actual value is greater than the maximum, the maximum is displayed.

In the Process Values control, **Min** and **Max** are used to determine the scale of the selected widget.
In the Trend control, **Min** and **Max** are used to calculate the percentage. (For example, if **Min** is 10, **Max** is 20, and the actual value is 16, then the percentage is 60%).

f) In the **Widget** list, select the type of widget that should be used to represent the process value in the Process Values control.
The following widgets are available.
<table>
<thead>
<tr>
<th>Option</th>
<th>Tag Types</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circular Gauge</td>
<td>Integer, Real</td>
<td><img src="image" alt="Circular Gauge" /></td>
</tr>
<tr>
<td>Semi-Circular Gauge</td>
<td>Integer, Real</td>
<td><img src="image" alt="Semi-Circular Gauge" /></td>
</tr>
<tr>
<td>Horizontal Gauge</td>
<td>Integer, Real</td>
<td><img src="image" alt="Horizontal Gauge" /></td>
</tr>
<tr>
<td>Switch</td>
<td>Boolean</td>
<td><img src="image" alt="Switch" /></td>
</tr>
<tr>
<td>Text Box</td>
<td>any</td>
<td><img src="image" alt="Text Box" /></td>
</tr>
</tbody>
</table>

If you select **None**, the process value will not be displayed in the Process Values control.

**g)** In the **Widget Size** list, select how large the widgets should be displayed. Larger widgets are clearer and easier to use on mobile devices, but they also take up more screen space.

**h)** In the **Write Access Level** box, type the minimum security level that the user must have in order to write new values to project tags.
6. In the **Screens** section of the worksheet, specify which project screens should be displayed in this area’s Screens control during run time. For each row, do the following.
   a) In the **Screen** column, click the list menu, and then select a project screen.

   **Tip:** The list menu should include all project screens that have been published for web thin clients. If you do not see the screen(s) that you want to select, do one of the following:
   - Verify your project (i.e., on the **Home** tab of the ribbon, in the **Tools** group, click **Verify**); or
   - Re-publish your project screens for web thin clients (i.e., click the Application/File button in the top-left corner of the development environment, and then on the **File** menu, click **Publish > Save All As HTML**).

   Please note that screen names should not include spaces. If a screen name does include a space, first save that project screen with a new file name and then re-publish it for web thin clients.

   b) In the **Label** column, type a label for the project screen that you selected.

   This label is displayed only in the Mobile Access web interface; it is not saved with the original screen file.

   All project screens that have been published for web thin clients can also be selected for Mobile Access. However, not all screen objects and animations are fully supported by Mobile Access at this time, so make sure that you test the selected screens before you deploy your project. For more information, see **List of features supported in project screens in Mobile Access** on page 563.

   Also, with regards to project security, the **Disable** and **Security** settings are enforced on all screen objects and animations, but the **E-sign** setting is not.

7. Save and close the **Mobile Access Configuration** worksheet.

**CONFIGURE THE MOBILE ACCESS GLOBAL SETTINGS**

Configure the Mobile Access global settings in order to set certain default values — such as alarm columns and colors, trend control duration, and update rates — that will apply to all areas of the Mobile Access web interface.

To configure the global settings:

1. Do one of the following:
   - On the **Project** tab of the ribbon, in the **Thin Clients** group, click **Mobile Access**.
   - On the **Graphics** tab of the Project Explorer, double-click **Thin Clients > Mobile Access**.

   The **Mobile Access Configuration** worksheet is opened for editing.

2. Click the **Global Settings** tab of the worksheet.
The *Global Settings* worksheet is displayed.

3. In the *Alarm Control* area, select and arrange the columns that all of the Alarm controls in the Mobile Access web interface should display by default.

   ![](Global_Settings_worksheet.png)

   **Note:** These settings can be overridden by the user in the client-side settings.

   The available columns are listed on the left. The displayed columns are listed on the right. For more information about what each column means, see *Alarm/Event Control object* on page 297.

   a) To display a column, select it in the list of available columns and then click the **>>** button.

   b) To hide a column, select it in the list of displayed columns and then click the **<<** button.

   c) To arrange the displayed columns, select a column and then click **Move Up** or **Move Down**.

   The order in which the columns are listed here is the order in which they will be displayed, from left to right, in the alarm control.

4. Select the alarm colors: for each alarm state (Active, Acknowledged, Normalized), click the color picker and then select a new color, if you wish.

5. In the *Trend Control* area, in the *Default duration* box, type the number of seconds that all of the Trend controls in the Mobile Access web interface should display by default.

   **Note:** This setting can be overridden by the user in the client-side settings.

6. In the *Update Rates* area, in the *Alarm*, *Trend*, and *Process Values* boxes, type how often (in milliseconds) each type of page should update its data during run time.

   Please note that the more quickly you update the pages, the more of a load you are putting on your network and the project runtime server. In most cases, you should leave the default values.

7. In the *Log* area, in the *Log level* box, type a number from 1 to 5 for the level of log messages that you want to display for each web interface.

   Mobile Access automatically generates an activity log for the web interface. This log is similar to but separate from the project runtime log that is displayed in the *Output window*; it comprises only messages that describe the behavior of project screens in the web interface. Level 1 (default) is the least detailed and verbose, level 5 is the most detailed and verbose. For more information, see *Troubleshooting project screens in Mobile Access* on page 595.
Navigating the Mobile Access web interface

Navigate the Mobile Access web interface as you configured it for your IWS project.

Before you can use the web interface, the Mobile Access Runtime software must be properly installed and configured on Microsoft IIS your IWS project must be running.

The home page of the web interface shows the top-level "Main" area and the other second-level areas that you configured in the Mobile Access Configuration worksheet. The page is automatically created as wide as it needs to be to accommodate all of the areas, but that means some of the page may be out of view depending on the size of your browser window or mobile device.

Each area has an Alarm control, a Process Values control, a Trend control, and a Screens control, represented by the green buttons. Some areas may also have sub-areas, if that is how you structured your web interface in the Mobile Access Configuration worksheet. Sub-areas are represented by orange buttons.

To navigate the Mobile Access web interface:

1. Pan/scroll left or right to find the area that you want.
2. Click/tap a green control button to access that control, or click/tap an orange sub-area button to access that sub-area.
   The button spins to show that it was clicked/tapped, and then the page for the selected control or sub-area is displayed.
3. On any page other than the home page, click/tap the Return button to return to the previous page.

LOG ON TO THE MOBILE ACCESS WEB INTERFACE

Log on to the Mobile Access web interface to view the configured alarms, process values, trends, and project screens during run time.

Please note that the Mobile Access web interface uses HTML5 (including CSS3 and AJAX) for its animated effects and real-time data exchange. That means you must use an HTML5-compatible browser to access the web interface. Most modern browsers are HTML5-compatible, but browsers change constantly and the HTML5 specification itself is still being improved, so we cannot provide a comprehensive list of platforms and browsers on which Mobile Access might be used. We can, however, provide a list of the specific platforms and browsers on which we have validated Mobile Access:

List of validated browsers

<table>
<thead>
<tr>
<th>Platform</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Phone 8</td>
<td>Internet Explorer (default browser)</td>
</tr>
<tr>
<td>Android Gingerbread (version 2.3.x)</td>
<td>Android Browser (default browser)</td>
</tr>
</tbody>
</table>
You can open only one session of the Mobile Access web interface per browser. You cannot open sessions in additional tabs or even additional instances of the same browser. So, for example, if you want to open two sessions of the web interface on the same computer, you must open one in Google Chrome and the other in Internet Explorer.

To log on to your project’s Mobile Access web interface:

1. On the client computer or device, in the web browser, go to https://address/ma/
   - If the project server is a computer on your local network, address is the Windows host name (or IP address, if you know it) of that computer. For example, https://development-pc/ma/
   - If the project server is a computer on the Internet, address is the full domain name (or IP address, if you know it) of that computer. For example, https://scada.ourcompany.com/ma/
   - If the project server is located on a private network behind a network router, address is the public IP address of that router and the number of the port that has been mapped to the project server. For example, https://200.128.128.0:3040/ma/

   **Note:** The network router must be configured to do port forwarding, and the Mobile Access Runtime software must be told about the router. For more information, see Specify the address of the network router for Mobile Access on page 576.

You must include the prefix https in order to connect securely via SSL. You may choose not to connect via SSL (if the server allows it), but we do not recommend it. For more information, see Enable SSL encryption in Microsoft IIS on page 574.

The **Mobile Access Logon** screen is displayed.

![Mobile Access Logon screen](image)
the certificate on your computer or device. The procedure to do this varies by operating system, so for more information, please consult the operating system’s own documentation.

2. Type your user name and password in the respective boxes, and then either press Return on your keyboard or click/tap the green arrow. You are logged on to the Mobile Access web interface and the home screen is displayed.

![Example of the Mobile Access home screen](image)

**Note:** If the project is not currently running, you will not be able to log on and an alert message will be displayed. Please keep in mind that the project runtime and the web server are two separate processes, so you may be able to go to the logon page even when the project is not running.

To log off of the Mobile Access web interface, click/tap the Return button to return to the home screen (you may need to do this more than once if you are in a sub-area), and then on the home screen, click Log Off.
USE THE ALARM CONTROL

Use the Alarm control to view and acknowledge alarms.

The Alarm control is similar to the [Alarm/Event Control screen object](#). It displays online alarms and allows the user to acknowledge them.

<table>
<thead>
<tr>
<th>Activation Time</th>
<th>Type</th>
<th>Message</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:01:00</td>
<td>Hi</td>
<td>WT #1: Wind Turbine fault</td>
<td>1</td>
</tr>
<tr>
<td>14:01:00</td>
<td>Hi</td>
<td>WT #2: Wind Turbine fault</td>
<td>1</td>
</tr>
<tr>
<td>06:07:19</td>
<td>Hi</td>
<td>Level too high (LUT-101)</td>
<td>95</td>
</tr>
<tr>
<td>06:07:28</td>
<td>Hi</td>
<td>Level too high (LUT-102)</td>
<td>95</td>
</tr>
<tr>
<td>06:08:11</td>
<td>Lo</td>
<td>Level too high (LUT-103)</td>
<td>95</td>
</tr>
<tr>
<td>06:09:03</td>
<td>Hi</td>
<td>High pressure (PIT-101)</td>
<td>60.824</td>
</tr>
</tbody>
</table>

By default, active alarms are written in red, acknowledged alarms are written in green, and normalized alarms are written in blue. However, you can change these colors in the Global Settings tab of the Mobile Access Configuration worksheet.

The grid columns are also similar to those in the Alarm/Event Control screen object. The following columns are included by default:

**Activation Time**

The time when the alarm became active.

**Type**

The type of alarm (e.g., Hi, Lo, HiHi, LoLo).

**Message**

The message that was displayed when the alarm became active.

**Value**

The current value of the project tag.

Like the default alarm colors, you can change these default columns in the Global Settings tab of the Mobile Access Configuration worksheet. However, those changes will apply to the entire Mobile Access website. If you only want to change the columns for a specific Alarm control, during a specific user session, see below.

To use the Alarm control:

1. To acknowledge all currently active alarms:
   a) Click/tap **Ack All**. The **Ack** screen is displayed.
   b) Type a comment that will be saved with the acknowledged alarms, and then click/tap **Confirm**. The alarms are acknowledged.

2. To acknowledge only selected alarms:
   a) In the * column of the grid, select the alarms that you want to acknowledge.

   **Note:** The Alarm control will not be updated while you have alarms selected.

   b) Click/tap **Ack ***.
The Ack screen is displayed.

c) Type a comment that will be saved with the acknowledged alarms, and then click/tap **Confirm**.

The alarms are acknowledged.

3. To change the grid columns for a specific Alarm control:
   a) Click/tap the **Settings** button.
      The **Settings** screen is displayed with a list of all of the available grid columns and a toggle switches for each column.
   b) Click/tap the switches to turn the columns on or off.
   c) Click **Alarm** to apply your changes and return to the previous screen.

4. To return to the home screen, click/tap the **Return** button.

**USE THE PROCESS VALUES CONTROL**

Use the Process Values control to view tag values as graphical widgets and also to update selected tags.

The Process Values control is similar to the Symbols library. It uses various pre-made widgets (i.e., gauges and switches) to graphically represent project tag values. It can also allow the user to change the values during run time, depending on how you configure the widgets.

![An example of the Process Values control](image)

All widgets are continuously updated to show the current values of their associated project tags.

If a widget is highlighted blue, then its associated project tag is writable. That means the user can use the widget to set a new value for the tag.

To use the Process Values control:

1. **To change a value:**
   a) Click/tap the widget.
      The widget must be highlighted blue, to indicate that the tag is writable.
      A new screen with an enlarged version of the widget is displayed.
   b) Either manipulate the widget (i.e., toggle the switch, move the gauge) to set the new value, or type the new value in the text box below the widget.
   c) Click/tap **Write**.
      The new value is written to the tags database.

2. **To return to the home screen,** click/tap the **Return** button.
USE THE TREND CONTROL

Use the Trend control to view trend graphs of selected process values.

The Trend control is similar to the Trend Control screen object. It graphs the changes in process values over a specified duration, and it also saves a history of those changes.

The X-axis of the graph is time, and the Y-axis is the value of the tags. The legend below the trend graph includes the following columns:

**Check Box**
Select to show the process value on the trend graph, or clear to hide it.

**Label**
The name of the process value. Please note that this may be different from the original tag name, depending on how you configure it in the Mobile Access Configuration worksheet.

**Value**
The current value of the project tag.

**Min and Max**
The Min and Max settings from the area settings. These are used to calculate the percentage on the Y-axis. For example, if Min is 10, Max is 20, and the actual value is 16, then the percentage is 60%.

By default, the Trend control runs in real-time (or "play") mode with a duration of 60 seconds. That means the graph is continuously updated with the current values of the tags, and only the last 60 seconds are actually shown on the graph. However, you can change all of this in the Settings window. For more information, see below.

To use the Trend control:

1. To hide or show the legend below the trend graph, click/tap **Toggle Legend**.
2. To hide or show a specific process value on the trend graph, click/tap the check box in the first column.
3. To add translucent fills below the trend lines:
   a) Click/tap the **Settings** button.
      The **Settings** screen is displayed.
   b) Toggle the **Fill** switch to **ON**.
   c) Click/tap **Trend** to apply your changes and return to the Trend control.
4. To change the duration (i.e., the X-axis) of the trend graph:
   a) Click/tap the **Settings** button.
      The **Settings** screen is displayed.
b) In the **Duration** text box, type the new duration in seconds.

c) Click/tap **Trend** to apply your changes and return to the Trend control.

5. To show the trend lines as absolute values rather than as percentages:

a) Click/tap the **Settings** button.
   The *Settings* screen is displayed.

b) Toggle the **Percentage Mode** switch to **OFF**.

c) In the **Min** and **Max** boxes, type the minimum and maximum values for the Y-axis of the trend graph. These apply to all of the process values, overriding the values' individual **Min** and **Max** settings that are used to calculate the percentages.

d) Click/tap **Trend** to apply your changes and return to the Trend control.

6. To switch the Trend control to historical mode:

a) Click/tap the **Settings** button.
   The *Settings* screen is displayed.

b) Toggle the **Play** switch to **OFF**.
   The **Duration** setting changes to **Period**.

c) Click/tap **Period**.
   The *Period* screen is displayed.

d) Use the date and time controls to set the **From** (start) and **To** (end) points of the graph’s X-axis.

e) Click/tap **Settings** to apply your changes and return to the **Settings** screen.

f) Click/tap **Trend** to apply your changes and return to the Trend control.

g) Click/tap **Toggle Cursor** to turn on a vertical cursor that you can slide left and right on the trend graph, in order to see the process values at a specific time.

   **Note:** The **Toggle Cursor** button is hidden when the Trend control is not in historical mode.

7. To return to the home screen, click/tap the **Return** button.
USE THE SCREENS CONTROL

Use the Screens control to view selected project screens.

The Screens control shows a list of project screens that you have selected to include in Mobile Access. In most cases, selected screens function the same as they do on web thin clients.

All screen objects and animations are continuously updated to show the current values of their associated project tags. Also, the Disable and Security settings are enforced on all screen objects and animations, but the E-sign setting is not.

To use the Screens control:

1. To go to a particular project screen, click/tap the screen name in the list.
2. To return to the list of project screens, click/tap the Screens button in the top-left corner.
3. To return to the home screen, click/tap the Return button ⏯️.
LINK DIRECTLY TO A PROJECT SCREEN IN THE MOBILE ACCESS WEB INTERFACE

Use a custom URL to bypass the Mobile Access web interface and link directly to a specific project screen.

In most cases, you would use the Mobile Access web interface to log onto the project runtime server and then navigate through your project. If you often visit a specific project screen, however, you might want to bookmark that screen and skip everything else. To do that, you can compose a custom URL that includes the server address, the name of the project screen, and if you wish, your user credentials.

Note: This procedure works only for the project screens that you have selected to include in the Screens controls in the Mobile Access web interface. It does not work for any other screens that have not been selected, nor does it work for the Alarm, Process Values, or Trend controls in the web interface. For more information, see Configure the Mobile Access area settings on page 580.

When you use this custom URL, your primary concern should be project security. If you have enabled the security system for your project, you will always need to log onto the project runtime server before you can access any screens. Therefore, you must decide whether to include your user credentials in the URL. If you do, you will be automatically logged onto the server and taken directly to the project screen. If you do not, you will be prompted to log on and then redirected to the project screen.

The key here is that if you save the custom URL with your user credentials included, anyone who has an opportunity to examine the URL — for example, by copying your bookmark or watching over your shoulder when you use it — might learn your credentials. You must consider the possibilities and weigh your convenience against project security.

There are four ways to compose the custom URL:

Screen only
This URL will take you to the Mobile Access Logon page, where you will be prompted for your user name and password. After you log on, you will be redirected to the specified project screen.

https://hostname_or_address/ma/index.htm?screen=screenname

Example:

Screen, with guest logon
This URL will log you on as a guest user and take you directly to the specified project screen.

https://hostname_or_address/ma/index.htm?screen=screenname&guestuser=1

Example:
https://scada.ourcompany.com/ma/index.htm?screen=Animations&guestuser=1

Screen, with user name
This URL will take you to the Mobile Access Logon page and automatically enter the specified user name. You will be prompted for the corresponding password. After you log on, you will be redirected to the specified project screen.

https://hostname_or_address/ma/index.htm?screen=screenname&user=username

Example:

Screen, with user name and password
This URL will automatically log you on as the specified user and then take you directly to the specified project screen.

https://hostname_or_address/ma/index.htm?screen=screenname&user=username&password=password

Example:


This is the least secure option, because the user name and password are both included in the URL.

**Note:** The custom URL cannot contain any spaces, which means that the screen name, user name, and password cannot include any spaces. (Character encoding — that is, substituting the character code `%20` for any spaces in the URL — is not supported at this time.)

If a screen name includes a space, save the project screen with a new file name and then re-publish it. For more information, see Configure the Mobile Access area settings on page 580.

If a user name or password includes a space, change it in the security system settings. For more information, see Creating and configuring users on page 486.

**Troubleshooting project screens in Mobile Access**

Use the activity log to troubleshoot project screens in the Mobile Access web interface.

Mobile Access currently supports many but not all features of InduSoft Web Studio. If you use an unsupported feature in a project screen, you might see unexpected behavior when you view that screen in the Mobile Access web interface. Such behavior can range from incorrect tag changes and function calls to objects, animations, or scripts that do not work at all.

Mobile Access automatically generates an activity log for the web interface, and you can use the browser console to get the log for the project screen that you are currently viewing. Then, using the information provided by the log, you can identify and resolve most issues that you might have.

There are also some common issues that can be resolved without using the activity log. Those issues are addressed at the end of this section.

**OPEN THE BROWSER CONSOLE TO VIEW THE ACTIVITY LOG**

Open the browser console in Internet Explorer, Google Chrome, or Safari in order to view the activity log for the Mobile Access web interface.

The console is a part of the web browser's developer tools, and it is used to troubleshoot code in a web page. When you are using the Mobile Access web interface, the console shows log messages for the project screen that you are currently viewing. For more information, see Use the activity log to troubleshoot a project screen on page 597.

Please note that you can open the browser console at any time, but it will not display any log messages unless both the project runtime server and the web server are running and you have successfully logged on to the web interface. For more information, see Log on to the Mobile Access web interface on page 586.

**Tip:** These instructions apply only to the desktop versions of these browsers. If you want to access the browser console or debugging tools in a mobile browser, please consult the documentation for that browser.

**Internet Explorer**

To open the browser console in Internet Explorer (Windows only), do one of the following:

- Click the Action tool (i.e., the gear icon) on the toolbar, and then on the menu, click F12 Developer Tools; or
- Press F12 on the keyboard.
The console is displayed at the bottom of the browser window.

**Google Chrome**

To open the browser console in Google Chrome (Windows, Linux, and Mac OS X), do one of the following:

- Click the Chrome tool (i.e., the menu icon) on the toolbar, and then on the menu, click **Tools > JavaScript console**; or
- Press **Ctrl+Shift+J** (**Cmd+Option+J** on Mac OS X) on the keyboard.

The console is displayed at the bottom of the browser window.

**Safari**

To open the browser console in Safari (Windows and Mac OS X):

1. Do one of the following:
   - In Safari for Windows, click the Action tool (i.e., the gear icon) on the toolbar, and then on the menu, click **Preferences**; or
   - In Safari for Mac OS X, click the Safari menu (i.e., the application menu) on the menu bar, click **Preferences**.

   The Preferences window is displayed.

2. Click the **Advanced** tab.

3. Select the **Show Develop menu in menu bar** check box.

4. Close the Preferences window.

5. Do one of the following:
   - In Safari for Windows, click the Page tool on the toolbar, and then on the menu, click **Develop > Show Error Console**; or
   - In Safari for Mac OS X, click the Develop menu on the menu bar, and then on the menu, click **Show Error Console**.
The console is displayed at the bottom of the browser window.

![Browser console in Safari](image)

**USE THE ACTIVITY LOG TO TROUBLESHOOT A PROJECT SCREEN**

Use the activity log, which can be viewed in the browser console, to troubleshoot a project screen in the Mobile Access web interface.

This log is similar to but separate from the project runtime log that is displayed in the **Output window**; it comprises only messages that describe the behavior of project screens in the web interface.

Please keep in mind that the console only displays the log for the project screen that you are currently viewing. It does not display the logs for any other client sessions, neither on other computers nor in another client application (e.g., Secure Viewer) on the same computer. It also does not display any information about the project runtime server.

![An example of the activity log in the Internet Explorer 11 console](image)

There are several different types of messages that you can see in the activity log:

**Error**

These are messages about serious errors encountered during run time, like unsupported features used in a screen and run-time errors in the Mobile Access engine. Mobile Access errors are marked by the yellow “Warning” icon in most browser consoles.

**Warning**

These are warnings about things in your project screen that can affect run-time performance, such as excessive tag synchronization and function calls that must be executed on the server. Mobile Access warnings are marked by the blue “Message” or “Info” icon in most browser consoles.

**Info**
These are informational messages that report individual function calls, tag value changes, screen openings and closings, and so on, as well as any trace messages that you choose to include in your project. Mobile Access info messages are not marked by icons.

**Note:** It is not a mistake that we use the "Warning" icon to mark Mobile Access errors and the "Message" icon to mark Mobile Access warnings. The console's red "Error" icon is reserved for critical errors in the browser itself.

**Use the activity log to troubleshoot an issue**

Here is an example of how to use the activity log to troubleshoot an issue in one of your project screens. The screen MyScreen contains a Text Box object that has been configured with the tag MyInput:

When you normally view the screen in the Mobile Access web interface, the text box shows a value of 0 as it waits for user input:
If the tag does not exist, however — for example, if you accidentally deleted it from the tags database — then the text box shows its mask instead and rejects all input:

Verifying the project will catch things like missing tags, but in order to verify the project, you must be able to use the InduSoft Web Studio development environment to open and edit the project. The user typically will not be able to do that during run time.

This is where you can use the activity log to troubleshoot the issue. Open the browser console, and then look for a message like this in the activity log:

Invalid Date Invalid Date, L1, Error, [04BF97E0] myscreen, myscreen (Screen Script) - Expression(1000000): MyInput, Line 1 => Error: Could not read objects database. Object MyInput not found [Error Code: 0x80070057]

Tip: The browser console cannot receive log messages unless it is open, so if the console is closed when you first notice an issue, open it and then access the screen again.

The message provides the name of the screen, the name of the affected object or tag, and the exact nature of the error. With this information, you should get some idea of how to resolve the issue in your project. This is a very simple example, of course, and an issue like this typically will be included in a Level 1 log; more complex issues might require increasing the log level and searching through more messages. But this essentially is the procedure for using the activity log to troubleshoot a project screen in Mobile Access.

Increase the log level to get more information

By default, the activity log is set to level 1, which provides the least amount of information. If you have issues while using the Mobile Access web interface — especially while trying to view selected project screens — and the log does not help, you can increase the log level to get more information.

Be careful not to set the log level too high, however, because the higher it is, the more detailed and verbose the log will be, which might make it hard for you to find the information that will actually help you to resolve your issues. Try increasing the log level in steps, first from level 1 to level 2, then from level 2 to level 3, and so on until you get the information you need.

Here is a basic guide to which levels you should try:
• If objects, animations, or scripts are not working at all, try level 1. This level should be enough to identify missing tags, unsupported features, and other such things. For more information, see List of features supported in project screens in Mobile Access on page 563.

• If a project screen is unusually slow to open or update, try level 3 (which includes levels 1–2). This level should help you to diagnose issues that affect run-time performance. For more information, see Tips for Mobile Access development and run time on page 569.

• If you simply do not see the behavior that you expect during run time, try level 5 (which includes levels 1–4). This level reports every tag update and function call.

Please note that these are only general suggestions; you will probably need to change the log level more than once to get the right amount of detail on your specific issues. For more information about changing the log level, see Configure the Mobile Access global settings on page 584.

Filter the activity log in the browser console

You can filter the activity log in the browser console in order to reduce the number of log messages that you must look through to find the information that you need. The tools for filtering the log vary by browser, so please consult the documentation for the browser that you are using.

For example, to filter by message type in Internet Explorer, right-click in the browser console and then select or clear the types on the shortcut menu:

![Filtering by message type in Internet Explorer](image)

In Google Chrome, click the Filter tool on the console tool bar to reveal additional tools:

![Additional filter tools in Google Chrome](image)

If you filter by message type, remember that Mobile Access errors are displayed as Warnings, Mobile Access warnings are displayed as Messages (in Internet Explorer) or Info (in Google Chrome), and Mobile Access info messages are displayed as Logs. Otherwise, you can try to filter by specific text.
Mobile Access Tabular

Mobile Access Tabular enables your project to send alarms and process values to cell phones, PDAs, and other mobile devices.

How It Works

When you enable the Mobile Access Tabular feature in your IWS project and then run your project, IWS creates a small, ASP-powered website and periodically refreshes it with alarm notifications and whatever tag values you choose to make available. You can also acknowledge alarms and write new tag values through the website. As long as the web server — typically Microsoft IIS, because it must support ASP — and network are properly configured to allow access, all you need to do is point your browser to the website and log on.

Licensing

One Mobile Access Tabular Client is included with every IWS runtime license. That means the Mobile Access Tabular website will allow only one user to connect at a time. If you want the website to accept more users, then you must upgrade your license to include additional SMA Clients. For more information, see License Settings.

Enabling and Configuring Mobile Access Tabular

To enable Mobile Access Tabular and configure the data to be served:


   The Mobile Access Settings dialog is displayed.

2. Select Enable Mobile Access.

3. In the LogOn Access Level text-box, type the user security level needed to log on to the website. For more information about security levels, see Security.

4. The website will show all active alarms to all logged-on users; there is currently no way to show or hide specific alarms. You can set the user security level needed to acknowledge alarms, however, and it may be different from the level needed to log on. In the Ack Access Level text-box, type the required level.

5. To have the website show tag values, select Enable in the Process group-box.

6. For each tag you want to show:
   a. In the Tag Name column, type the name of tag or double-click to open the Object Finder and select the tag.
b. In the **Description** column, type a description of the tag. This description is displayed only in the website and it may be different from the tag’s existing description in the **Project Tags** datasheet.

c. In the **Write** column, select the checkbox to make the tag writeable from the website.

7. In the **Write Access Level** text-box, type the user security level needed to write new tag values. This applies to all tags that are made writeable.

8. You may choose to decrease the data refresh rate to improve application performance, especially in non-critical applications where alarms are uncommon and/or tag values do not change frequently. The refresh rates for Alarms and for Process information can be adjusted separately — in the corresponding **Refresh Rate** text-box, type the new rate in seconds.

9. Click **OK** where you are done.

The following screenshot show Mobile Access enabled with a selection of tags:

![Mobile Access Settings](example.png)

**Example of Mobile Access settings**

### Accessing the website

Once you’ve enabled Mobile Access Tabular, configured IIS, and run your project, you can access your project by entering the URL in your mobile browser:

- If the IIS home directory is set to the `\Web` sub-folder in your project folder, then the URL is

  ![serveraddress/SMA/LogOn.asp](http://serveraddress/SMA/LogOn.asp)

- If the IIS home directory is set to the `\Web\SMA` sub-folder in your project folder, then the URL is

  ![serveraddress/LogOn.asp](http://serveraddress/LogOn.asp)

The first page is a standard security login, similar to the **LogOn** dialog in your project. Log on with your IWS username and password (not your Windows user account), and then the Main Menu is displayed.

### Main Menu
Main menu

Alarms

Process

Log Off

The main menu has three options:

• Click **Alarms** to see and acknowledge alarms.
• Click **Process** to see and write tags.
• Click **Log Off** to log off from the website.

This menu is also displayed in the Alarms and Process pages described below.

**Alarms**

<table>
<thead>
<tr>
<th>State</th>
<th>Message</th>
<th>Type</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hi</td>
<td>Hi Pressure</td>
<td>Hi</td>
<td>14:59</td>
</tr>
<tr>
<td>Lo</td>
<td>Low Temperature</td>
<td>Lo</td>
<td>14:44</td>
</tr>
<tr>
<td>Hi</td>
<td>Hi Level</td>
<td>Hi</td>
<td>14:59</td>
</tr>
</tbody>
</table>

The **Alarms** table shows the currently active alarms in your project. To acknowledge an alarm from your mobile browser, simply click on the alarm name.

**Process**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure Set Point</td>
<td>27</td>
</tr>
<tr>
<td>Temperature Set Point</td>
<td>20</td>
</tr>
<tr>
<td>Level Set Point</td>
<td>60</td>
</tr>
<tr>
<td>Main Motor</td>
<td>On</td>
</tr>
</tbody>
</table>

You can use the **Process** table to configure set points, turn pumps on and off, send messages to users — anything that involves writing to tags. To write to a tag, simply click on the tag value.

**Tip:** By default, a user session will automatically expire after 10 minutes (600 seconds) of inactivity. If you want a user to be able to stay logged on, then open the file `\Web\SMA\config.inc` in your project folder and change the parameter `logonExpiration` to the desired period in seconds.

For example, if you want a user to stay logged on for up to four hours, then change the parameter to:

`logonExpiration = 14400`
Please note that as long as a user is logged on, he counts against the number of SMA Clients in the runtime license. If too many users stay logged on for extended periods, then you may run out of available connections.
Configuring a database interface with IWS is basically linking tasks from IWS (Alarms, Events or Trends) to tables of external databases via a specific Database Provider that supports the database you have chosen.

Each history task (Alarm, Events or Trend) can be configured to save data either to files with the proprietary format from IWS or to external SQL Relational Databases. Use the Options tab to configure the database to save Alarm and Event history. (See the Trend Folder for instructions for saving history for the trend tasks.)

IWS supports ADO.NET to provide an intuitive, simple, flexible and powerful interface with standard technologies from MDAC (Microsoft Data Access Components) such as OLE-DB (Object Linking Embedded — Database) and ODBC (Open Database Connectivity). By using this capability, you can connect to any database that is MDAC compatible (please see the Conformance Table for the list of databases already tested).

The following tasks support the database interface:

- **Alarms**: The project can save and/or retrieve the alarm history messages in a relational database.
- **Events**: The project can save and/or retrieve the event messages in a relational database.
- **Trends**: The project can save and/or retrieve the Trend history values in a relational database.
- **Viewer**: Database information can be displayed both in table format (Alarm/Event Control and Grid objects) or in a graphical format (Trend Control object).
- **Web**: Because the items listed below are already available in IWS Web interface, you can deploy a project that stores/saves data in a relational database and have it working over the Web.

Using its embedded database interface, IWS can easily provide data from the plant floor to third-party systems (e.g., ERP) or get data from them.

IWS can interface with any relational database supported by a valid ADO.NET Provider, OLE DB provider or ODBC driver. However, the conformance tests were executed with the following databases:

### Conformance Test Table

<table>
<thead>
<tr>
<th>Database</th>
<th>Database Version</th>
<th>ADO.NET Provider</th>
<th>Assembly Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server 2000</td>
<td>8.0</td>
<td>System.Data.SqlClient</td>
<td>1.0.5000.0</td>
</tr>
<tr>
<td>Microsoft Access 2000</td>
<td>9.0.3821 SR-1</td>
<td>System.Data.OleDbClient</td>
<td>1.0.5000.0</td>
</tr>
<tr>
<td>Microsoft Excel 2000</td>
<td>9.0.3821 SR-1</td>
<td>System.Data.OleDbClient</td>
<td>1.0.5000.0</td>
</tr>
<tr>
<td>Oracle</td>
<td>10g Release 1 for Windows</td>
<td>System.Data.OracleClient</td>
<td>1.0.5000.0</td>
</tr>
<tr>
<td>Sybase</td>
<td>Anywhere 9.0.1.1751</td>
<td>iAnywhere.Data.AsaClient</td>
<td>9.0.1.1751</td>
</tr>
<tr>
<td>MySQL</td>
<td>4.0.20a</td>
<td>ByteFX.MySqlClient</td>
<td>0.7.6.15073</td>
</tr>
</tbody>
</table>

**Note:** For information about how to configure a specific database, please refer to the following:

- Database Appendix A: Using ODBC Databases
- Database Appendix B: Using Microsoft SQL Server
- Database Appendix C: Using ORACLE Databases
- Database Appendix D: Using Microsoft Access Databases
- Database Appendix E: Using SQL Server CE
- Database Appendix F: Using Sybase
- Database Appendix G: Using Microsoft Excel
- Database Appendix H: Using MySQL
SQL Relational Databases

A SQL Relational Database is a set of information stored in tables with fields and registers, which support SQL commands.

Each database can have one or more tables. Each table is composed of fields (columns) and registers (rows). Typically, the fields are pre-defined and the project adds or reads one or more registers, according to the query condition.

IWS uses Database Providers (ADO.NET) to interface with SQL Relational Databases. Database Providers are libraries developed to access data from different databases through SQL commands. The ADO.NET Provider for a specific database can be supplied by the operating system or by the database manufacturer.

The following picture illustrates how IWS can interface with different databases using a different Database Provider for each database.

The previous picture shows some of the most popular ADO.NET Providers for databases. Notice that the Microsoft ADO.NET Provider for ODBC Drivers allows you to access the database through an ODBC driver. See Database Appendix A: Using ODBC Databases for information about how to use this provider. It is also possible that you do not have an ADO.NET provider, but an OLE DB provider is available. By using the Microsoft ADO.NET Provider for OLE DB you can get access to the database; the Microsoft Jet OLE DB provider gives access to applications in the Microsoft Office package by using this approach.

Note: It is important to note that IWS provides the interface for ADO.NET Providers. However, the ADO.NET Providers and/or the ODBC Driver/OLE DB Provider must be supplied either by the
operating system or by the database manufacturer. If your Connection String does not refer to a valid ADO.NET Provider, the OLE.DB Provider will be used.

Although most projects typically link to only one type of database, IWS gives you the flexibility to link each task to a specific database supported by a Database Provider. Furthermore, by using this architecture, you do not need to worry about the specific characteristics of each database (it is mostly handled by the Database Provider for each database or by the IWS Database Gateway interface). Therefore, the project settings are mostly uniform, regardless of the specific database chosen by you.
Linking the Database Through a Remote DB Provider

Depending on the architecture of your project, the ADO.NET Provider for the SQL Relational Database may not be available in the same stations where IWS is running. This scenario is especially common when the project is run on a Windows Embedded target system (currently, most of the Providers are not supported for Windows Embedded). In order to solve this problem, we have designed a flexible solution that allows you to configure distributed systems, as illustrated in the picture below:

![Diagram showing the setup of the Studio Application, Studio Database Gateway, and Database.](image)

The project is running on the Server station (where InduSoft Web StudioEmbeddedView, or CEView is installed). The project can communicate with the IWS Database Gateway (running in a remote computer) via TCP/IP. The Gateway implements the interface with the Database through the Database Provider available in the computer where it is running.

The Studio Database Gateway does not require complex configuration. Just copy the files `STADOSvr.exe` and `StADOSrv.ini` from the `\BIN` sub-folder of IWS and paste them under any directory of the computer that will be used as the Gateway station and execute the `STADOSvr.exe` program. There are advanced settings associated with the IWS Database Gateway, but they should be changed only under special circumstances. See Studio Database Gateway for information on how to configure the IWS Database Gateway advanced settings.

**Note:** The Studio Database Gateway is a TCP/IP Server for the IWS project and it uses the TCP Port 3997 by default. You can specify a different port number when executing the `STADOSvr.exe` program according to the following syntax:

```
STADOSvr.exe Port Number
```

Example: `STADOSvr 3998`
**Studio Database Gateway**

The Studio Database Gateway is a TCP/IP server that interacts with databases using the Microsoft .Net Framework. It can run on the same computer that is running the IWS project, or on a different computer. The Database Gateway Host in the Advanced Settings (see Database Configuration dialog) specifies whether the gateway will be running on the local computer or not. If you are using the local computer you should enter either `localhost` or `127.0.0.1` in the Host name. You do not need to worry about starting or stopping the gateway because it will be done automatically by IWS tasks. On the other hand, when running the gateway remotely, you need to start the gateway manually. To do so, copy the files `StADOSvr.exe` and `StADOSvr.ini` from the `BIN` folder to the remote computer, and then execute the `StADOSvr.exe`.

The gateway can be started multiple times for different TCP/IP port numbers. The default port number is 3997, and it is changed by specifying the desired port number in the command prompt (e.g., `StADOSvr 1111`). When running the StADOSvr, it will add the following icon to the system tray:

![Studio Database Gateway](image)

When you right-click on the icon, the following shortcut menu is displayed:

```
About
- Hide
- Exit
```

The **Hide** option controls whether the Studio Database Gateway window is displayed on the desktop. (The gateway software runs continuously after you launch it, regardless of whether the window is displayed.) If you disable the **Hide** option, then the window is displayed:

![Studio Database Gateway](image)

Any failure that occurs during operations with databases will be displayed both in this window and also in the LogWin window. The messages are reported by exceptions generated by the ADO.NET Provider. (Please refer to Database Troubleshooting for more information about error messages in the gateway module.)

You can configure the output in this window by using the **Log** menu:

- **Show Log** menu option: Shows the IWS Database Gateway log files.
• **Options** menu option: Open the *Configure Messages* dialog.

![Configure Messages dialog](image1)

**Studio Database Gateway: Configure Messages dialog**

• *Show Messages* pane: Select **Errors Only** to show only error messages in the log, or select **All Messages** to show all database messages.

• *Additional Information* pane: Configure to show additional information about each database message.
  - **Message Type** checkbox: Click (check) this option to show the type of the message.
  - **Date/Time** checkbox: Click (check) this option to show the timestamp of the message.

Also, you can directly monitor database connections using the **Monitor** menu:

• **Connections** menu option: Displays the *Database Connections* window.

![Database Connections window](image2)

**Studio Database Gateway: Database Connections window**

• **Log to File** menu option: Logs the monitor output to a file named *logdate.txt* in the same folder as *StADOSvr.exe*.

**Advanced Settings**

The Studio Database Gateway has Advanced Settings that are configured in the *StADOSvr.ini* file. If you are having problems interfacing with a specific database, you will probably need to change some of these settings or add new providers to the file. The following parameters are available:

<table>
<thead>
<tr>
<th>Section of .INI File</th>
<th>Parameter</th>
<th>Accepted Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Providers</td>
<td>SaveMSec</td>
<td>1: Disable</td>
<td>This setting specifies the default behavior for the provider when saving milliseconds. The default can be changed on the Advanced Settings in the Database Configuration Dialogs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Enable 3: Separate Column</td>
<td></td>
</tr>
<tr>
<td>Assembly</td>
<td></td>
<td>Any string that contains a .NET Framework assembly</td>
<td>Assembly option for all providers. The assembly has all the classes required to interface with the database. Most of the providers are inside the System.Data assembly.</td>
</tr>
<tr>
<td>ConnectionClass</td>
<td></td>
<td>Any connection class inside the assembly</td>
<td>The Connection Class is the one that implements the System.Data.IDbConnection interface.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Accepted Values</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>DateAdapterClass</td>
<td>Any data adapter class inside the assembly</td>
<td>The Data Adapter class is used on operations where updates to the database are necessary. It must be compatible with the connection class specified and it should implement IDbDataAdapter.</td>
<td></td>
</tr>
<tr>
<td>CommandBuilderClass</td>
<td>Any command builder class inside the assembly</td>
<td>The Command Builder class is also responsible for updates on databases. It must be compatible with the connection class.</td>
<td></td>
</tr>
<tr>
<td>Provider</td>
<td>Name of the provider</td>
<td>One of the parameters in the connection string is the &quot;Provider&quot;. The Studio ADO Gateway compares the value on the connection string with the value for this parameter in each provider and defines the proper one to be used.</td>
<td></td>
</tr>
<tr>
<td>ColumnDelimiterPrefix</td>
<td>Any character or group of characters</td>
<td>Specify a character that will be placed before column names on SQL statements.</td>
<td></td>
</tr>
<tr>
<td>ColumnDelimiterSuffix</td>
<td>Any character or group of characters</td>
<td>Specify a character that will be placed after column names on SQL statements.</td>
<td></td>
</tr>
<tr>
<td>TableDelimiterPrefix</td>
<td>Any character or group of characters</td>
<td>Specify a character that will be placed before table names on SQL statements.</td>
<td></td>
</tr>
<tr>
<td>TableDelimiterSuffix</td>
<td>Any character or group of characters</td>
<td>Specify a character that will be placed after table names on SQL statements.</td>
<td></td>
</tr>
<tr>
<td>ValueString</td>
<td>Any string</td>
<td>This value indicates how constant values are identified on SQL statements. For Microsoft SQL databases for instance, the value should be @Value, for ODBC question mark (?).</td>
<td></td>
</tr>
<tr>
<td>ValueStringPrefix</td>
<td>Any string</td>
<td>This value indicates a prefix to be used before the values. Oracle values, for instance, require the prefix. The SQL statements use value identifiers by using their prefixes, but the parameters in the Connection class do not use the prefix.</td>
<td></td>
</tr>
<tr>
<td>ValueAddNumber</td>
<td>0 or 1</td>
<td>Indicates whether a sequential number should be added to the ValueString to identify the parameter or not. For Microsoft SQL database, this parameter should have the value 1, because parameters are identified by using @Value1, @Value2, …, @ValueN. For ODBC, this parameter should be 0.</td>
<td></td>
</tr>
<tr>
<td>BoolType</td>
<td>Any string representing a valid data type for the database</td>
<td>When trying to create columns to store boolean values, the data type specified on this parameter will be used. You need to make sure that the data type specified is able to save boolean values.</td>
<td></td>
</tr>
<tr>
<td>IntegerType</td>
<td>Any string representing a valid data type for the database</td>
<td>When trying to create columns to store integer values, the data type specified on this parameter will be used. You need to make sure that the data type specified here is able to store 32 bit values.</td>
<td></td>
</tr>
<tr>
<td>RealType</td>
<td>Any string representing a valid data type for the database</td>
<td>When trying to create columns to store real values, the data type specified on this parameter will be used. You need to make sure that the data type specified here is able to store 64 real values.</td>
<td></td>
</tr>
<tr>
<td>StringType</td>
<td>Any string representing a valid data type for the database</td>
<td>When trying to create columns to store string values, the data type specified on this parameter will be used. You need to make sure that the data type specified is able to save the number of characters that you are willing to save on your project.</td>
<td></td>
</tr>
<tr>
<td>TimeStampType</td>
<td>Any string representing a valid data type for the database</td>
<td>When trying to create columns to store TimeStamp values, the data type specified on this parameter will be used.</td>
<td></td>
</tr>
</tbody>
</table>
The parameters are grouped into four sections — Providers, Communication, Connection, and Options — but all of the parameters for configuring database providers are listed in the Providers section of the file. The default values are specified in the beginning of the file, using the prefix "Default" in each parameter as shown below:

```ini
[Providers]
DefaultSaveMSec=3
DefaultAssembly=System.Data
DefaultConnectionClass=System.Data.OleDb.OleDbConnection
DefaultDataAdapterClass=System.Data.OleDb.OleDbDataAdapter
DefaultCommandBuilderClass=System.Data.OleDb.OleDbCommandBuilder
DefaultValueString=@Value
DefaultValueAddNumber=1 DefaultBoolType=INTEGER
DefaultIntegerType=INTEGER DefaultRealType=REAL
```

<table>
<thead>
<tr>
<th>Section of .INI File</th>
<th>Parameter</th>
<th>Accepted Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EnableTop</strong></td>
<td></td>
<td>0 or 1</td>
<td>When this field is set to 1, the ADO will place the TOP in the SQL statement to limit the amount of registers required.</td>
</tr>
<tr>
<td><strong>SingleConnection</strong></td>
<td></td>
<td>0 or 1</td>
<td>When this field is set to 1, the ADO will open only one connection with the database, regardless of how many tasks or computers are requesting services from it. The synchronization between the tasks will be performed by the gateway, and they will not be able to be executed simultaneously if this option is enabled.</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td><strong>TimeOut</strong></td>
<td>Any integer</td>
<td>Timeout (in seconds) to perform insert and update operations. If no value is specified, then the default of 2 seconds is used.</td>
</tr>
<tr>
<td></td>
<td><strong>LongTimeOut</strong></td>
<td>Any integer</td>
<td>Timeout (in seconds) to perform connection and query updates. If no value is specified, then the default of 5 seconds is used.</td>
</tr>
<tr>
<td></td>
<td><strong>SyncTimeOut</strong></td>
<td>Any integer</td>
<td>Timeout (in seconds) to perform synchronization. If no value is specified, then the default of 60 seconds is used.</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td><strong>RegBufSize</strong></td>
<td>Any integer</td>
<td>Size (in bytes) of the internal buffer used by the database API. If no value is specified, then the default of 128 bytes is used.</td>
</tr>
<tr>
<td></td>
<td><strong>InsertBufferSize</strong></td>
<td>Any integer</td>
<td>Size (in bytes) of the buffer for all data to be inserted in the database. This is to prevent alarms/events from individually timing out, stacking up, and causing the project to freeze. If no value is specified, then the default of 1024 bytes is used.</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td><strong>CultureInfo</strong></td>
<td>Any standard language-country code (e.g., en-US), which is a combination of an ISO 639-1 language code and an ISO 3166-1 country code</td>
<td>The language or culture that should be used to format values in SQL statements. For example, this determines whether the decimal mark in numeric values is a point (###.##) or a comma (###,##). This is important when the option Disable SQL variables is selected in the Database Configuration settings. For more information, see Database Configuration on page 98. The default value of this parameter is en-US (i.e., &quot;English – United States&quot;).</td>
</tr>
<tr>
<td></td>
<td><strong>DisableCloseQuestion</strong></td>
<td>0 or 1</td>
<td>When this field is set to 1, no user confirmation is required to close StADOSvr.exe. This is important for devices that have alternative methods for exiting applications and restarting.</td>
</tr>
<tr>
<td></td>
<td><strong>Monitor</strong></td>
<td>0 or 1</td>
<td>When this field is set to 1, the Monitor menu is displayed in the Studio Database Gateway. This is enabled by default now, but it was not in previous versions of InduSoft Web Studio.</td>
</tr>
</tbody>
</table>
DefaultStringType=VARCHAR(255) DefaultTimeStampType=DATETIME
DefaultSingleConnection=0

The next item on the file lists the amount of providers:

Count=5

The providers are identified by the "Provider" parameter followed by a number. When connecting to a
database, the Provider parameter in the connection string is compared to the provider's identification, in order
to determine which provider will be used. If there is no provider with the value on the connection string, all
the default values are assumed. Besides its identification, each provider can have its own value per each
parameter. Again, if no value is specified, the default is used. Below is an example with seven providers:

Count=7

Provider1=MICROSOFT.JET.OLEDB
SaveMSec1=3
ColumnDelimiterPrefix1=[
ColumnDelimiterSuffix1=]
SingleConnection1=1

Provider2=SQLOLEDB
ConnectionClass2=System.Data.SqlClient.SqlConnection
DataAdapterClass2=System.Data.SqlClient.SqlDataAdapter
CommandBuilderClass2=System.Data.SqlClient.SqlCommandBuilder
ColumnDelimiterPrefix2=[
ColumnDelimiterSuffix2=]
TableDelimiterPrefix2=[
TableDelimiterSuffix2=]
RealType2=FLOAT

Provider3=MSDASQL
ConnectionClass3=System.Data.Odbc.OdbcConnection
DataAdapterClass3=System.Data.Odbc.OdbcDataAdapter
ValueString3=?
ValueAddNumber3=0
StringType3=VARCHAR(128)
EnableTop3=0

Provider4=ORAOLEDB
Assembly4=System.Data.OracleClient
ConnectionClass4=System.Data.OracleClient.OracleConnection
DataAdapterClass4=System.Data.OracleClient.OracleDataAdapter
ValueString4=Value
ValueAddNumber4=1
ValueStringPrefix4=: BoolType4=Number(1)
IntegerType4=Number(10)
RealType4=Number
StringType4=VARCHAR(255)
TimeStampType4=TIMESTAMP(0)
EnableTop4=0

Provider5=ASAPROV
Assembly5=iAnywhere.Data.AsaClient
ConnectionClass5=iAnywhere.Data.AsaClient.AsaConnection
DataAdapterClass5=iAnywhere.Data.AsaClient.AsaDataAdapter
ValueString5=?
ValueAddNumber5=0
ColumnDelimiterPrefix5=[
ColumnDelimiterSuffix5=]
TableDelimiterPrefix5=[
TableDelimiterSuffix5=

Provider6=MYSQLPROV
Assembly6=ByteFX.MySqlClient
ConnectionClass6=ByteFX.Data.MySqlClient.MySqlConnection
DataAdapterClass6=ByteFX.Data.MySqlClient.MySqlDataAdapter
CommandBuilderClass6=ByteFX.Data.MySqlClient.MySqlCommandBuilder
ValueString6=@Value
ValueAddNumber6=1
StringType6=VARCHAR(128)
EnableTop6=0

Provider7=MSDAORA
Assembly7=System.Data.OracleClient
ConnectionClass7=System.Data.OracleClient.OracleConnection
DataAdapterClass7=System.Data.OracleClient.OracleDataAdapter
ValueString7=Value
ValueAddNumber7=1
BoolType7=Number(1)
IntegerType7=Number(10)
RealType7=Number
StringType7=VARCHAR(255)
TimeStampType7=TIMESTAMP(0)
EnableTop7=0
Database Configuration

The Database Configuration dialog allows you to configure the necessary settings to link IWS to an external database file.

![Database Configuration dialog](image)

- **Database** combo-box: Allows you to select either Primary or Secondary. With Primary, all settings displayed in the Database Configuration window apply to the Primary Database interface. Otherwise, they apply to the Secondary Database interface. You can configure the Secondary database in the following modes:
  - **Disabled**: In this mode, IWS saves data in the Primary Database only. If the Primary Database is unavailable for any reason, the data is not saved anywhere else. This option may cause loss of data if the Primary Database is not available.
  - **Redundant**: In this mode, IWS saves data in both Primary and Secondary Databases. If one of these databases is unavailable, IWS keeps saving data only in the database that is available. When the database that was unavailable becomes available again, IWS synchronizes both databases automatically.
  - **Store and Forward**: In this mode, IWS saves data in the Primary Database only. If the Primary Database becomes unavailable, IWS saves the data in the Secondary Database. When the Primary Database becomes available again, IWS moves the data from the Secondary Database into the Primary Database.

**Note**: The Primary and Secondary can be different types of databases. However, they must have the same fields.

Using the Secondary Database, you can increase the reliability of the system and use the Secondary Database as a backup when the Primary Database is not available. This architecture is particularly useful when the Primary Database is located in the remote station. In this case, you can configure a Secondary Database in the local station to save data temporarily if the Primary Database is not available (during a network failure, for instance).

- **Use project default** checkbox: When this option is checked, IWS uses the settings configured in the Default Database for the task that is being configured (Connection string, User name, Password, Retry Interval and Advanced Settings). When this option is not checked, you can configure these settings individually to the current task.

- **Connection string** field: This field defines the database where IWS will write and read values as well as the main parameters used when connecting to the database. Instead of writing the Connection string
manually, you can press the browse button (...) and select the database type from the Data Link Properties window.

![Data Link Properties dialog](image)

*Note:* The list of Database Providers shown in the Data Link Properties window depends on the providers actually installed and available in the computer where you are running IWS. Consult the operating system documentation (or the database documentation) for further information regarding the settings of the Provider for the database that you are using.

- **User name** field: User name used to connect to the database. The user name configured in this field must match the user name configured in the database.
- **Password** field: Password used to connect to the database. The password configured in this field must match the password configured in the database.
- **Retry Interval** field: If IWS is unable to connect to the database for any reason, it retries automatically to connect to the database after the number of seconds configured in this field have passed.
- **Advanced** button: After pressing this button, you have access to customize some settings. For most projects, the default value of these settings do not need to be modified and should be kept.

![Database Configuration: Advanced dialog](image)

- **Time Zone** combo box:
  - **Local Time + Time Difference**: Save the local time on the computer, plus the difference (bias) between the local time zone and Coordinated Universal Time (UTC).
  - **Local Time**: Save the local time only with no bias. This is not recommended.
  - **UTC**: Save the UTC time only. This is the default, and it is strongly recommended for most situations.
• **Milliseconds** combo box: You can configure how the milliseconds will be saved when saving the date in the database. Each database saves the date in different formats; for example, some databases do not support milliseconds in a **Date** field. The following options are available:

  - **Default**: Uses the format pre-defined for the current database. The databases previously tested by InduSoft are previously configured with the most suitable option. When selecting Default, IWS uses the setting pre-configured for the current database type. If you are using a database that has not been previously configured, the Default option attempts to save the milliseconds in a separate field.

    **Tip:** The default option for each database is configured in the **StADOSvr.ini** file, stored in the **\BIN** sub-folder of IWS. See **Studio Database Gateway** for information about how to configure the **StADOSvr.ini** file.

  - **Disable**: Does not save the milliseconds at all when saving the date in the database.
  - **Enable**: Saves the milliseconds in the same field where the date is saved.
  - **Separate Column**: Saves the milliseconds in a separated column. In this case, the date is saved in one field (without the milliseconds precision) and the number of milliseconds is saved in a different column. This option is indicated where you want to save timestamps with the precision of milliseconds but the database that you are using does not support milliseconds for the **Date** fields.

  - **Database Gateway**: Enter the Host Name/IP Address where the IWS Database Gateway will be running. The TCP Port number can also be specified, but if you are not using the default, you will have to configure the IWS Database Gateway with the same TCP Port. See the **Studio Database Gateway** section for information about how to configure the advanced settings for the IWS ADO Gateway.

  - **Disable Primary Keys**: For some modules, IWS will try to define a primary key to the table in order to speed up the queries. If you are using a database that does not support primary keys (e.g., Microsoft Excel), then you should select (check) this option.

  - **Disable Delimiters**: Select this troubleshooting option to disable the delimiters that are used to format communications with the database. Delimiters can cause problems when a Trend Control or Grid builds a query that includes aggregates such as Min and Max.

  - **Disable SQL variables**: Select this troubleshooting option to disable SQL variables, such as `@Value1` and `?`, that are often used in SQL statements and queries. Some specific database providers do not support these variables.

    **Note:** If you select this option, you might need to specify the language or culture that should be used to format values in SQL statements. For more information, see "Advanced Settings" in **Studio Database Gateway**.

**Table Pane**

This area allows you to configure the settings of the Table where the data will be saved. All tasks can share the same database. However, each task (Alarm, Events, Trend worksheets) must be linked to its own Table. IWS does not check for invalid configurations on this field, therefore you should make sure that the configuration is suitable for the database that you are using.

- **Use default name** checkbox: When this option is checked (default), IWS saves and/or retrieves the data in the Table with the default name written in the **Name** field.

- **Automatically create** checkbox: When this option is checked (default), IWS creates a table with the name written in the **Name** field automatically. If this option is not checked, IWS does not create the table automatically. Therefore, it will not be able to save data in the database, unless you have configured a table with the name configured in the **Name** field manually in the database.

- **Name**: Specifies the name of the Table from the database where the history data will be saved.

  **Tip:** To specify a sheet in a Microsoft Excel spreadsheet file, use the following syntax:

  \[sheetname$\]
• **Refresh** button: If the database configured is currently available, you can press the **Refresh** button to populate the **Name** combo-box with the name of the tables currently available in the database. In this way, you can select the table where the history data should be saved instead of writing the Table name manually in the **Name** field.

**Run-Time Pane**

This area allows you set runtime values. The following fields are available:

• **Status** (output) checkbox: The tag in this field will receive one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disconnected from the database. The database is not available; your configuration is incorrect or it is an illegal operation.</td>
</tr>
<tr>
<td>1</td>
<td>The database is connected successfully.</td>
</tr>
<tr>
<td>2</td>
<td>The database is being synchronized.</td>
</tr>
</tbody>
</table>

• **Reload** (output): Specify a reload tag if you are using curly brackets in any of the configuration fields. When you want to reconnect to the database using the updated values on your tags, set the tag on this field to 1. IWS will update the configuration when trying to perform an action in the database, setting the tag back to 0 when it is finished.

**See also:**

Configuring a Default Database for All Task History.
Configuring a Default Database for All Task History

You can configure a Default Database that will save the historical data from all Tasks in a project. After you do, when you create a new Task worksheet, you can choose either to use the Default Database or to configure a new database for that specific worksheet.

To configure the connection settings for the Default Database:

1. On the Project tab of the ribbon, in the Settings group, click Options. The Project Settings dialog is displayed.
2. Click Configure. The Default Database Configuration dialog is displayed.

![Default Database Configuration dialog]

Please refer to Database Configuration dialog for help completing the fields in this window.
**Database Troubleshooting**

IWS database interface provides powerful tools that will help you to identify configuration problems with databases. If you are having problems interfacing with a database, you should first enable the **Database Messages** in the Log window. You can do so by following the steps below:

1. In the IWS Development environment, make sure to show the *Output window* (**Output Window** check box on the View tab of the ribbon).
2. Right-click in the *Output window* (usually located in the lower-right corner of the development environment), and then click *Settings* on the shortcut menu:

3. In the *Log Settings* dialog, check the **Database Messages** option:

After enabling this option, the *Output* window will display error messages related to the database. The FAQ section below lists some common errors that you can see in the *Output* window.

**GENERAL QUESTIONS**

**Q**: I configured my database, but the runtime modules (Alarm, Trend, and Events) are not being saved to the database. I only see the following error message in the *Output* window:

*Database: Error: Error to add new register [CMD_ADD].*

**A**: Most of the database errors in the *Output* window will be followed by additional information such as the SQL command being executed, the Connection String and the Table name. Error messages such as the one described above, will usually happen after a more detailed message. For example, if your Trend task fails to add a register in the database because the cable is disconnected, you should first receive a network error; if the task tries to add more registers before the time specified in the *Retry* field (see **Database Configuration** dialog), it will only display *Database: Error: Error to add new register [CMD_ADD]*. If you think that your configuration is correct, and you want to debug this type of problem, reduce the Retry. Then you should see more detailed information.

**Q**: When I try to access the MySQL database server, I get the following message:

*Object is not set to an instance of an object.*

**A**: This problem was detected under the following conditions:

- A known bug in MySQL Connector/Net v6.1.2 would not correctly specify the charset; and
- The database table you are trying to access doesn't exist.
To solve this problem, make sure you are using MySQL Connector/Net v6.2.0 and that the table you are accessing exists in the database.

**Q:** Why is the Database Interface automatically closing some connections?

**A:** By default, the Database Interface can have a maximum of 1000 connections. When this maximum is exceeded, the oldest connection is automatically closed to allow the new connection and the Output window displays an extended message describing which connection was closed and what was the last command executed.

To increase the maximum number of database connections, open the project file `{project_name}.app` in a text editor and change the following setting:

```
[StDB]
MaxConnections=number_of_connections
```

Keep in mind that increasing the maximum number of connections may decrease project performance.

**Q:** I configured my Connection String using the browser and the Data Link Properties Window. When I click the Test button, it says "Test succeeded". However, when I run my project, the Database Interface displays error messages, and I am not able to save data.

**A:** The Data Link Properties Window uses OLE DB to interface with the Database. IWS Database Interface uses ADO.NET; therefore, you can have the OLE DB provider on your machine and be missing the ADO.NET provider. It is also possible that you are using an ADO.NET provider that is not listed in the `StADOSvr.ini` file. Please refer to Studio Database Gateway for more information about adding ADO.NET providers to the `StADOSvr.ini` file.

**Q:** Why, when I update information in one line in the Grid object, is it updating more than one line in my database?

**A:** The grid object issues an update command in the database using the values in all the columns for the specific row that you are trying to update. If you have rows with duplicate values, you might see this problem. If your table has a primary key or any other unique field that you do not want to display in the Grid object, you can add it to the Columns but specify the Width 0. This will fix the problem.

**Q:** Why do I have to use a separate Column to store the milliseconds on my database?

**A:** Some databases do not support milliseconds in the Time Stamp field. IWS Database interface, by default, requires another column for the milliseconds. If your database can handle milliseconds, or if you do not want to record the milliseconds, you can change the default behavior in the Advanced settings. Note that some databases are able to store milliseconds, but they have lower precision. If you mix different databases with different precisions in redundant mode, you can get synchronization problems.

**Q:** My project works fine when I run in emulation mode. But when I send to the Windows Embedded device, it cannot communicate with my database.

**A:** It might be the case that your Windows Embedded device does not have the .Net Framework or that it does not have the provider that you are using. Try to use the gateway remotely by following the instructions in Linking the Database Through a Remote DB Provider.

**Q:** When I try to connect to the database, why do I receive the message, `Error to create connection class`?

**A:** The .Net Provider that you are trying to use is not installed on your machine. This error message is usually followed by the provider name; if you are using the Sybase database, for instance, the message is followed by `[iAnywhere.Data.AsaClient.AsaConnection]`. The Provider is the `iAnywhere.Data.AsaClient`. You can check if the provider is installed on your machine by going to the Control Panel > Administrative Tools > Microsoft .Net Framework xx Configuration. The provider should be listed in the Assembly Cache.

**Q:** What if I have the provider assembly (usually a .dll file) but it is not listed in the Assembly Cache?

**A:** If your assembly has a strong name, you can register it in the Assembly Cache using the gcautil program. Or it should work if you copy your assembly to the same folder as the `StADOSvr.exe` (usually the `[...]\InduSoft Web Studio v7.1\Bin` folder).

**Q:** I am not able to access my table from the Grid when I use a specific condition. But if no condition is applied, it works fine. Why is that?

**A:** You should check for the following items:
1. Follow the Troubleshooting steps, and look for error messages in the log. An error message can tell you if you have made a mistake, such as entering with a wrong column name or specifying an invalid data format.

2. Some databases have problems when you use reserved words as column names. Therefore, you should avoid using column names such as Time, Date, Numeric, etc.

3. If your column name starts with AND or OR (e.g., ORange), enter the name surrounded by square brackets. For example, instead of ORange=10, enter [ORange]=10.

4. If you are using SQL Server CE, you might have some problems when querying string fields. It has been identified that filters do not work with NCHAR data types; however, they do work if you declare these fields as NVARCHAR(<Number>). You might try to recreate your table by using this data type. An example of a command that creates a table with strings that can be queried is displayed below:

    CREATE TABLE Table1 (Name NVARCHAR(128), Age Numeric, Sex NVARCHAR(1))

**MYSQL**

**Q:** When I try to access the database from my local machine it works fine, but when I move my project to a remote machine, it says Access Denied.

**A:** Each user on a MySQL database has a property associated with it that indicates the computer from which it can get access to the database. By default, this property is set to localhost, so you will only be able to access the database if you are accessing from the local computer. You should read the MySQL manual for information about changing this setting.

**Q:** Sometimes when I try to synchronize a remote MySQL database with a local MySQL database, or if I try to use application redundancy, a connection to the ADO.NET interface is opened and never closed.

**A:** Go to the Database Configuration dialog and uncheck the Automatically Create option.

**SYBASE**

**Q:** I configured my Sybase database using the Browse button. When I click the test button, the test succeeds, but when I try to run my project I get the following error: Database: Error: Parse error: DSN 'MyDatabase' does not exist. What am I doing wrong?

**A:** Please refer to Database Appendix F - Using Sybase for more information about this problem.

**Q:** Why, when I try to connect to the Sybase database, am I receiving the error Error to create connection class [iAnywhere.Data.AsaClient.AsaConnection]?

**A:** You do not have the ADO.NET Provider installed on your computer. The database setup program has an option to install the Provider. Rerun the setup program, and make sure to check that option.

**SQL SERVER CE**

**Q:** Why does the gateway show TypeLoad failure when I try to access my SQL Server CE database?

**A:** This problem usually happens when you do not have the SQL Server CE .NET Provider installed on your CE Device.

**Q:** Why am I getting the error message, There is a file sharing violation. A different process might be using the file?

**A:** You have another program with the SQL Server CE database open. For instance, this will happen if you are using the SQL Server CE configuration software.
Appendices

Using ODBC Databases

Almost every database provides an ODBC interface that can be used to interface with it. The database features provided by IWS can be used with ODBC drivers through the ADO.NET interface for ODBC. In order to use this capability, you must use Microsoft .NET Framework 1.1 or higher.

Note: Microsoft .NET Framework 2.0 is automatically installed with InduSoft Web Studio v6.1+SP5 and later.

The Database Configuration dialog allows you to provide connection strings that will connect to an ODBC DSN. The connection string can be built automatically by clicking on the Browse button (...). When the Data Link window displays, you should select the option Microsoft OLE DB Provider for ODBC Drivers as shown below:
When you click **Next**, the following window will display:

![Data Link Properties, Provider - Connection](image)

Select the DSN that you want to connect to and click **OK**. If you want to specify the user name and password on this window instead of specifying on the **Object Properties** dialog, remember to check the **Allow saving password** checkbox.

**Using Microsoft SQL Server**

IWS Database Interface allows you to retrieve and store information on Microsoft SQL Server relational databases. You should follow the steps below in order to configure the SQL Server database:

1. Click on the **Browse** button in the **Database Configuration Dialog window**. The following window will display:

![Data Link Properties, Provider - SQL Server](image)
2. Select the Microsoft OLE Provider for SQL Server and click **Next**. The following window will display:

![Data Link Properties, Connection - SQL Server](image)

3. Fill out the fields on this window with your database information. If you are not using Windows NT Integrated security, remember to check the **Allow saving password** checkbox to save the password when the **Data Link Properties** window is closed.

4. Click **OK** to finish the **Connection String** configuration.

Your connection string should be very similar to this one:

```
Provider=SQLOLEDB.1; Integrated Security=SSPI; Initial Catalog=MyDatabase; Data Source=192.168.23.200
```

**Note:** These procedures were tested using Microsoft SQL Server 2000.

**Tip:** The Database Gateway ([StADOSvr.exe](#)) now uses an updated time stamp when saving data to Microsoft SQL Server databases, so that milliseconds do not need to be stored in a separate column. However, this only works with Microsoft SQL Server 2008 or later, so if you are using an earlier version of SQL Server, then you must edit the program settings to reverse the change:

1. If your project is running, stop it.
2. Locate the program settings file at: C:\Program Files\InduSoft Web Studio v7.1\Bin\StADOSvr.ini
3. Open the file with a text editor, such as Notepad.
4. Delete the following line:

   ```
   TimeStampType2=DATETIME2
   ```

5. Save your changes, and then exit the text editor.

---

**Using ORACLE Databases**

IWS Database Interface allows you to retrieve and store information on ORACLE relational databases. You should follow the steps below in order to configure the ORACLE database:
1. Click on the **Browse** button in the **Database Configuration Dialog window**. The following window will display:

![Data Link Properties, Provider - Oracle](image1)

2. Select the Microsoft OLE Provider for Oracle and click **Next**. The following window will display:

![Data Link Properties, Connection - Oracle](image2)

3. Fill out the fields on this window with your database information. Remember to check the **Allow saving password** checkbox to save the password when the **Data Link Properties** window is closed. The server name information has the following format:

   **Server/TNS**

   Where:
   
   - **Server**: Computer where the Oracle Database is running
• TNS: Oracle TNS name

⚠️ **Caution:** At the time that this document was written, the **Server Name** field could not be configured with more than 16 letters. If more than 16 letters were specified, you would receive the following error: **ORA-00162: external dbid length 19 is greater than maximum (16)**, where 19 is the number of letters in the **Server Name**. Please see Database Troubleshooting for more hints to work around this problem.

4. Click **OK** to finish the **Connection String** configuration.

---

**Note:** These procedures were tested using ORACLE 10g Release 1.

---

**Using Microsoft Access Databases**

IWS's database interface lets you store information in and retrieve information from Microsoft Access database (ACCDB) files.

You do not need to have Access installed on the same computer as IWS; IWS can directly read from and write to an existing ACCDB file. However, you do need to use Access to create the initial file — either a blank database for new data or a populated database for reference data. Once you've created the file, you can move it to any location and establish a connection to it there.

---

**Note:** This procedure was last tested using Microsoft Access 2007 (12.0.6211.1000).

To establish a connection between your IWS project and your ACCDB file:

1. In the **Database Configuration property sheet**, click the Browse button (…).
   The **Data Link Properties** property sheet is displayed:

   ![Selecting the OLE DB Provider](image)

2. Select the appropriate OLE DB Provider for your database:
   • For a Microsoft Access 2003 database file, select **Microsoft Jet 4.0**.
   • For a Microsoft Access 2007 database file, select **Microsoft Office 12.0 Access Database Engine**.
3. Click **Next**.
The *Connection* tab of the property sheet is displayed:

![Database Configuration dialog](image)

**Specifying the location of the file**

4. In the *Data Source* text box, type the complete file path for your ACCDB file.

   Although the file can be located anywhere on your computer or network, it may be useful to keep it in your project folder. For example:

   ```
   C:\Users\username\Documents\InduSoft Web Studio v7.1 Projects\project_name\Database1.accdb
   ```

   If you do this, however, then you must update the file path whenever you move the project folder.

5. Specify the *User name* and *Password*, if any, for the ACCDB file.

6. Click *Test Connection*.

   If a connection can be successfully established, then an appropriate message is displayed.

7. Click *OK* to save your changes and return to the *Database Configuration* property sheet.

   **Note:** Be sure to select (check) the *Disable Primary Keys* option in the *Database Configuration* dialog. If you do not, then IWS will not be able to connect to your Access database.

   **Important:** Desktop office applications such as Microsoft Access and Microsoft Excel cannot efficiently handle large amounts of data. If you try to save all of your project’s historical data in an Access database or Excel spreadsheet, then the queries will become slow and you might get unexpected results. Therefore, we recommend that Access or Excel be used only as a Secondary Database, with the *Store and Forward* option enabled, or to relay data to third-party software.

   To handle large amounts of historical data, we recommend that you use either IWS's proprietary format or a dedicated relational database such as Microsoft SQL Server or ORACLE.

**Using SQL Server CE**

IWS projects running in EmbeddedView or CEView can interface with Microsoft SQL Server CE (SQL CE) databases on the same device by using the OLEDB Provider for SQL Server CE. This provider must be installed on your Windows Embedded device.
Because the provider is only installed on the device and not on your development workstation, you cannot select it in the *Data Link Properties dialog* when you develop your project. Instead, you must manually enter the connection string using the following format:

<table>
<thead>
<tr>
<th>Version</th>
<th>Connection String</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL CE 2.0</td>
<td>Provider=SQLCE; Data Source=database_path</td>
</tr>
<tr>
<td>SQL CE 3.0</td>
<td>Provider=SQL_CE3.0; Data Source=database_path</td>
</tr>
<tr>
<td>SQL CE 3.5</td>
<td>Provider=SQL_CE3.5; Data Source=database_path</td>
</tr>
</tbody>
</table>

**Note:** The database interface was fully tested with these versions of SQL CE. However, other versions might also be supported; please contact Customer Support for more information.

Examples:

- Access a database file at the fixed location `\Harddisk\MyDatabase.sdf`:

  ```sql
  Provider=SQLCE; Data Source=\Harddisk\MyDatabase.sdf
  ```

  The exact path depends on how your device's non-volatile memory is organized. Please consult the manufacturer's documentation.

- Access a database file at the location specified by the String tag `DatabaseFile`:

  ```sql
  Provider=SQLCE; Data Source={DatabaseFile}
  ```

  The curly brackets (e.g., `{tagname}`) indicate that it is an indirect tag.

Please keep in mind that SQL CE is only intended for lightweight databases and simple transactions. It cannot efficiently handle large amounts of data, such as the Alarm and Event histories. In those cases, we recommend that you either use the Proprietary history format or use a more powerful relational database like the full version of Microsoft SQL Server. If you *must* use SQL CE, then we recommend that you use it only as a Secondary database with the Store and Forward option selected. For more information, see *Saving your alarm history / event log to an external database*.

**Caution:** When using a *Database worksheet* or the *DB/ERP functions* to access a SQL CE database, remember...

- Column/field names must match exactly or the database commands will fail; and
- IWS data types will be converted into possibly unexpected SQL data types. The following table shows how they're converted:

<table>
<thead>
<tr>
<th>IWS</th>
<th>SQL CE</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>!</code> Boolean</td>
<td><code>int</code></td>
</tr>
<tr>
<td><code>!</code> Integer</td>
<td><code>int</code></td>
</tr>
<tr>
<td><code>*</code> Real</td>
<td><code>real</code></td>
</tr>
<tr>
<td><code>*</code> String</td>
<td><code>nvarchar</code></td>
</tr>
</tbody>
</table>
Microsoft SQL Server is not included by the manufacturer on most Windows Embedded and Windows Mobile devices, so if your IWS project requires SQL Server, then you may need to download and install it on the device yourself.

Before you proceed, check the device manufacturer's documentation to see if the device includes Microsoft SQL Server. If it does, then you should not need to install the software again unless you have problems with it.

If it does not, then at least verify the device meets the system requirements for running a IWS project and note the device processor.

Because only the device manufacturer can permanently "install" software on the device using Platform Builder, what you will actually do is copy CAB files to the device's non-volatile memory and then configure the device to load those files every time it starts up.

**Note:** At the time of this writing, the latest stable version of SQL Server is Microsoft SQL Server Compact Edition 3.5 Service Pack 2.

2. Extract and install the software on your desktop. A new directory is created containing the Microsoft SQL Server files, including files that are customized for different device processors.
3. Open the directory: `C:\Program Files\Microsoft SQL Server Compact Edition\v3.5\Devices\wce500`
4. In the directory, select the correct sub-directory for your device processor. For example, if your device uses an x86 processor, then select the x86 sub-directory.
5. In the sub-directory, select the CAB files that need to be copied to your device. Continuing with the x86 example, select the following files:
   - `sqlce.dev.ENU.wce5.x86.CAB`
   - `sqlce.repl.wce5.x86.CAB`
   - `sqlce.wce5.x86.CAB`
6. Copy the CAB files to your device's non-volatile memory and then note where they are located. The exact method you use to copy files to the device depends on your setup. You might use desktop syncing or a network connection or a USB flash drive. Check the manufacturer's documentation to see what methods are available.
7. On your desktop, create a new text file named `install.bat`.
8. Edit `install.bat` to create a batch script that loads the CAB files on startup. Continuing with the x86 example, edit the file to contain the following code:

   ```batch
   wceload /noui /delete 0 "filepath\sqlce.dev.ENU.wce5.x86.CAB"
   wceload /noui /delete 0 "filepath\sqlce.repl.wce5.x86.CAB"
   wceload /noui /delete 0 "filepath\sqlce.wce5.x86.CAB"
   ```

   ...where `filepath` is the location to which you copied the CAB files.

   For more information, see the MSDN article about the Wceload tool.
9. Copy `install.bat` to your device's `\Windows\Startup` directory. If the device does not have a `\Windows\Startup` directory, then follow the manufacturer's documentation for configuring startup items.
10. Restart the device. If you have correctly selected and installed the files, then SQL Server will run on startup on you should be able to access it with your IWS project.
**Using Sybase**

You need to install the AsaClient provider on your computer; the tests with IWS were performed using the architecture explained in the topic **Linking the Database Through a Remote DB Provider**.

If you are using the browse button to automatically generate the connection string, the string returned will have the following format:

```
Provider=ASAProv.90; Data Source=Test
```

This format requires that you create an ODBC DSN with the same name as the **Data Source** (in this case, **Test**) in order to communicate with the database. If the DSN is not created, the following error will display in the LogWin when connecting to the database:

```
Database: Error: Parse error: DSN 'Test' does not exist
```

To void an ODBC DSN, you can enter with the connection string manually as shown in the example below:

```
Provider=ASAProv.90; DBF=C:\ Test.db
```

Note: These procedures were tested using Sybase Server Anywhere 9.0.1.1751.

**Using Microsoft Excel**

IWS’s database interface lets you store information in and retrieve information from Microsoft Excel spreadsheet (XLS or XLSX) files.

You do not need to have Excel installed on the same computer as IWS; IWS can directly read from and write to an existing XLS/XLSX file. However, you do need to use Excel to create the initial file — either a blank spreadsheet for new data or a populated spreadsheet for reference data. Once you’ve created the file, you can move it to any location and establish a connection to it there.

Note: This procedure was last tested using Microsoft Excel 2007 (12.0.6331.5000).

To establish a connection between your IWS project and your XLS/XLSX file:

1. In the **Database Configuration property sheet**, click the Browse button (...).
The *Data Link Properties* property sheet is displayed:

![Data Link Properties](image)

**Selecting the OLE DB Provider**

2. Select the appropriate OLE DB Provider for your database:
   - For a Microsoft Excel 2003 spreadsheet file, select *Microsoft Jet 4.0*.
   - For a Microsoft Excel 2007 spreadsheet file, select *Microsoft Office 12.0 Access Database Engine*.

3. Click **Next**.

   The *Connection* tab of the property sheet is displayed:

![Connection Tab](image)

**Specifying the location of the file**

4. In the **Data Source** text box, type the complete file path for your XLS/XLSX file.
Although the file can be located anywhere on your computer or network, it may be useful to keep it in your project folder. For example:

```
C:\Users\username\Documents\InduSoft Web Studio v7.1 Projects\project_name\Book1.xlsx
```

If you do this, however, then you must update the file path whenever you move the project folder.

5. Click the **All** tab.

6. Select **Extended Properties** and then click **Edit Value**.

   The **Edit Property Value** dialog is displayed.

7. In the **Property Value** text box, type one of the following values:
   - For a Microsoft Excel 2003 spreadsheet file, type **Excel 11.0**.
   - For a Microsoft Excel 2007 spreadsheet file, type **Excel 12.0**.

8. Click **OK** to close the **Edit Property Value** dialog.

   ![](image)

   Editing the value of Extended Properties

9. Click the **Connection** tab.

10. Click **Test Connection**.

    If a connection can be successfully established, then an appropriate message is displayed.

11. Click **OK** to save your changes and return to the **Database Configuration** property sheet.

    **Note:** Be sure to select (check) the **Disable Primary Keys** option in the **Database Configuration** dialog. If you do not, then IWS will not be able to connect to your Excel spreadsheet.

    **Important:** Desktop office applications such as Microsoft Access and Microsoft Excel cannot efficiently handle large amounts of data. If you try to save all of your project's historical data in an Access database or Excel spreadsheet, then the queries will become slow and you might get unexpected results. Therefore, we recommend that Access or Excel be used only as a Secondary Database, with the **Store and Forward** option enabled, or to relay data to third-party software.

    To handle large amounts of historical data, we recommend that you use either IWS's proprietary format or a dedicated relational database such as Microsoft SQL Server or ORACLE.
**Using MySQL**

IWS can interface with MySQL databases, but to do so, you must install an ADO.Net provider for MySQL.

The provider required by IWS is MySQL Connector/Net, and at the time of this writing, the necessary software can be downloaded from the official MySQL site. (Please note that the linked site is beyond our control and may change without notice.)

Once the provider is installed, you can use the **Database Configuration property sheet** to configure a MySQL database connection. However, unlike for other database types, you cannot use the **Data Link Properties** dialog (which is accessed by clicking ... to the right of the **Connection string** box) to form the connection string. Instead, you must directly enter the connection string using this basic format:

```
Provider=MYSQCLIENT; Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

The following optional parameters can be appended to the connection string:

**Optional parameters for the MySQL Connector/Net connection string**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port=number;</td>
<td>Specifies what port to use for the connection. The default port is 3306, but any port can be specified as long as it matches the server configuration.</td>
</tr>
<tr>
<td></td>
<td>If a port of -1 is specified, then the connection will use the named pipes network protocol (see <strong>Protocol</strong> below).</td>
</tr>
<tr>
<td>Server=myServerAddress1 &amp; myServerAddress2 &amp; ... &amp; myServerAddressN;</td>
<td>Use any server in a replicated server configuration.</td>
</tr>
<tr>
<td>Encryption=true;</td>
<td>Enables SSL encryption for all data sent between the client and the server. The server must have a valid certificate installed.</td>
</tr>
<tr>
<td>Encrypt=true;</td>
<td>An alternative to <strong>Encryption</strong> above, in case there are errors.</td>
</tr>
<tr>
<td>Default Command Timeout=millisecon;ds;</td>
<td>Specifies a default command timeout for the connection. This does not supersede any timeout properties on individual commands.</td>
</tr>
<tr>
<td>Connection Timeout=seconds;</td>
<td>Specifies how long the client will wait for a server connection before terminating the attempt.</td>
</tr>
<tr>
<td>Ignore Prepare=true;</td>
<td>Instructs the database provider to ignore <strong>Command_Prepare</strong> statements, to prevent corruption from server-side prepared commands.</td>
</tr>
<tr>
<td>Protocol=myProtocol;</td>
<td>Specifies which network protocol to use. The default is <em>socket</em> or <em>tcp</em>, but you can specify <em>pipe</em> to use a named pipes connection or <em>memory</em> to use a shared memory object.</td>
</tr>
<tr>
<td>Shared Memory Name=MySQL;</td>
<td>Specifies the name of the shared memory object to be used for communication. (This parameter applies only if the <strong>Protocol</strong> parameter above is set to <em>memory</em>.)</td>
</tr>
<tr>
<td>CharSet=UTF8;</td>
<td>Specifies which character set to use to encode queries to the server. Please note that query results are encoded in the same character set that the data itself is recorded.</td>
</tr>
</tbody>
</table>

**Note:** These procedures were tested using MySQL v5.1.11 and MySQL Connector/Net v6.2.0.
If you do find yourself in need of technical assistance, there are certain things that you will need to know before you contact technical support. Regardless of the problem, you will need to know the sequence of events that led to you discovering the problem. It must be explained in as much detail as possible and you should be careful not to ad-lib, as it may drastically affect troubleshooting time and procedures. It's also best to be in front of the computer you are having problems with, and to keep a pen and paper handy.

**Before Contacting Technical Support**

Some things you should try before you contact technical support are:

- **Check out the documentation**
  
  The application help and release notes can be accessed on the Help tab of the ribbon, and more documentation is available on our website. You may find that your particular issue has already been documented.

- **Consider recent changes on your system**
  
  If something used to work, think about what may have changed. New software installation or general system changes can affect performance and general functionality of other software on your system.

- **Try reproducing the problem in a new file**
  
  If the problem cannot be reproduced in a new test file, compare the new file with your original file to find and eliminate the differences. This will help narrow down the cause of the issue.

- **Try reproducing the problem on another machine**
  
  If the problem goes away on another machine, compare what is different between the two systems. If this is the case, there is most likely a system conflict.

**Verifying Your Project**

If you change, reorder or delete any tags in the Tags database, or if you reconfigure any settings in the Web tab of the Project Settings dialogue, then you must verify your project to realign all of your screens and worksheets to the current state of your database. On the Home tab of the ribbon, in the Tools group, click **Verify**.

**Related Publications**

The *IWS Getting Started Guide* is designed for first-time users. This publication contains information about the basic functions of InduSoft Web Studio, and it is provided in the **Documentation** folder on the InduSoft Web Studio installation CD.

The driver **User Guides** explain how to configure the direct communication drivers, according to their unique protocol characteristics. One customized user guide is included with each driver. These publications are provided in the *\[\ldots\]\InduSoft Web Studio v7.1\Drv* directory, or from the Help tab of the ribbon. (On the Help tab of the ribbon, in the Documentation group, click **Communication Drivers**. When the **Object Properties** dialog is displayed, highlight the driver you’re using and then click **Help**.)

**Contacting Technical Support**

If you cannot find an answer to your technical question in the product documentation or help system, our Technical Support Specialists are available to assist any customer with current product maintenance. The telephone number is 1-877-INDUSOFT (1-877-463-8763).
Please try to define the problem before you contact Technical Support so that you can repeat the steps that led to the problem and specifically identify when and how the problem occurred. The support representative will need to know exactly what the problem is in order to provide help. These steps will help us pinpoint and solve your problem more quickly.

Please have the following information available:

• Hardware environment: available memory, processor type, output device
• Software environment: operating system, version of Windows®, network platform
• Product name, version number, and product registration number
• Amount of memory installed on your system
• Amount of free hard disk space on your system
• Screen resolution (screen size in pixels, for example, 1024 by 768)
• Screen color depth (number of colors or bits, for example, 256 colors or 8-bit color)
• Graphics card manufacturer, model name, and driver version number
• Sound card manufacturer and model name
• A list of external devices connected to the computer
• Brief description of the problem or error, and the specific text of any error messages
• Description of the steps you have taken to troubleshoot the issue, for example, how many machines you have tested on, and whether the issue is reproducible in a new file
• Steps to reproduce the issue, if it is reproducible. If the issue is not reproducible, it may be an development issue rather than an issue with the product.

If your project crashes completely during runtime, it will generate a debugger report and save it to:

\[
\text{[...]}\text{Documents}\text{\backslash InduSoft Web Studio v7.1 Projects\project_name\Web\Dump\WindDump.dmp}
\]

Please have this file ready to send to Technical Support for analysis.

When you contact us, please have your system information ready. You can get this by using the Support Information command located in the Help menu.

If your problem or question is not urgent, please go to our website at: www.indusoft.com
Frequently Asked Questions

Database & Security System

What does the Shared Tags folder store?

The Shared Tags folder stores the tags imported from the PC-based Control linked to the IWS project. The PC-based Control is linked to the project by the New Project wizard.

How do I count how many tags are configured in the project database?

The number of tags currently used in the project is displayed in the status bar at the bottom of the development environment. Individual array elements and class members are counted as tags.

Graphics

How do I insert and configure an ActiveX object in a project?

To insert an ActiveX object in a project screen:

1. On the Graphics tab of the ribbon, in the Libraries group, click ActiveX Control.
2. Select the ActiveX control that you want to insert from the list, and then click OK. The ActiveX object will then appear on the screen. (Unregistered ActiveX objects will not be available in this list box.)
3. Double-click on the ActiveX object and assign a name to it (enter a value in the Name field). The animations and methods list can be viewed by selecting the Methods button. The static properties can be set by the Properties button (A detailed description about the objects properties can be found in the component documentation, provided by the component developer).

There are three functions to access the ActiveX component during runtime:

- **XGet(strName, strProperties):** Returns the value of the properties strProperties from the object strName. The list of properties which can be read from the object are listed in the Object Properties dialog from the object, with the syntax Properties Name(PropGet) (for example, Color(PropGet)).

- **XSet(strName, strProperties, Value):** Writes the value Value to the properties strProperties of the object strName. The list of properties which can be set to the object are listed in the Object Properties dialog from the object, with the syntax Properties Name(PropPut) (for example, Color(PropPut)).

- **XRun(strName, strMethod, Parameter1, Parameter2, ..., ParameterN):** Executes the method strMethod from the object strName, according to the parameters Parameter1, Parameter2, ..., ParameterN. The list of methods available in the object is listed in the Object Properties dialog from the object, with the syntax Method Name(Method) (for example, OpenFile(Method)).

Tip: Before inserting an ActiveX control (usually an OCX file) into the project, make sure it has been properly registered in the computer. To register an ActiveX control from with the development application, click Register Controls on the Home tab of the ribbon.

Note: The amount of parameters set in the XRun function can vary from 0 up to 255 and it depends each the ActiveX component. It's possible to use tags to set the parameters; however, the tag type must match the component parameter type (Boolean, integer, string or real).

How do I designate one screen that will open each time I start the project?
On the Project tab of the ribbon, in the Settings group, click Viewer, and then in the Project Settings dialog, type the startup screen name in the Startup screen box.

How do I insert a background picture on the screen?

Right click on the screen and select the option Screen Attributes from the popup menu. Enable the checkbox Enable Background and choose the picture format in the combo-box besides this label. Copy the picture file to the Screen sub-folder of the project and rename it with the same name of the screen (ScreenName.scr file). Using the Shared image option, it’s possible to copy a bitmap file to the Screen sub-folder and share this picture with more than one screen. In this case, it’s necessary to type the bitmap name in the Shared image field.

Why are my linked pictures no longer displayed on Windows Embedded devices?

InduSoft Web Studio v7.1+SP3 uses a new drawing system to display images on Windows Embedded devices. This new system requires that your device is running a recent version of Windows Embedded and that all of the supported image file formats are included in the device’s run-time image. If an image is not displayed correctly on your device, there are two possible solutions. First, you may disable the new drawing system and use the legacy drawing system instead. If you do this, however, only JPG, PNG, and BMP images will be supported in your project. To use the legacy system, open your project file (projectname.APP) in a text editor and then edit the following property:

```
[Objects]
DrawingSystemID=<0 for new system, 1 for legacy system>
```

Second, you might need to update the run-time image on your Windows Embedded device in order to include all of the supported image file formats. (The run-time image is typically set by the device manufacturer.) That is beyond the scope of this documentation, however; please refer to the technical note Image Requirements for IWS v7.1 Runtime on Embedded Systems, which is available on the InduSoft Web Studio installation DVD and the InduSoft website (www.indusoft.com).

Tasks

How do I convert a trend history from the proprietary binary file to an ASCII text file?

In Windows, select the history file (*.HST) that you want to convert, and then drag it to the HST2TXT utility in the program folder ([...]\InduSoft Web Studio v7.1\Bin\HST2TXT.exe). The converted file will be automatically created in that folder.

Alternatively, you can use the HST2TXT function in a Script or Math worksheet to convert files automatically during project run time.

How do I set a DATE field for an ODBC interface with an Oracle package?

Configure the "Column" cells in the ODBC worksheet with the syntax ColumnName.ts (for example: MyDate.ts).

How do I execute a Math worksheet during the startup and another Math worksheet during the project shutdown?

- **Startup**: Execute a Math worksheet during the startup by creating a Math worksheet and filling in its Execution field with the expression <TagName>=0 (for example, StartTag=0 ). In the last line of the Math worksheet, set the value 1 to the <TagName> tag. The <TagName> tag type should be Boolean.

- **Shutdown**: Instead of executing the ShutDown() function directly, execute one Math worksheet and configure the ShutDown() function in the last line of this Math worksheet.

Communication

How do I set a "communication error" alarm?
Configure a tag in the Write Status or Read Status field of the driver worksheets and configure an alarm whenever this tag is not 0 (zero).

How do I start and stop communication drivers during runtime?

There are three functions available to handle the execution of the communication drivers during runtime:

• Start all drivers configured in the project: \texttt{StartTask("Driver")}
  
  For example: \texttt{StartTask("Driver")}

• Start a specific driver configured in the project:
  
  \texttt{WinExec("StudioPath\Bin\Studio Manager.exe" + " + " + StudioPath\bin \Driver.dll" + " + " + DriverName")}

  For example:

  \texttt{WinExec( Asc2Str(34) + "C:\Program Files\InduSoft Web Studio v7.1\Bin \Studio Manager.exe" + Asc2Str(34) + " + Asc2Str(34) + "C:\Program Files\InduSoft Web Studio v7.1\Bin\Driver.dll" + Asc2Str(34) + " + Asc2Str(34) + "MODBU" + Asc2Str(34) )}

  \textbf{Note:} The \texttt{Asc2Str(34)} function is used to concatenate quotation marks for paths where there are space chars.

• Stop a specific driver configured in the project: \texttt{EndTask("DriverDriverName")}
  
  For example: \texttt{EndTask("DriverMODBU")}

  \textbf{Tip:} You can start or stop other tasks using the \texttt{StartTask(TaskName)} and \texttt{EndTask(TaskName)} functions. For example, \texttt{StartTask("Viewer")}, \texttt{Endtask("Viewer")}.

What are the parameters of the IWS DDE Server?

The IWS DDE Server and NetDDE Server parameters are shown in the table below:

<table>
<thead>
<tr>
<th>Comm. Type</th>
<th>Application</th>
<th>Topic</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network DDE</td>
<td>(&lt;\text{Computer Name}&gt;\text{NDDE$})</td>
<td>UNISOFT$</td>
<td>$\text{TagName}$</td>
</tr>
<tr>
<td>Local DDE</td>
<td>UNIDDE</td>
<td>DB</td>
<td>$\text{TagName}$</td>
</tr>
<tr>
<td>Network DDE</td>
<td>(&lt;\text{Computer Name}&gt;\text{NDDE$})</td>
<td>UNISOFT$</td>
<td>$\text{TagName}$</td>
</tr>
<tr>
<td>Local DDE</td>
<td>UNIDDE</td>
<td>DB</td>
<td>$\text{TagName}$</td>
</tr>
</tbody>
</table>

How do I exchange data with Excel by using NetDDE?

NetDDE can be used to exchange data, via the DDE protocol, between networked stations.

1. Start the DDEServer module from the development application (Tasks on the Home tab of the ribbon).

2. Run Excel in the remote station.

3. Open an Excel worksheet and fill the cells that must exchange data with IWS, using the following syntax:

   \[
   ='\text{computer name}\text{\textbackslash NDDE$}'!'\text{UNISOFT$}'!'\text{tag name}
   \]
For example:

```bash
='\PC\NDDE$'|'UNISOFT$'!second
```

**Note:**
- When running under Windows NT or Windows2000, it is necessary to make sure that the services Network DDE and Network DDE DSDM are started. (Use the Services shortcut from Control Panel to start these services).
- When running under Windows 98, it is necessary to run the program `WindowsPath etdde.exe` in both computers (for example, `c:\Windows etdde.exe`).

---

**Is the IWS OPC interface compliant with OPC specification v1.0a or v2.0?**

The IWS OPC Client and OPC Server modules are compliant with both OPC specification v1.0a and v2.0.

**How do I get errors from Intellution / GE Fanuc iFIX applications?**

If your project is communicating via TCP/IP with an iFIX application, then you should add the following key to your project file (i.e., `project_name.APP`):

```
[TCP]
SetQualityToBadOnError=1
```

After you do this, if the iFIX application generates an error during runtime, then the quality of the affected tags in your project will be set to BAD. You can get this information by reading the `Quality` tag field (i.e., `tagname->Quality`).

**General**

**What operating systems are compatible with InduSoft Web Studio, EmbeddedView, and CEView?**

See table below. "N" means that the operating system is NOT supported, and "Y" means that the operating system is supported.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>InduSoft Web Studio</th>
<th>EmbeddedView or CEView</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>v2.x</td>
<td>v3.x thru v4.1</td>
</tr>
<tr>
<td>Name</td>
<td>Version</td>
<td>v2.x</td>
</tr>
<tr>
<td>Windows Vista</td>
<td>Any</td>
<td>N</td>
</tr>
<tr>
<td>Windows XP</td>
<td>Any</td>
<td>N</td>
</tr>
<tr>
<td>Windows 2000</td>
<td>Any</td>
<td>N</td>
</tr>
<tr>
<td>Windows NT</td>
<td>v4.0+SP4 or higher</td>
<td>Y</td>
</tr>
<tr>
<td>Windows ME</td>
<td>Any</td>
<td>N</td>
</tr>
<tr>
<td>Windows 98</td>
<td>Any</td>
<td>N</td>
</tr>
<tr>
<td>Windows 95</td>
<td>Any</td>
<td>Y</td>
</tr>
<tr>
<td>Windows CE</td>
<td>v2.12</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>v3.x</td>
<td>N</td>
</tr>
<tr>
<td>Windows CE.Net</td>
<td>v4.0</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>v4.1</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>v4.2</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>v5.0</td>
<td>N</td>
</tr>
</tbody>
</table>
How do I start IWS automatically when the computer is powered on?

Create a shortcut to \[\ldots\]\InduSoft Web Studio v7.1\Bin\RunStudio.exe and then place it in your Startup folder \{C:\WINDOWS\Documents and Settings\All Users\Start Menu \Programs\Startup\}.

What features are not supported by EmbeddedView and CEView?

The main features not supported by EmbeddedView and CEView are: DDE, NetDDE, ODBC, PasteLink, and a number of functions.

How do I show a splashwindow when starting a project in EmbeddedView or CEView?

To enable your project to show a splash window during startup, add the following key to the device's CEView.ini file:

```
[OEM]
SplashWnd = Path to bitmap File // default is Splash.bmp
SplashWndTime = Time in milliseconds // default is 1000
```

How do I see runtime messages and errors on a "blind" Windows Embedded device — that is, a device that has no display?

If the device has a network connection, then you can use the Remote LogWin tool to view the device's messages as they are generated.

You can also save the messages to a dump file \(dump.txt\) and access the file at your convenience. To create the file, add the following key to the device's CEView.ini file:

```
[OEM]
BlindDevice = 1
```

What are the main steps to create a Web-based application?

Follow the procedure below:

1. Develop the project locally. Don’t use features that are not supported by Thin Clients for the screens which will be saved as HTML format.
2. After saving the screens in the standard format \(\text{Save} \) in the Application menu), save the screens that must be available for the Thin Client in HTML format \(\text{Save as HTML} \) in the Application menu).
3. On the Project tab of the ribbon, in the Web group, click Thin Client.
4. In the Data Server IP box, type the IP address of the Server station (i.e., the station where the project is running).
5. On the Home tab of the ribbon, click Tasks, and then set the TCP/IP Server module as Startup=Automatic.
6. Open the project tags database and set the option Server instead of Local in the Scope column for all tags that must exchange value between the Server and the Thin Client station.
7. Verify the project (Verify on the Home tab of the ribbon) to update the HTML files with these new settings.
8. If there is no Web Server running on the computer, copy the program NTWebServer.exe from the Bin sub-folder of the program directory into the Web directory \(e.g., \{project name\}\Web\) and run it. The path where this Web Server program is executed will be the root directory
of the server. The Web Server is necessary to export data (web files) in HTTP protocol to the Thin Clients.

9. Run the project on the Server station.

10. Using a browser (for example, Internet Explorer v4.1+SP1 or newer version) in the Thin Client station, type the URL address to download the screen that had been saved in HTML format (for example, http://ServerIPAddress/ScreenName.html).

**Note:** The Thin Client requires an ActiveX component (ISSymbol.ocx) to handle the screens on the browser. If the Thin Client is connected to the Internet, this component is downloaded and registered automatically. Otherwise, it’s necessary to copy it to the \OSPath\System32 directory of the Thin Client and register it by the command regsvr32 ISSymbol.ocx. This file can be found in the BIN folder from the IWS installation directory.

**How do I maintain communication between a Thin Client connecting via proxy and a Web Gateway application running on Microsoft IIS?**

Microsoft Internet Information Services (IIS) has a configuration option to keep HTTP connections alive. When this option is enabled, it may conflict with Thin Clients that are connecting via proxy. To disable this option:

1. Start Internet Services Manager.
2. In the Internet Information Services window, open the local server (*server name*).
3. Right-click on Default Web Sites and select Properties from the shortcut menu. The Default Web Site Properties dialog is displayed.
4. Select the Web Site tab of the Default Web Site Properties dialog.
5. In the Connections pane of the Web Site tab, uncheck the HTTP Keep-Alives Enabled option.
6. Click OK to save the change and close the dialog.

**How do I send an email from the IWS project?**

Follow the procedure below:

- Execute the function `CNFEMail(strSMTP,strFrom,strPOP3,strUser,strPassword,numTimeOut)` to configure the overall parameters used to send emails. After executing this function once, the parameters set by it are kept in the system until the project is shut down. So, most projects execute this function just once, after starting the project;

- Execute the function `SendEMail(strSubject,strMessage,strTO)` and/or `SendEMailExt(strSubject,strMessage,strTO,strCC,strBCC,strFile1,...,strFileN)` each time that an email message must be sent. The main difference between both functions are listed in the next table:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>SendEmail</th>
<th>SendEmailExt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Supports Subject text</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Supports Message text</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Supports TO addresses</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Supports CC addresses</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Supports BCC addresses</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Supports attached files</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

The runtime task (TCP/IP, OPC, DDE, ODBC, etc) does not work.
Make sure the runtime task is set to **Automatic** in the *Execution Tasks* dialog (*Tasks* on the Home tab of the ribbon). Select a runtime task that must be executed (for example, TCP/IP Server), click **Startup**, and then set it as **Automatic**.

**The browser of the Thin Client launches an error message missing the ISSymbol.ocx and does not display the screens from the Server.**

**ISSymbol.ocx** is the ActiveX object used by the browser from the Thin Client to view the web pages. If the Thin Client is connected to the Internet, the **ISSymbol.ocx** control is automatically downloaded and registered in the Thin Client station. Otherwise, it's necessary to copy it to the \WinNT\System32 folder of the Thin Client station and register it manually. Once it is registered your browser will be able to see the pages.

![Note](image)

**Note:** Use the command `regsvr32 ISSymbol32.ocx` to register the ActiveX component in the Thin Client.

**The screens are shown on the Thin Client (Browser); however, the data (tags values) are not read from the Server.**

Make sure the parameter in the column **Scope** from the project tags database is set as **Server** instead of **Local**. The tags set as **Server** keep the same value in the Server and in the Thin Client (Browser). The tags set as **Local** have independent values in the Server and in the Thin Client (Browser).

![Caution](image)

**Caution:** It's necessary to verify the project (**Verify** on the Home tab of the ribbon) after modifying the tags settings. Otherwise, the changes will not be updated in the Web pages.

**The "On Up" expressions configured in the Command animation are not executed.**

The "On Up" expressions from the Command animation are not executed if the mouse pointer is dragged out the object area before releasing it. If the checkbox **Release from the Command Object Properties** window is enabled, the On Up expression is executed even if the mouse pointer is dragged out the object area before releasing it.

**The Trend History does not work after adding or removing tags in the Trend worksheet.**

When a tag is inserted or removed FROM a *Trend* worksheet, the format of the history files (*.*.hst) is modified. The same .hst file cannot have two different formats; otherwise, the data will not be retrieved from it properly by the Trend object. If you need to add or remove tags for history files, there are two valid procedures: Create a new *Trend* worksheet or delete the old *.hst* files.

**Which functionalities are not supported by PocketPC platforms?**

Windows CE devices powered PocketPC do not support some functionality which are supported by Windows CE devices powered by the "standard" Windows CE version:

<table>
<thead>
<tr>
<th>Functionality not supported by PocketPC devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCOM (Distributed Component Object Model): It means that all features based on DCOM (for example, remote OPC communication) are not supported by PocketPC devices.</td>
</tr>
<tr>
<td><strong>DialGetClientIP()</strong> function does not work for PocketPC devices</td>
</tr>
</tbody>
</table>

---

Page 644
Help tab

The Help tab of the ribbon provides additional help with using the software.

![Help tab of the ribbon](image)

The tools are organized into the following groups:

- **Documentation**: Access the documentation for the development application, including this help file / technical reference and notes for the individual communication drivers.
- **Information**: Access other information about InduSoft Web Studio, including the license agreement, product website, and release notes, as well as system and support details that make it easier for Customer Support to assist you.

**Technical Reference**

To open the help system for the InduSoft Web Studio software, click Help on the Help tab of the ribbon.

![Tip: This documentation is also available as a PDF on the InduSoft Web Studio installation CD.](image)

**Communication Drivers**

To see the available documentation for the communication drivers, click Communication Drivers on the Help tab of the ribbon.

![Communication Drivers dialog](image)

From this dialog, you can select an installed driver then click the Help button to open Adobe Acrobat® Reader™ and display a detailed document about that driver in PDF format.

**License Agreement**

To display a PDF copy of the InduSoft Web Studio software license, click License Agreement on the Help tab of the ribbon.

**Home Page**

To go to the InduSoft company site, click Product Web Site on the Help tab of the ribbon.
**Release Notes**

To view the release notes for InduSoft Web Studio, click Release Notes on the Help tab of the ribbon.

The release notes are distributed as a styled XML file, so you must have an XML-capable browser like Internet Explorer 7 or Internet Explorer 8 to view the file as intended.

**System Information**

Select System Information to open the System Information window, which provides information about the following:

- **System Information**: Displays details about your operating system.
- **Network Resources**: Displays details about your computer’s network.
- **Applications**: Lists the applications that are running.
- **Processes**: Displays all Windows tasks that are running.
- **Services**: Lists the Windows NT/2000 services being used by IWS (Windows NT/2000 only).

![System Information Window](image)

**Note**: Although you open the System Information window from IWS, this window provides general information about the local station and the network only. The System Information window does not provide specific information about the application.

**Support Information**

The Support Information dialog displays basic information about your computer’s operating system, your InduSoft Web Studio installation and license, and your project settings. If you need to contact Customer Support, then you should have this information ready to answer their questions.
To open the dialog, click **Support** on the Help tab of the ribbon. The dialog will be displayed:

![Support Information dialog](image)

**Support Information dialog**

To copy the information to the Clipboard, click **Copy**. You can then paste the information into another window or text field, such as the body of an email message.

To save the information to a file, click **Save to File**. A standard **Save As** dialog will be displayed.

When you are done, click **Close**.

**About**

To get more information about the InduSoft Web Studio software, click **About** on the Help tab of the browser.

![About dialog](image)
This section describes the functions that make up the built-in scripting language. Each function description includes complete syntax, possible returned values, and examples of usage.
Logic and arithmetic operators

The built-in scripting language supports the following logic and arithmetic operators.

### Logic operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>A AND B</td>
<td>TRUE if A and B are both TRUE</td>
</tr>
<tr>
<td>OR</td>
<td>A OR B</td>
<td>TRUE if A is TRUE, or B is TRUE, or both</td>
</tr>
<tr>
<td>XOR</td>
<td>A XOR B</td>
<td>TRUE if A is TRUE, or B is TRUE, but not both</td>
</tr>
<tr>
<td>NOT</td>
<td>NOT A</td>
<td>TRUE if A is FALSE</td>
</tr>
<tr>
<td>=</td>
<td>X = Y</td>
<td>TRUE if X is equal to Y</td>
</tr>
<tr>
<td>&gt;</td>
<td>X &gt; Y</td>
<td>TRUE if X is greater than Y</td>
</tr>
<tr>
<td>&gt;=</td>
<td>X &gt;= Y</td>
<td>TRUE if X is greater than or equal to Y</td>
</tr>
<tr>
<td>&lt;</td>
<td>X &lt; Y</td>
<td>TRUE if X is less than Y</td>
</tr>
<tr>
<td>&lt;=</td>
<td>X &lt;= Y</td>
<td>TRUE if X is less than or equal to Y</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>X &lt;&gt; Y</td>
<td>TRUE if X is not equal to Y</td>
</tr>
<tr>
<td>&amp;</td>
<td>X &amp; Y</td>
<td>Bitwise AND:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101 (decimal 5) AND 0011 (decimal 3) = 0001 (decimal 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101 (decimal 5) OR 0011 (decimal 3) = 0111 (decimal 7)</td>
</tr>
<tr>
<td>^</td>
<td>X ^ Y</td>
<td>Bitwise XOR:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101 (decimal 5) XOR 0011 (decimal 3) = 0110 (decimal 6)</td>
</tr>
<tr>
<td>~</td>
<td>~ X</td>
<td>Bitwise NOT:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOT 0101 (decimal 5) = 1010 (decimal 10)</td>
</tr>
<tr>
<td>&gt;&gt; n</td>
<td>X &gt;&gt; Y</td>
<td>Rotate n bits to right:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0110 (decimal 6) ROTATE RIGHT = 0011 (decimal 3)</td>
</tr>
<tr>
<td>&lt;&lt; n</td>
<td>X &lt;&lt; Y</td>
<td>Rotate n bits to left:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0110 (decimal 6) ROTATE LEFT</td>
</tr>
</tbody>
</table>
### Arithmetic operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>X + Y</td>
<td>Add (plus)</td>
</tr>
<tr>
<td>-</td>
<td>X - Y</td>
<td>Subtract (minus)</td>
</tr>
<tr>
<td>*</td>
<td>X * Y</td>
<td>Multiply by</td>
</tr>
<tr>
<td>/</td>
<td>X / Y</td>
<td>Divide by</td>
</tr>
</tbody>
</table>

Arithmetic operators are resolved from left to right according to the standard order of evaluation. To change the order, enclose in parentheses the part of the equation to be resolved first. For example, the following equation produces a result of 11 because multiplication is evaluated before addition; the equation multiplies 2 by 3 and then adds 5 to the result:

\[ 5 + 2 \times 3 \]

In contrast, if you use parentheses to change the syntax, 5 and 2 are added together and then multiplied by 3 to produce 21:

\[ (5 + 2) \times 3 \]

**Tip:** For more complex math, try the Arithmetic, Statistical, Logarithmic and Trigonometric functions.
How to read function descriptions

This is a key to reading the descriptions of the built-in scripting functions.

Each function description is broken into several sections.

**Function attributes**

Every function has certain attributes that are described in a single-row table:

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>function name</td>
<td>group name</td>
<td>synchronous or asynchronous</td>
<td>supported or not supported</td>
<td>supported or not supported</td>
<td>supported or not supported</td>
<td>supported or not supported</td>
</tr>
</tbody>
</table>

First, obviously, the exact name of the function as it should be used in your project.

Next, the functions are organized into groups according to the type of calculation they perform or the part of your project upon which they act. You can use the group names to find the functions you want in the Object Finder and in this documentation.

Next, the execution of the function is either synchronous or asynchronous:

**Synchronous**

When the function is executed on either the project server or the project client, that station requires some response or acknowledgement from the other. The project pauses, however briefly, while it waits for the response. In other words, the server and client must remain synchronized.

This is normally not an issue because most functions are executed almost instantly, but if a client makes unusually frequent function calls or your network is slow, your project may suffer decreased performance.

**Asynchronous**

The function can be executed on either the project server or the project client without waiting for the other. The project continues to run without interruption.

Finally, the function is either supported or unsupported on each of the target system types:

**Windows**

The full InduSoft Web Studio software running as a project runtime server on Windows or Windows Server.

**Embedded**

Either EmbeddedView running on Windows Embedded Standard or CEView running on Windows Embedded Compact.

**Thin Client**

Either Secure Viewer (a.k.a. the InduSoft Thin Client software) or Web Thin Client (i.e., the ISymbol plug-in for Internet Explorer).

**Mobile Access**

Project screens viewed in the browser via the Mobile Access web interface.

For more information about these system types, see About target platforms and product types on page 91.

**Syntax diagram and parameters**

A basic syntax diagram shows how the function should be entered and what parameters it takes.

In most cases, a parameter can take either a literal value or the name of a project tag that contains the value. The data type of the parameter is indicated by its prefix:

**bool**

The parameter can take either a literal Boolean value or the name of a Boolean tag. For example, either 0 or MyBoolTag.

**num**

The parameter can take either a literal numerical value or the name of an Integer or Real tag. For example, either 45.6543 or MyNumTag.
str

The parameter can take either a text string enclosed in quotation marks or the name of a String tag. For example, either "My string" or MyStrTag.

The additional prefix opt indicates that a parameter is optional. If you do not specify a value for the parameter, the function will take the default value mentioned in the parameter description.

In the few cases where a parameter must take a project tag or some other special input, it will be fully explained in the parameter description.

**Returned value**

This section describes the value returned by the function, if any.

Some functions return a calculated value, depending on the nature of the function.

Other functions return an error code that indicates how well the function was executed. The possible codes and their meanings are provided in a table.

**Notes**

This section describes any additional notes or cautions on the use of the function.

**Examples**

This section shows how the function can be called in your project. Multiple examples are provided to show how the function can take both literal values and project tags, as well as how the function may be called if it has optional parameters.
Log Message functions

These functions are used to display status and debug messages in the Output window (for local runtime) or Remote LogWin window (for remote runtime).

**Trace**

*Trace* is a built-in scripting function that displays a text message in the Output window. It is typically used to debug the project.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace</td>
<td>Log Message</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

**Syntax**

*Trace*(strOutputMessage)

*strOutputMessage*

The text of the message to be displayed.

**Returned value**

This function does not return any value.

**Notes**

On Mobile Access, trace messages are displayed in the activity log in the browser console. For more information, see Use the activity log to troubleshoot a project screen on page 597.

**Examples**

Display static text that reports a specific event:

*Trace*("Beginning step 5.")

Display a date or time stamp by referencing the appropriate system tag:

*Trace*(Date)

Concatenate static text, tag references, and function calls to form a complex message:

Trace("The current second of the minute is " + Second + " and the system tick is " + GetTickCount() + " ms.")
### Arithmetic functions

These functions are used to perform advanced arithmetic operations and bit manipulation on numeric values.

#### Abs

Abs is a built-in scripting function that gets the absolute value of a specified numeric value.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

Abs(\textit{numValue})

\textit{numValue}

The numeric value from which the function takes the absolute value.

**Returned value**

The absolute value of the specified numeric value.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ABS( &quot;-54.9788&quot; ) // Returned value = 54.9788</td>
</tr>
<tr>
<td>Tag</td>
<td>ABS( \textit{numValue} ) // Returned value = absolute value of the number in the \textit{numValue} tag.</td>
</tr>
</tbody>
</table>

#### Div

This function returns the dividend of two whole numbers.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Div</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

Div(\textit{numNumerator, numDenominator})

\textit{numNumerator}

The numerator of the division operation. Please note that if you specify a decimal value, then it will be truncated.

\textit{numDenominator}

The denominator of the division operation. Please note that if you specify a decimal value, then it will be truncated.

**Returned value**

This function returns the dividend only as a whole number. The remainder is omitted.

⚠️ **Tip:** To get the remainder instead of the dividend, use the function \textit{Mod}. 
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numValue</td>
<td><code>Div( 100, 8 )</code> // Returns the value 12</td>
<td></td>
</tr>
<tr>
<td>numValue</td>
<td><code>Div( 16, 4 )</code> // Returns the value 4</td>
<td></td>
</tr>
<tr>
<td>numValue</td>
<td><code>Div( 100, 2.5 )</code> // Returns the value 50</td>
<td></td>
</tr>
</tbody>
</table>

**Format**

Format is a built-in scripting function that formats a numerical value and returns it as a string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`Format(strFlag, numValue, optStrDecimalMark, optStrThousandSep)`

`Format(strFlag, numValue| , optStrDecimalMark| | , optStrThousandSep) }}`

`strFlag` is a description of how the given numerical value should be formatted, according to the syntax `width.precisionFormat`, where:

- `width` is the minimum number of characters to be returned by the function. If the value to be returned is shorter than this, then it is padded with either blank spaces (" ") or zeroes ("0"); see "Examples" below. The value is not truncated even if the result is longer than the specified width. Applicable for formats `d, x, X, o, b, f, e, E, g, G, s, c` and `h`.
- `precision` is the number of decimal places for a floating-point number. Applicable for formats `f, e, E, g` and `G`.
- `Format` is the specific format:
Alternatively, the format can be set using the syntax `##.###`, where the numerical value is rounded to the specified number of decimal places.

**numValue**
The numerical value to be formatted.

**optStrDecimalMark**
The character used as the decimal mark, which separates the integer and fractional parts of the numerical value.

This is an optional parameter; if no value is specified, then the default is a period (.) For example: "123.45"

**optStrThousandSep**
The character used as the thousands separator, which separates the *hundreds* and *thousands* digits of the numerical value.

This is an optional parameter; if no value is specified, then the default is a comma (,). For example: "12,345"

** Returned value**
This function returns a string that contains the formatted numerical value. See "Examples" below.

**Notes**
*Format* is similar to the *printf* function in other programming languages, and it allows most of the same formatting options. However, unlike *printf*, *Format* can be used to format only one numerical value at a time.

This function is particularly useful for formatting values to be printed in reports.

**Examples**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format(&quot;%d&quot;,12.34)</td>
<td>12</td>
</tr>
<tr>
<td>Format(&quot;%04d&quot;,12.34)</td>
<td>0012</td>
</tr>
<tr>
<td>Format(&quot;%4d&quot;,12.34)</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format(&quot;%x&quot;,26)</td>
<td>1a</td>
</tr>
<tr>
<td>Format(&quot;%04x&quot;,26)</td>
<td>001a</td>
</tr>
<tr>
<td>Format(&quot;%4x&quot;,26)</td>
<td>1a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format(&quot;%X&quot;,26)</td>
<td>1A</td>
</tr>
<tr>
<td>Format(&quot;%04X&quot;,26)</td>
<td>001A</td>
</tr>
<tr>
<td>Format(&quot;%4X&quot;,26)</td>
<td>1A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format(&quot;%o&quot;,16)</td>
<td>20</td>
</tr>
<tr>
<td>Format(&quot;%04o&quot;,16)</td>
<td>0020</td>
</tr>
<tr>
<td>Format(&quot;%4o&quot;,16)</td>
<td>20</td>
</tr>
<tr>
<td>Expression</td>
<td>Returned value</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Format(&quot;%b&quot;,2)</td>
<td>10</td>
</tr>
<tr>
<td>Format(&quot;%4b&quot;,2)</td>
<td>0010</td>
</tr>
<tr>
<td>Format(&quot;%04b&quot;,2)</td>
<td>0010</td>
</tr>
<tr>
<td>Format(&quot;%0.1f&quot;,12.34)</td>
<td>12.3</td>
</tr>
<tr>
<td>Format(&quot;%06.1f&quot;,12.34)</td>
<td>0012.3</td>
</tr>
<tr>
<td>Format(&quot;%6.1f&quot;,12.34)</td>
<td>12.3</td>
</tr>
<tr>
<td>Format(&quot;%e&quot;,12.34)</td>
<td>1.234000e+001</td>
</tr>
<tr>
<td>Format(&quot;%0.1e&quot;,12.34)</td>
<td>1.2e+001</td>
</tr>
<tr>
<td>Format(&quot;%09.1e&quot;,12.34)</td>
<td>01.2e+001</td>
</tr>
<tr>
<td>Format(&quot;%9.1e&quot;,12.34)</td>
<td>1.2e+001</td>
</tr>
<tr>
<td>Format(&quot;%0.1g&quot;,12.34)</td>
<td>1e+001</td>
</tr>
<tr>
<td>Format(&quot;%0.2g&quot;,12.34)</td>
<td>12</td>
</tr>
<tr>
<td>Format(&quot;%0.3g&quot;,12.34)</td>
<td>12.3</td>
</tr>
<tr>
<td>Format(&quot;%05.3g&quot;,12.34)</td>
<td>012.3</td>
</tr>
<tr>
<td>Format(&quot;%5.3g&quot;,12.34)</td>
<td>12.3</td>
</tr>
<tr>
<td>Format(&quot;%0.1G&quot;,12.34)</td>
<td>1E+001</td>
</tr>
<tr>
<td>Format(&quot;%0.2G&quot;,12.34)</td>
<td>12</td>
</tr>
<tr>
<td>Format(&quot;%0.3G&quot;,12.34)</td>
<td>12.3</td>
</tr>
<tr>
<td>Format(&quot;%05.3G&quot;,12.34)</td>
<td>012.3</td>
</tr>
<tr>
<td>Format(&quot;%5.3G&quot;,12.34)</td>
<td>12.3</td>
</tr>
<tr>
<td>Format(&quot;%s&quot;,12.34)</td>
<td>12</td>
</tr>
<tr>
<td>Format(&quot;%04s&quot;,12.34)</td>
<td>0012</td>
</tr>
<tr>
<td>Format(&quot;%4s&quot;,12.34)</td>
<td>12</td>
</tr>
<tr>
<td>Format(&quot;%c&quot;,97)</td>
<td>a</td>
</tr>
</tbody>
</table>
### GetBit

GetBit is a built-in scripting function that gets the value of a single bit in a numeric value.

#### Syntax

`GetBit( tagName, numBitNumber )`

- **tagName**: The name of an Integer tag from which the bit value will be gotten.
  - **Note**: To directly specify the name of a tag, rather than take the value of the tag, you must enclose the tag name in double-quotes. For example, `GetBit( "Second", 1 ).`

- **numBitNumber**: A numeric tag or value specifying the position (0…31) of the bit to get.

#### Returned value

Returns the value (0 or 1) of the specified bit.

#### Notes

You also can use the Bit field to read/write values from specific bits in an integer tag. For example, enter `Second->b0` to access the LSB (Least Significant Bit of the Second tag), and `Second->b31` to access the MSB (Most Significant Bit of the Second tag).

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>GetBit( &quot;numSource&quot;, 4 )</code> // If the tag <code>numSource</code> holds a value of 15, then this function returns 0.</td>
</tr>
<tr>
<td>Tag</td>
<td><code>GetBit( &quot;numSource&quot;, 1 )</code> // If the tag <code>numSource</code> holds a value of 19, then this function returns 1.</td>
</tr>
</tbody>
</table>
**Mod**

Mod is a built-in scripting function that gets the remainder from a division operation.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mod</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`Mod( numNumerator, numDenominator )`

**numNumerator**

Integer or Real tag containing the Numerator of the function.

**numDenominator**

Integer or Real tag containing the Denominator of the function.

**Returned value**

Returns the remainder (as a real number) after dividing `numNumerator` by `numDenominator`.

**Tip:** Use the `Div` function to get the whole number dividend of the operation.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>Mod( 50, 4 )</code> // Returned value = 2</td>
</tr>
<tr>
<td>Tag</td>
<td><code>Mod( 16, 4 )</code> // Returned value = 0</td>
</tr>
<tr>
<td>Tag</td>
<td><code>Mod( 100, 8.2 )</code> // Returned value = 1.600</td>
</tr>
</tbody>
</table>

**Pow**

Pow is a built-in scripting function that gets the result of raising a numeric value to a specified exponent.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pow</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`Pow( numBase, numExponent )`

**numBase**

Integer or Real tag containing the Base of the function.

**numExponent**

Integer or real tag containing the Exponent of the function.

**Returned value**

Returns the result of raising the base to the exponent.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>Pow( 2, 3 )</code> // Returned value = 8</td>
</tr>
<tr>
<td>Tag</td>
<td><code>Pow( 10, 4 )</code> // Returned value = 10000</td>
</tr>
</tbody>
</table>

**ResetBit**

Resets a single bit in an Integer tag to 0.

Syntax

```plaintext
ResetBit( tagName, numBitNumber )
```

tagName

The name of an Integer tag where the bit value will be reset.

```plaintext
Note: To directly specify the name of a tag, rather than take the value of the tag, you must enclose the tag name in double-quotes. For example, `ResetBit( "Second", 1 )`.
```

numBitNumber

A numeric tag or value specifying the position (0…31) of the bit to reset.

Returned value

<table>
<thead>
<tr>
<th>0</th>
<th>No error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>2</td>
<td>Tag does not exist</td>
</tr>
</tbody>
</table>

Notes

You can use the Bit field to read/write values from specific bits in an integer tag. For example, enter `Second->b0` to access the LSB (Least Significant Bit of the Second tag), and `Second->b31` to access the MSB (Most Significant Bit of the Second tag).

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ResetBit( &quot;numSource&quot;, 4 )</code> // If the tag <code>numSource</code> held a value of 16, then the function returns 0 and <code>numSource</code> holds a new value of 0.</td>
</tr>
<tr>
<td>Tag</td>
<td><code>ResetBit( &quot;numSource&quot;, 1 )</code> // If the tag <code>numSource</code> held the value 19, then the function returns 0 and <code>numSource</code> holds a new value of 17.</td>
</tr>
</tbody>
</table>

**Round**

Rounds `numValue` to the nearest integer.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

Syntax

Round( numValue )

numValue
A Real tag that holds the value to be rounded.

Returned value
Returns the integer result of the round function.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Round( &quot;345.87&quot; ) // Returned value = 346</td>
</tr>
<tr>
<td>Tag</td>
<td>Round( &quot;65.323&quot; ) // Returned value = 65</td>
</tr>
</tbody>
</table>

SetBit
Sets a single bit in an Integer tag to 1.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetBit</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

SetBit( tagName, numBitNumber )

tagName
The name of an Integer tag where the bit value will be set.

Note: To directly specify the name of a tag, rather than take the value of the tag, you must enclose the tag name in double-quotes. For example, SetBit( "Second", 1 ).

numBitNumber
A numeric tag or value specifying the position (0…31) of the bit to set.

Returned value

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>2</td>
<td>Tag does not exist</td>
</tr>
</tbody>
</table>

Notes
You can also use the Bit field to read/write values from specific bits of an integer tag. For example, enter Second->b0 to access the LSB (Least Significant Bit of the Second tag), and Second->b31 to access the MSB (Most Significant Bit of the Second tag).
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetBit( &quot;numSource&quot;, 4 ) // If the tag numSource held a value of 0, then this function returns 0 and numSource holds a new value of 16.</td>
</tr>
<tr>
<td>Tag</td>
<td>GetBit( &quot;numSource&quot;, 1 ) // If the tag numSource held the value 17, then this function returns 0 and numSource holds a new value of 19.</td>
</tr>
</tbody>
</table>

**Sqrt**

Takes the square root of `numValue`.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sqrt</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
Sqrt( numValue )
```

**numValue**

Integer or Real tag to be square rooted.

**Returned value**

Returns the square root of the value in the `numValue` tag.

⚠️ **Note:** If `numValue` has a negative value, then this function returns the value 0 and sets the quality of the returned tag to BAD.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SQRT( 25 ) // Returns the value 5</td>
</tr>
<tr>
<td>Tag</td>
<td>SQRT( 67 ) // Returns the value 8.185353</td>
</tr>
</tbody>
</table>

**Swap16**

Swaps the two lower bytes of a tag.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap16</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
Swap16( numValue )
```

**numValue**

Integer tag that holds the numeric value of the bytes to be swapped.

**Returned value**

Returns the numeric value after swapping the bytes.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>( \text{Swap16( 16 )} ) // 16 = 0000000000010000 in binary. Returned value = 4096 = 0001000000000000 in binary.</td>
</tr>
<tr>
<td>Tag</td>
<td>( \text{Swap16( 43760 )} ) // 43760 = 1010010111110000 in binary. Returned value = 61610 = 1111000010100101 in binary.</td>
</tr>
</tbody>
</table>

**Swap32**
Swaps two words in a tag.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap32</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

\( \text{Swap32( numValue )} \)

**numValue**
Integer tag that holds the numeric value of the words to be swapped.

**Returned value**
Returns the numeric value after swapping the words.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>( \text{Swap32( 16 )} ) // 16 = 00000000000000000000000000010000 in binary. Returned value = 1048576 = 00000000000100000000000000000000 in binary.</td>
</tr>
<tr>
<td>Tag</td>
<td>( \text{Swap32( 246333120 )} ) // 246333120 = 1010101010101010101111111100000000 in binary. Returned value = -1094705864 = 1111111100000000101010101010 in binary.</td>
</tr>
</tbody>
</table>

**Trunc**
Truncates the value of \( \text{numValue} \).

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trunc</td>
<td>Arithmetic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

\( \text{Trunc( numValue )} \)

**numValue**
Real tag to be truncated.

**Returned value**
Returns the integer portion of the real number value of \( \text{numValue} \).

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>( \text{Trunc( 234.987 )} ) // Returned value = 234</td>
</tr>
<tr>
<td>Tag Name</td>
<td>Expression</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
</tr>
<tr>
<td>Trunc( -3465.9 )</td>
<td>// Returned value = -3465</td>
</tr>
</tbody>
</table>
Statistical functions

These functions are used to get certain statistics — such as average, maximum, and minimum — from two or more numeric values.

**Avg**

Calculates the average value of a set of numbers.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>Statistical</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`Avg( numValue1, numValue2, ... , numValueN )`

`Avg( "tagArray", numSample, optNumIgnore )`

**Note:** This function has two formats:

- If the first parameter is a numeric tag or value, you must use the `Avg( numValue1, numValue2, ... , numValueN )` format.
- If the first parameter is an array tag in double-quotes or a string tag, you must use the `Avg( "tagArray", numSample, optNumIgnore )` format.

**numValue (1...N)**

Integer or Real tags containing the numbers to be averaged together.

**tagArray**

Name of array tag (Real or Integer) containing the values to be averaged.

**numSample**

The number of array elements to be averaged.

**optNumIgnore**

Optional Integer or Real tag containing the value to be ignored in calculating the average.

**Returned value**

Returns the average of the values.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>Avg( 1, 2.34, 5, 7, 4, 8, 9.4 )</code> // Returned value = 5.248571</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td><code>Avg( 1, 5, -9, 0, 5, 3 )</code> // Returned value = 0.833333</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
</table>
Max

Returns the maximum value of a set of numbers.

Syntax

Max( numValue1, numValue2, ... , numValueN )
Max( "tagArray", numSample, optNumIgnore )

Note: This function has two formats:

- If the first parameter is a numeric tag or value, you must use the Max( numValue1, numValue2, ... , numValueN ) format.
- If the first parameter is an array tag in double-quotes or a string tag, you must use the Max( "tagArray", numSample, optNumIgnore ) format.

numValue (1...N)
Integer or Real tags containing the numbers to be analyzed.

tagArray
Name of array tag (Real or Integer) containing the values to be analyzed.

numSample
The number of array elements to be analyzed.

optNumIgnore
Integer or Real tags containing the value to be ignored in the analysis.

Returned value
Returns the maximum value of the set.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Max( 1, 2.34, 5, 7, 4, 8, 9.4 ) // Returned value = 9.4</td>
</tr>
<tr>
<td>Tag</td>
<td>Max( 1, 5, -9, 0, 5, 3 ) // Returned value = 5</td>
</tr>
</tbody>
</table>

Min

Returns the minimum value of a set of numbers.

Syntax

Min( numValue1, numValue2, ... , numValueN )
Min( "tagArray", numSample, optNumIgnore )
Note: This function has two formats:

- If the first parameter is a numeric tag or value, you must use the \texttt{Min( numValue1, numValue2, \ldots, numValueN) } format.
- If the first parameter is an array tag in double-quotes or a string tag, you must use the \texttt{Min( "tagArray", numSample, optNumIgnore ) } format.

\textbf{numValue (1...N)}

Integer or Real tags containing the numbers to be analyzed.

\textbf{tagArray}

Name of an array tag (Real or Integer) containing the values to be analyzed.

\textbf{numSample}

The number of array elements to be analyzed.

\textbf{optNumIgnore}

Integer or Real tags containing a value to be ignored in the analysis.

\textbf{Returned value}

Returns the minimum value of the set.

\textbf{Examples}

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>\texttt{Min( 1, 2.34, 5, 7, 4, 8, 9.4 )} // Returned value = 1</td>
</tr>
<tr>
<td>Tag</td>
<td>\texttt{Min( 1, 5, -9, 0, 5, 3 )} // Returned value = -9</td>
</tr>
</tbody>
</table>

\textbf{Rand}

Generates a random number between 0 and 1.

\begin{center}
<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand</td>
<td>Statistical</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>
\end{center}

\textbf{Syntax}

\texttt{Rand()}

This function has no parameters.

\textbf{Returned value}

Returns a real number between 0 and 1.

\textbf{Examples}

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>\texttt{Rand()} // Returned value = ?, Where: 0&lt;=?&lt;1</td>
</tr>
</tbody>
</table>
Logarithmic functions

These functions are used to perform logarithmic operations on numeric values.

**Exp**
Calculates the value of $e^{(2.718282)}$ raised to the power of $numValue$.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp</td>
<td>Logarithmic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**
Exp( *numValue* )

**numValue**
Integer or Real tag containing the exponent of $e$.

**Returned value**
Returns the value of $e^{( numValue )}$.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Exp( 1 )</td>
<td>// Returned value = 2.718282</td>
</tr>
<tr>
<td>Tag</td>
<td>Exp( 5.25896 )</td>
<td>// Returned value = 192.281415</td>
</tr>
</tbody>
</table>

**Log**
Calculates the natural log of $numValue$.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log</td>
<td>Logarithmic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**
Log( *numValue* )

**numValue**
Integer or Real tag from which the natural log is taken.

**Returned value**
Returns the value of $\ln( numValue )$.

**Note:** If *numValue* has a negative value, then this function will return the value 0 and it will set the quality of the returned tag to BAD.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Log( 2.718282 )</td>
<td>// Returned value = 1</td>
</tr>
<tr>
<td>Tag</td>
<td>Log( 100 )</td>
<td>// Returned value = 4.605170</td>
</tr>
</tbody>
</table>
**Log10**

Calculates the log base 10 of `numValue`.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log10</td>
<td>Logarithmic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

\[
\text{Log10}(\text{numValue})
\]

**numValue**

Integer or Real tag, from which the log base 10 is taken.

**Returned value**

Returns the value of \( \log_{10}(\text{numValue}) \).

*Note:* If `numValue` has a negative value, then this function will return the value 0 and it will set the quality of the returned tag to BAD.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>( \text{Log10}(1000) ) // Returned value = 3</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>( \text{Log10}(43.05) ) // Returned value = 1.633973</td>
<td></td>
</tr>
</tbody>
</table>
Logical functions

These functions are used to perform logical operations (e.g., if/then, true/false) on tags and expressions.

**False**

Determines whether the specified tag or expression is logically false.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>Logical</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

False(*TagOrExpression*)

*TagOrExpression*

Tag or expression to be used in the function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If the tag or expression is not logically false.</td>
</tr>
<tr>
<td>1</td>
<td>If the tag or expression is logically false.</td>
</tr>
</tbody>
</table>

**Tip:** You may find this function useful if you need to return an actual value of 0 when the expression returns some value other than 0.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>False( 1 ) // Returned value = 0</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>False( 5 &lt; 2 ) // Returned value = 1</td>
<td></td>
</tr>
</tbody>
</table>

**If**

The function If evaluates a specified tag/expression to determine whether it is logically true or false, and then it returns a corresponding value.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>If</td>
<td>Logical</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

If(*numExpression*, *numThen*, *optNumElse*)

If(*numExpression*, *numThen*{ | , *optNumElse*})

*numExpression*

The tag or expression to be evaluated.

*numThen*

The tag, expression, or value that is returned if *numExpression* is logically true (i.e., not 0).

*optNumElse*

The tag, expression, or value that is returned if *numExpression* is logically false (i.e., 0).
This parameter is optional; see "Returned value" below.

**Returned value**
This function returns either `numThen` or `optNumElse`, depending on how `numExpression` is evaluated. However, if `numExpression` is logically false and `optNumElse` is not specified, this function returns no value.

**Notes**
`numExpression` can be a combination of logic statements (e.g., **AND**, **OR**, **NOT**). For example:

```plaintext
If(TagA>TagB AND TagA=10,1,0)
```

`numThen` can be another function, including the function `If`. Therefore, you can use the functions in cascade. For example:

```plaintext
If(TagA>TagB,If(TagA<TagC,1,2),3)
```

**Examples**
Evaluate the expression "5 > 4", and then return the corresponding value (i.e., return 10, because the expression is logically true):

```plaintext
If(5>4,10,6)
```

Evaluate the expression "5 < 2", and then return the corresponding value (i.e., return 2, because the expression is logically false):

```plaintext
If(5<2,0,2)
```

Evaluate the expression "3 = 9", and then return the corresponding value (i.e., return no value, because the expression is logically false and `optNumElse` is not specified):

```plaintext
If(3=9,67)
```

**Toggle**
Returns the toggled value from the contents of `numValue` tag.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toggle</td>
<td>Logical</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
Toggle( numValue )
```

**numValue**
Boolean tag containing the value to be toggled.

**Returned value**
Numerical result (0 or 1) of the value to be toggled.

**Notes**
This function does not actually change the value of the tag, but it can be used in a command or operation that does.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Toggle( MyBoolTag ) // Returned value = 1 if MyBoolTag value equals 0, or 0 if MyBoolTag value equals 1</td>
</tr>
<tr>
<td>Tag</td>
<td>Toggle( numValue ) // Returned value = toggled value of the number in the numValue tag</td>
</tr>
</tbody>
</table>

**True**

Determines whether the specified tag or expression is logically true.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Logical</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

True( **TagOrExpression** )

**TagOrExpression**

Tag or expression to be used in the function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If the tag or expression is not logically true.</td>
</tr>
<tr>
<td>1</td>
<td>If the tag or expression is logically true.</td>
</tr>
</tbody>
</table>

**Tip:** You may find this function useful if you need to return an actual value of 1 when the expression returns some value other than 0.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>True( 1 )  // Returned value = 1</td>
</tr>
<tr>
<td>Tag</td>
<td>True( 5 &lt; 2 ) // Returned value = 0</td>
</tr>
</tbody>
</table>
String functions

These functions are used to manipulate text strings or convert them into numeric values.

**Asc2Str**

This function converts one or more Unicode character codes to a string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asc2Str</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`Asc2Str( numChar1, numChar2, ... , numCharN )`

**numChar (1-N)**

A Unicode character code (in decimal).

**Returned value**

Returns a string comprising the converted codes.

**Notes**

Although the name of this function implies it only supports ASCII characters, it is in fact a legacy of previous versions of the software. The current version supports the full Unicode character set.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Asc2Str( 65 ) // Returned value = &quot;A&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>Asc2Str( 65, 66, 67 ) // Returned value = &quot;ABC&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>Asc2Str( Array[0], Array[1], Array[2] ) // Returned value = &quot;ABC&quot;</td>
</tr>
</tbody>
</table>

**CharToValue**

This function converts a string to Unicode character codes and then stores those values in an integer array.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharToValue</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`CharToValue("tagString", "tagArray")`

**tagString**

The name of the string tag, whose value will be converted.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**tagArray**

The name of the integer array that will receive the converted values. If no array index is specified, then the default is 0.
Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

Returned value
Returns the number of array elements used, which should be equal to the number of characters in the string.

Examples
If StrTag = "ABC", then Array[0] = 65, Array[1] = 66, and Array[2] = 67:

CharToValue( "StrTag", "Array" )


CharToValue( "StrTag", "Array[10]" )

CharToValueW
This function converts a string to Unicode character codes, combines each two codes into a double-byte word, and then stores those values in an integer array.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharToValueW</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax
CharToValueW("tagString", "tagArray")
tagString
The name of the string tag, whose value will be converted.

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.
tagArray
The name of the integer array that will receive the converted values. If no array index is specified, then the default is 0.

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

Returned value
Returns the number of array elements used, which should be equal to half the number of characters in the string.

Notes
Because of how each two character codes are combined into single value, this function only supports Unicode character codes 0 through 255. For character codes greater than 255, or when double-byte words are not needed, use the CharToValue function.
**Examples**

If StrTag = "Studio", then Array[0] = 29779 ("St"), Array[1] = 25717 ("ud"), and Array[2] = 28521 ("io"):

CharToValue( "StrTag", "Array" )


CharToValue( "StrTag", "Array[10]" )

**ClassMembersToStrVector**

Transfers values from a Class tag to an Array tag.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassMembersToStrVector</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

ClassMembersToStrVector( "strClassTag" , numStartPos , numNumPos , "strArrayTag" , optBooStartPosTarget )

**strClassTag**
String value containing the Class tag name.

**numStartPos**
Start position (array index) of strClassTag.

**numNumPos**
Number of positions (array indexes) to be transferred from strClassTag.

**strArrayTag**
String value containing the array tag that will receive the values from strClassTag.

**optBooStartPosTarget**
Start position (array index) of strArrayTag. If omitted, the default value 1 is used.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-6</td>
<td>Array size of strClassTag is not big enough for numStartPos</td>
</tr>
<tr>
<td>-5</td>
<td>strClassTag is not a Class tag</td>
</tr>
<tr>
<td>-4</td>
<td>strClassTag is not found</td>
</tr>
<tr>
<td>-3</td>
<td>strArrayTag is not found</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid data type of the parameters</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>0</td>
<td>Transferred successfully</td>
</tr>
</tbody>
</table>

**Notes**

If strClassTag has more than one member, the value of each member will be transferred to strArrayTag. Therefore, it is importatnt to make sure that the array size of strArrayTag is big enough to receive all values from strClassTag.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ClassMembersToStrVector( &quot;Classtag&quot;, 5, 3, &quot;Arraytag&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>ClassMembersToStrVector( &quot;Classtag&quot;, 5, 3, &quot;Arraytag&quot;, 0 )</td>
</tr>
<tr>
<td>Tag</td>
<td>ClassMembersToStrVector( TagName, 0, 1, ArrayName )</td>
</tr>
</tbody>
</table>

**NCopy**

Copies a defined section of a larger string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCopy</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

\[ \text{NCopy}( \text{strSource}, \text{numStartChar}, \text{numQtdChar} ) \]

**strSource**

The source string.

**numStartChar**

Integer tag containing a number corresponding to the first character being copied.

**numQtdChar**

The number of characters to be copied.

**Returned value**

Returns a string that is part of the source string (as defined by the function).

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>NCopy( &quot;Studio version 7.0&quot;, 7, 7 ) // Returned value = &quot;version&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>NCopy( &quot;Technical Reference&quot;, 0, 9 ) // Returned value = &quot;Technical&quot;</td>
</tr>
</tbody>
</table>

**Note:** The first character in the string will be assigned the value 0.

**Num**

Converts a string into a float.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

\[ \text{Num}( \text{strValue} ) \]

**strValue**

The number of characters to be converted into float format.
Returned value
Returns the number (formerly in a string format) in float format.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Num( &quot;321654.987&quot; ) // Returned value = 321654.987</td>
</tr>
<tr>
<td>Tag</td>
<td>Num( &quot;5.6589626246&quot; ) // Returned value = 5.6589626246</td>
</tr>
</tbody>
</table>

Note: The float string cannot use characters other than the numbers (0..9) and a decimal point (.), or the function returns the value 0.0.

Str
Converts a number into a string.

Syntax

Str( numValue )

numValue

Integer or float tag containing a number to be converted to a string.

Returned value
Returns the string, in a float format.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Str( 321654.987 ) // Returned value = &quot;321654.987&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>Str( 5.6589626246 ) // Returned value = &quot;5.658962&quot;</td>
</tr>
</tbody>
</table>

Str2Asc
This function converts a character to its corresponding Unicode character code.

Syntax

Str2Asc( strChar )

strChar

The character to be converted.

Returned value
Returns the Unicode character code (in decimal) for the specified character.
Notes
Although the name of this function implies it only supports ASCII characters, it is in fact a legacy of previous versions of the software. The current version supports the full Unicode character set.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>Str2Asc( &quot;C&quot; )</code> // Returned value = 67</td>
</tr>
<tr>
<td>Tag</td>
<td><code>Str2Asc( &quot;o&quot; )</code> // Returned value = 111</td>
</tr>
</tbody>
</table>

StrCompare

Compares two strings to see if they are identical.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrCompare</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Syntax

`StrCompare( strValue1, strValue2 )`

**strValue1**
A string, or a tag of String type. This is the first string in the comparison.

**strValue2**
A string, or a tag of String type. This is the second string in the comparison.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>The value of <code>strValue1</code> is less than the value of <code>strValue2</code>.</td>
</tr>
<tr>
<td>0</td>
<td><code>strValue1</code> and <code>strValue2</code> are identical.</td>
</tr>
<tr>
<td>1</td>
<td>The value of <code>strValue1</code> is greater than the value of <code>strValue2</code>.</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>StrCompare( &quot;Text1&quot;, &quot;Text2&quot; )</code> // Returned value = -1</td>
</tr>
<tr>
<td>Tag</td>
<td><code>StrCompare( Tag1, Tag2 )</code> // Returned value = 0</td>
</tr>
<tr>
<td>Tag1 = &quot;Text&quot; Tag2 = &quot;Text&quot;</td>
<td><code>StrCompare( Tag1, Tag2 )</code> // Returned value = 0</td>
</tr>
<tr>
<td>Tag Tag1 = &quot;Text1&quot; Tag2 = &quot;Text2&quot;</td>
<td><code>StrCompare( Tag1, Tag2 )</code> // Returned value = -1</td>
</tr>
</tbody>
</table>

StrCompareNoCase

Compares two strings to see if they are identical, ignoring the case of letters (i.e., the lower-case "a" is considered to have the same value as the upper-case "A").

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrCompareNoCase</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>
Syntax

`StrCompareNoCase( strValue1, strValue2 )`

**strValue1**
A string, or a tag of `String` type. This is the first string in the comparison.

**strValue2**
A string, or a tag of `String` type. This is the second string in the comparison.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>The value of <code>strValue1</code> is less than the value of <code>strValue2</code>.</td>
</tr>
<tr>
<td>0</td>
<td><code>strValue1</code> and <code>strValue2</code> are identical.</td>
</tr>
<tr>
<td>1</td>
<td>The value of <code>strValue1</code> is greater than the value of <code>strValue2</code>.</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>StrCompareNoCase( &quot;Text1&quot;, &quot;TEXT1&quot; )</code> // Returned value = 0</td>
</tr>
<tr>
<td>Tag Tag1 = &quot;Text1&quot; Tag2 = &quot;TEXT1&quot;</td>
<td><code>StrCompareNoCase( Tag1, Tag2 )</code> // Returned value = 0</td>
</tr>
</tbody>
</table>

**StrFromInt**

Converts an integer into its string representation in another base number system, such as binary (base-2) or octal (base-8).

**Function**

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrFromInt</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`StrFromInt( numValue, numBase )`

**numValue**

The numeric value to be converted into a string.

**numBase**

The base number system to convert into.

**Returned value**

This function returns a string representation of the given integer, in the specified base number system. The returned value can be stored in any tag of `String` type.

**Notes**

You can specify a real number instead of an integer, but only the whole part of the number will be converted. To convert the entire real number, use the `StrFromReal` function instead.

Also, if you do not need to change the base, then use the `Str` function instead.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrFromInt( 26, 2 ) // Returned value = &quot;11010&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrFromInt( 26, 8 ) // Returned value = &quot;32&quot;</td>
</tr>
</tbody>
</table>

**StrFromReal**

StrFromReal is a built-in scripting function that converts a real numerical value to a string value, in either floating-point or exponential notation.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrFromReal</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

StrFromReal(numValue, numPrecision, { strType | f | e | E })

- **numValue**
  - The numerical value to be converted.

- **numPrecision**
  - The number of decimal places to be shown in the resulting string. Please note that the value will be rounded rather than truncated.

- **strType**
  - A single-character code that specifies how the resulting string should be formatted, as described in the following table:

<table>
<thead>
<tr>
<th>Value of strType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>Formatted in floating-point notation.</td>
</tr>
<tr>
<td>e</td>
<td>Formatted in exponential notation with a lower-case &quot;e&quot;.</td>
</tr>
<tr>
<td>E</td>
<td>Formatted in exponential notation with an upper-case &quot;E&quot;.</td>
</tr>
</tbody>
</table>

**Returned value**

This function returns a string representation of the given numerical value, with the specified precision and notation.

**Examples**

```
StrFromReal( 263.355, 2, "f" )
```

...returns a string value of "263.36".

```
StrFromReal( 263.355, 2, "e" )
```

...returns a string value of "2.63e+002".

```
StrFromReal( 263.355, 2, "E" )
```

...returns a string value of "2.63E+002".
**StrFromTime**

Converts a timestamp from UTC standard notation into a formatted string, adjusted to reflect the Time Zone setting in the Control Panel of the local computer.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrFromTime</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

**Syntax**

`StrFromTime( numUTCTime, numType )`

**numUTCTime**

An integer, or a tag of `Integer` type. A timestamp given in UTC standard notation.

**numType**

An integer, or a tag of `Integer` type. Specifies the format of the resulting string, as described in the following table:

<table>
<thead>
<tr>
<th>Value of numType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Displays the date in the same format that is selected in the Control Panel on the local computer.</td>
</tr>
<tr>
<td>2</td>
<td>Displays the time in the same format that is selected in the Control Panel on the local computer.</td>
</tr>
<tr>
<td>3</td>
<td>Displays a standard 24-character string that shows both date and time.</td>
</tr>
<tr>
<td>4</td>
<td>Displays the abbreviated name of the day of the week.</td>
</tr>
<tr>
<td>5</td>
<td>Displays the full name of the day of the week.</td>
</tr>
</tbody>
</table>

**Returned value**

This function returns a string representation of the given timestamp, with the specified formatting. The returned value can be stored in any tag of `String` type.

**Notes**

The Coordinated Universal Time (UTC) standard counts the number of seconds elapsed since 12:00 AM GMT on January 1, 1970. Each day consists of 86,400 seconds.

**Examples**

```plaintext
Note: The examples below are for a computer set to Eastern Standard Time (or UTC −05:00).

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrFromTime( 86400, 1 )</td>
<td>&quot;1/1/70&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrFromTime( 86400, 2 )</td>
<td>&quot;07:00:00 PM&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrFromTime( 86400, 3 )</td>
<td>&quot;Thu Jan 01 19:00:00 1970&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrFromTime( 86400, 4 )</td>
<td>&quot;Thu&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrFromTime( 86400, 5 )</td>
<td>&quot;Thursday&quot;</td>
</tr>
</tbody>
</table>
```
**StrGetElement**

Gets a specific element from a string source.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrGetElement</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

\[
\text{StrGetElement}( \text{strSource}, \text{strDelimiter}, \text{numElementNumber} )
\]

**strSource**

The source string.

**strDelimiter**

Char used as delimiter between the elements.

**numElementNumber**

Number of the element which will be returned by the function. The first element has the number 1. The second element has the number 2 and so forth.

**Returned value**

Returns the element (string value) retrieved from strSource.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrGetElement( &quot;a</td>
</tr>
<tr>
<td>Tag</td>
<td>StrGetElement( &quot;a,b,c&quot;, &quot;,&quot;, 3 ) // returned value = &quot;c&quot;</td>
</tr>
</tbody>
</table>

**StrLeft**

Copies the first characters of a larger string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrLeft</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

\[
\text{StrLeft}( \text{strSource}, \text{numQtdChar} )
\]

**strSource**

The source string.

**numQtdChar**

The number of characters to be copied.

**Returned value**

Returns a string containing the left-most characters in the source string.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrLeft( &quot;Studio version 7.0&quot;, 8 ) // Returned value = Studio v</td>
</tr>
<tr>
<td>Tag</td>
<td>StrLeft ( &quot;Technical Reference&quot;, 9 ) // Returned value = Technical</td>
</tr>
</tbody>
</table>

**StrLen**

Determines the length of a string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrLen</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

StrLen( strSource )

**strSource**

The string.

**Returned value**

Returns an integer that is the number of characters in the string.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrLen( &quot;Studio version 7.0&quot; ) // Returned value = 18</td>
</tr>
<tr>
<td>Tag</td>
<td>StrLen( &quot;Technical Reference&quot; ) // Returned value = 19</td>
</tr>
</tbody>
</table>

**StrLower**

Converts a string to all lower case characters.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrLower</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

StrLower( strSource )

**strSource**

The string to be converted.

**Returned value**

Returns the string, where all the characters are in lowercase.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrLower( &quot;Studio version 7.0&quot; ) // Returned value = &quot;studio version 7.0&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrLower( &quot;Technical Reference&quot; ) // Returned value = &quot;technical reference&quot;</td>
</tr>
</tbody>
</table>
**StrRChr**

Isolates the final occurrence of a character sequence within a string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrRChr</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
StrRChr( strSource, strChrSequence )
```

*strSource*

The source string.

*strCharSequence*

The reference string.

**Returned value**

Returns a string of characters following the last occurrence of a character within the source string.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrRChr( &quot;Studio version 7.0&quot;, &quot;i&quot; ) // Returned value = &quot;ion 7.0&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrRChr( &quot;Technical Reference&quot;, &quot;n&quot; ) // Returned value = &quot;nce&quot;</td>
</tr>
</tbody>
</table>

**StrRight**

Copies the last characters in a larger string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrRight</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
StrRight( strSource, numQtdChar )
```

*strSource*

The source string.

*numQtdChar*

The number of characters to be copied.

**Returned value**

Returns a string containing the right-most characters in a source string.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrRight( &quot;Studio version 7.0&quot;, 8 ) // Returned value = &quot;sion 7.0&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrRight( &quot;Technical Reference&quot;, 9 ) // Returned value = &quot;Reference&quot;</td>
</tr>
</tbody>
</table>
**StrSetElement**

Sets a specific element in a string source.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrSetElement</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
StrSetElement( strSource, strDelimiter, numElementNumber, strValue )
```

**strSource**

The source string.

**strDelimiter**

Char used as delimiter between the elements.

**numElementNumber**

Number of the element where the string value will be written by the function. The first element has the number 1. The second element has the number 2 and so forth.

**strValue**

String value that will be written to the numElementNumber of the strSource string tag.

**Returned value**

Returns the string value updated with the `strValue`.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringTag</td>
<td>StrSetElement( strSource, &quot;</td>
</tr>
<tr>
<td>StringTag</td>
<td>StrSetElement( strSource, &quot;,&quot;, 3, &quot;defg&quot; )</td>
</tr>
</tbody>
</table>

**StrStr**

Isolates the first occurrence of a character sequence within a string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrStr</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
StrStr( strSource, strCharSequence )
```

**strSource**

The source string.

**strCharSequence**

The reference string.

**Returned value**

Returns the string of characters following the first occurrence of a character within the source string.
Appendix: Built-in Scripting Language

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrStr( &quot;Studio version 7.0&quot;, &quot;i&quot; ) // Returned value = &quot;io version 7.0&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrStr( &quot;Technical Reference&quot;, &quot;n&quot; ) // Returned value = &quot;nical Reference&quot;</td>
</tr>
</tbody>
</table>

**StrStrPos**

Finds the first occurrence of a character within a string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrStrPos</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Syntax

StrStrPos( strSource, strCharSequence )

strSource

The source string.

strCharSequence

The reference string.

Returned value

Returns an integer corresponding to the first occurrence of a character within the source string.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrStrPos( &quot;Studio version 7.0&quot;, &quot;i&quot; ) // Returned value = 4</td>
</tr>
<tr>
<td>Tag</td>
<td>StrStrPos( &quot;Technical Reference&quot;, &quot;a&quot; ) // Returned value = 7</td>
</tr>
</tbody>
</table>

Note: The first character in the string assigned the value 0.

**StrTrim**

Removes unwanted spaces from a string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrTrim</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Syntax

StrTrim( strReference, optNumFlag )

strReference

A string, or a tag of String type that contains the source string.

optNumFlag

An optional integer or tag of Integer type:
**Value of optNumFlag**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Removes all spaces from both the beginning and the end of the string.</td>
</tr>
<tr>
<td>1</td>
<td>Removes all spaces only from the beginning of the string.</td>
</tr>
<tr>
<td>2</td>
<td>Removes all spaces only from the end of the string.</td>
</tr>
<tr>
<td>3</td>
<td>Removes all spaces except for single spaces between words.</td>
</tr>
</tbody>
</table>

**Note:** If no value is given for `optNumFlag`, then 0 is the default.

**Returned value**

This function returns a string equal to `strReference` minus the specified space characters. The returned value can be stored in any tag of `String` type.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>StrTrim( &quot; Studio version 7.0 &quot; ) // Returned value = &quot;Studio version 7.0&quot;</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>StrTrim( &quot; Studio version 7.0 &quot;, 0 ) // Returned value = &quot;Studio version 7.0&quot;</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>StrTrim( &quot; Studio version 7.0 &quot;, 1 ) // Returned value = &quot;Studio version 7.0 &quot;</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>StrTrim( &quot; Studio version 7.0 &quot;, 2 ) // Returned value = &quot; Studio version 7.0&quot;</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>StrTrim( &quot; Studio version 7.0 &quot;, 3 ) // Returned value = &quot;Studio version 7.0&quot;</code></td>
</tr>
</tbody>
</table>

**`StrTrimAll`**

Eliminates a specific char from the whole string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrTrimAll</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`StrTrimAll( strReference, optStrTrimChar )`

**strReference**

A The source string.

**optStrTrimChar**

Char that will be removed from the string. If this parameter is omitted, the space char will be removed from the string, by default.

**Returned value**

Returns a string equal to `strReference` minus the characters removed by the function.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>StrTrimAll( &quot;Studio version 7.0 &quot;, &quot; &quot; ) // Returned value = &quot;Studioversion7.0&quot;</code></td>
</tr>
</tbody>
</table>
### StrUpper

Converts a string to all uppercase characters.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StrUpper</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

#### Syntax

\[ \text{StrUpper}( \text{strSource} ) \]

**strSource**

The string.

#### Returned value

Returns the string with all characters are in uppercase.

#### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>StrUpper( &quot;Studio version 7.0&quot; ) // Returned value = &quot;STUDIO VERSION 6.1&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>StrUpper( &quot;Technical Reference&quot; ) // Returned value = &quot;TECHNICAL REFERENCE&quot;</td>
</tr>
</tbody>
</table>

### ValueToChar

This function converts an integer array of Unicode character codes to a string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueToChar</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

#### Syntax

\[ \text{ValueToChar}(\text{tagArray}, \text{numQtdChars}) \]

**tagArray**

The name of the integer array containing the values to be converted. If no array index is specified, then the default is 0.

**numQtdChars**

The number of values to be converted (minimum of 1), starting with the specified array index.

#### Returned value

Returns a string comprising the converted values.

#### Examples

If Array[0] = 65, Array[1] = 66, and Array[2] = 67, then the returned value will be "ABC":

\[ \text{ValueToChar}(\text{"Array"}, 3) \]
Appendix: Built-in Scripting Language


ValueToChar( "Array[10]", 3 )

**ValueWToChar**

This function converts an integer array of Unicode character codes to a string, where each value in the array is a double-byte word.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueWToChar</td>
<td>String</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ValueWToChar("tagArray", numQtdChars)`

- **tagArray**
  The name of the integer array containing the double-byte values to be converted. If no array index is specified, then the default is 0.

  Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

- **numQtdChars**
  The number of values to be converted (minimum of 1), starting with the specified array index.

**Returned value**

Returns a string comprising the converted values.

**Notes**

Each value in the array must be a double-byte word, and each word will be split into two bytes for conversion. As such, this function only supports Unicode character codes 0 through 255. For character codes greater than 255, or when double-byte words are not needed, use the ValueToChar function.

**Examples**

If Array[0] = 29779, Array[1] = 25717, and Array[2] = 28521, then the returned value is "Studio":

ValueWToChar( "Array", 3 )


ValueWToChar( "Array[10]", 3 )
Date & Time functions

These functions are used to interact with the system clock or manipulate timestamps.

**ClockGetDate**

This function calculates the date that corresponds to a specified number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClockGetDate</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

**Syntax**

ClockGetDate( numSeconds, optTimeZone )

```plaintext
ClockGetDate(numSeconds{ | , optTimeZone })
```

**numSeconds**

A number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970. This number is typically provided by the function GetClock, but it can be any number.

**optTimeZone**

A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| -i                  | A time zone index (i), up to the maximum index returned by the function GetTimeZoneCount.  
|                     | On Windows Embedded Compact, the only valid index is \(-1\) (i.e., the current time zone). |
| 0                   | The current time zone in either the project runtime or the Viewer module / thin client, depending on where the function is called from.  
|                     | This may be different from the current time zone in the local computer's system settings, if the function SetTimeZone was executed previously. |
| 1                   | Coordinated Universal Time (UTC).                                           |
| 2                   | The current time zone in the local computer's system settings.  
|                     | On Windows Embedded Compact, this value is invalid.                         |
| "timezone"          | The full name of a time zone (e.g., "Central Standard Time").  
|                     | On Windows Embedded Compact, this value is invalid.                         |

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

**Returned value**

If this function is executed successfully, it returns a string that contains the date that corresponds to the specified number of seconds. If a time zone is specified, the date is adjusted to reflect the change from the current time zone to the specified time zone. The date is formatted as MM/DD/YYYY.

If this function is not executed successfully — for example, if the specified time zone is invalid — it returns a value of 0.
Notes
If this function is called on Mobile Access, it is executed using the system time on the project runtime server.

Examples
Get the date when the system clock started counting (i.e., 01/01/1970):

ClockGetDate(0,"Greenwich Mean Time")

Get a returned value of 04/15/2002:

ClockGetDate(1018886359)

Get the current number of seconds elapsed since 12:00 AM GMT on January 1, 1970, then format that as a date, then adjust it from the current time zone to Central Standard Time:

ClockGetDate(GetClock(),"Central Standard Time")

ClockGetDayOfWeek
This function calculates the day of the week that corresponds to a specified number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClockGetDayOfWeek</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

Syntax

ClockGetDayOfWeek( numSeconds, optTimeZone )

ClockGetDayOfWeek(numSeconds{   | ,optTimeZone })

numSeconds
A number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970. This number is typically provided by the function GetClock, but it can be any number.

optTimeZone
A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>A time zone index (i), up to the maximum index returned by the function GetTimeZoneCount. On Windows Embedded Compact, the only valid index is -1 (i.e., the current time zone).</td>
</tr>
<tr>
<td>0</td>
<td>The current time zone in either the project runtime or the Viewer module / thin client, depending on where the function is called from. This may be different from the current time zone in the local computer's system settings, if the function SetTimeZone was executed previously.</td>
</tr>
<tr>
<td>1</td>
<td>Coordinated Universal Time (UTC).</td>
</tr>
<tr>
<td>2</td>
<td>The current time zone in the local computer's system settings. On Windows Embedded Compact, this value is invalid.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&quot;timezone&quot;</td>
<td>The full name of a time zone (e.g., &quot;Central Standard Time&quot;).</td>
</tr>
<tr>
<td></td>
<td>On Windows Embedded Compact, this value is invalid.</td>
</tr>
</tbody>
</table>

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

**Returned value**

If this function is executed successfully, it returns the day of the week as an integer:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sunday</td>
</tr>
<tr>
<td>1</td>
<td>Monday</td>
</tr>
<tr>
<td>2</td>
<td>Tuesday</td>
</tr>
<tr>
<td>3</td>
<td>Wednesday</td>
</tr>
<tr>
<td>4</td>
<td>Thursday</td>
</tr>
<tr>
<td>5</td>
<td>Friday</td>
</tr>
<tr>
<td>6</td>
<td>Saturday</td>
</tr>
</tbody>
</table>

If a time zone is specified, the day is adjusted to reflect the change from the current time zone to the specified time zone.

**Notes**

If this function is called on Mobile Access, it is executed using the system time on the project runtime server.

**Examples**

Get the day of the week when the system clock started counting (i.e., Wednesday):

```
ClockGetTime(0)
```

Get a returned value of 1 (Monday):

```
ClockGetDayOfWeek(1018886359)
```

Get the current number of seconds elapsed since 12:00 AM GMT on January 1, 1970, and then calculate the day of the week:

```
ClockGetDayOfWeek(GetClock())
```

**ClockGetTime**

This function calculates the time that corresponds to a specified number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970.

**Syntax**

```
ClockGetTime( numSeconds, optTimeZone )
```
ClockGetTime(numSeconds | , optTimeZone)}

numSeconds
A number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970. This number is typically provided by the function GetClock, but it can be any number.

optTimeZone
A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>A time zone index (i), up to the maximum index returned by the function GetTimeZoneCount. On Windows Embedded Compact, the only valid index is -1 (i.e., the current time zone).</td>
</tr>
<tr>
<td>0</td>
<td>The current time zone in either the project runtime or the Viewer module/thin client, depending on where the function is called from. This may be different from the current time zone in the local computer's system settings, if the function SetTimeZone was executed previously.</td>
</tr>
<tr>
<td>1</td>
<td>Coordinated Universal Time (UTC).</td>
</tr>
<tr>
<td>2</td>
<td>The current time zone in the local computer's system settings. On Windows Embedded Compact, this value is invalid.</td>
</tr>
<tr>
<td>&quot;timezone&quot;</td>
<td>The full name of a time zone (e.g., &quot;Central Standard Time&quot;). On Windows Embedded Compact, this value is invalid.</td>
</tr>
</tbody>
</table>

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

Returned value
If this function is executed successfully, it returns a string that contains the time that corresponds to the specified number of seconds. If a time zone is specified, the time is adjusted to reflect the change from the current time zone to the specified time zone. The time is formatted as HH:MM:SS.

If this function is not executed successfully — for example, if the specified time zone is invalid — it returns a value of 0.

Notes
If this function is called on Mobile Access, it is executed using the system time on the project runtime server.

Examples
Get the time when the system clock started counting (i.e., 00:00:00):

ClockGetTime(0,"Greenwich Mean Time")

Get a returned value of 10:59:19:

ClockGetTime(1018886359)

Get the current number of seconds elapsed since 12:00 AM GMT on January 1, 1970, then format that as a time, then adjust it from the current time zone to Central Standard Time:

ClockGetTime(GetClock(),"Central Standard Time")
**DateTime2Clock**

This function calculates how many seconds have elapsed since 12:00 AM GMT on January 1, 1970, given a specified date and time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateTime2Clock</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
DateTime2Clock( strDate, strTime, optTimeZone )
```

```plaintext
DateTime2Clock( strDate, strTime | , optTimeZone )
```

**strDate**

The date to be used in the calculation, formatted as **MM/DD/YYYY**.

**strTime**

The time to be used in the calculation, formatted as **HH:MM:SS**.

**optTimeZone**

A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>A time zone index (i), up to the maximum index returned by the function <code>GetTimeZoneCount</code>. On Windows Embedded Compact, the only valid index is -1 (i.e., the current time zone).</td>
</tr>
<tr>
<td>0</td>
<td>The current time zone in either the project runtime or the Viewer module / thin client, depending on where the function is called from. This may be different from the current time zone in the local computer's system settings, if the function <code>SetTimeZone</code> was executed previously.</td>
</tr>
<tr>
<td>1</td>
<td>Coordinated Universal Time (UTC).</td>
</tr>
<tr>
<td>2</td>
<td>The current time zone in the local computer's system settings. On Windows Embedded Compact, this value is invalid.</td>
</tr>
<tr>
<td>&quot;timezone&quot;</td>
<td>The full name of a time zone (e.g., &quot;Central Standard Time&quot;). On Windows Embedded Compact, this value is invalid.</td>
</tr>
</tbody>
</table>

This parameter is optional; if you do not specify a tag or value, then the default value is **0**.

**Returned value**

This function returns the number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970, given the specified date and time. If a time zone is specified, then the number of seconds is adjusted to reflect the change from the current time zone to the specified time zone.

**Notes**

If this function is called on Mobile Access, it is executed using the system time on the project runtime server.
Examples
Get the number of seconds elapsed when the system clock started counting:

    DateTime2Clock("01/01/1970", "00:00:00", "Greenwich Mean Time")

Get the number of seconds elapsed at 10:59:19 AM on April 15, 2002, in the current time zone:

    DateTime2Clock("04/15/2002", "10:59:19")

DateTime2UTC
This function converts a date and time from the specified time zone to Coordinated Universal Time (UTC).

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateTime2UTC</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

Syntax

    DateTime2UTC(strDateTime, optTimeZone)
    DateTime2UTC(strDateTime | optTimeZone)

strDateTime
The date and time to be converted, in the format "MM/DD/YYYY HH:MM:SS".

optTimeZone
A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| -1             | A time zone index (i), up to the maximum index returned by the function GetTimeZoneCount.  
|                | On Windows Embedded Compact, the only valid index is -1 (i.e., the current time zone). |
| 0              | The current time zone in either the project runtime or the Viewer module / thin client, depending on where the function is called from. The current time zone in the local computer's system settings, if the function SetTimeZone was executed previously. |
| 1              | Coordinated Universal Time (UTC).                                           |
| 2              | The current time zone in the local computer's system settings. On Windows Embedded Compact, this value is invalid. |
| "timezone"     | The full name of a time zone (e.g., "Central Standard Time"). On Windows Embedded Compact, this value is invalid. |

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

Returned value
If this function is executed successfully, it returns a string that contains the specified date and time converted from the specified time zone to UTC. Otherwise, it returns one of the following possible values:
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>Invalid time zone name or index.</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid or improperly formatted date/time.</td>
</tr>
<tr>
<td>-3</td>
<td>The specified value is not a string.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

### Notes

The list of available time zones varies by operating system version and configuration. If necessary, you can use the functions `GetTimeZoneCount` and `GetTimeZone` to generate a list that is specific to your computer.

If this function is called on Mobile Access, it is executed using the system time on the project runtime server.

### Examples

Convert the specified date and time from the current time zone to UTC:

```plaintext
DateTime2UTC("07/15/2013 19:54:46")
```

Convert the specified date and time from Central Standard Time to UTC:

```plaintext
DateTime2UTC("07/15/2013 19:54:46","Central Standard Time")
```

### GetClock

This function gets how many seconds have elapsed since 12:00 AM GMT on January 1, 1970, according to the system clock.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetClock</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

### Syntax

```plaintext
GetClock( optTimeZone )
```

```plaintext
GetClock({ | optTimeZone |})
```

**optTimeZone**

A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>A time zone index (i), up to the maximum index returned by the function <code>GetTimeZoneCount</code>. On Windows Embedded Compact, the only valid index is -1 (i.e., the current time zone).</td>
</tr>
<tr>
<td>0</td>
<td>The current time zone in either the project runtime or the Viewer module / thin client, depending on where the function is called from. This may be different from the current time zone in the local computer's system settings, if the function <code>SetTimeZone</code> was executed previously.</td>
</tr>
<tr>
<td>1</td>
<td>Coordinated Universal Time (UTC).</td>
</tr>
<tr>
<td>2</td>
<td>The current time zone in the local computer's system settings. On Windows Embedded Compact, this value is invalid.</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

Value | Description
---|---
"timezone" | The full name of a time zone (e.g., "Central Standard Time").
| On Windows Embedded Compact, this value is invalid.

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

Returned value

This function returns the number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970, according to the system clock. If a time zone is specified, the number of seconds is adjusted to reflect the change from the current time zone to the specified time zone.

Notes

If this function is called on Mobile Access, it is executed using the system time on the project runtime server.

GetTimeZone

This function gets a specified time zone name or index.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetTimeZone</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

Syntax

GetTimeZone( optTimeZone )

GetTimeZone({ | optTimeZone |})

optTimeZone

A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| -i | A time zone index (i), up to the maximum index returned by the function GetTimeZoneCount.
| | On Windows Embedded Compact, the only valid index is -1 (i.e., the current time zone).
| 0 | The current time zone in either the project runtime or the Viewer module / thin client, depending on where the function is called from.
| "timezone" | The full name of a time zone (e.g., "Central Standard Time").
| | On Windows Embedded Compact, the only valid name is the current time zone.

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

Returned value

If optTimeZone is a time zone name, this function returns the corresponding index. If optTimeZone is a time zone index, this function returns the corresponding name. Otherwise, this function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>3</td>
<td>Invalid time zone name or index.</td>
</tr>
<tr>
<td>4</td>
<td>Windows Embedded Compact only supports the current time zone.</td>
</tr>
</tbody>
</table>

Notes

The list of available time zones varies by operating system version and configuration. If necessary, you can use this function in combination with the function `GetTimeZoneCount` to generate a list that is specific to your computer.

If this function is called on Mobile Access, it is executed using the system time on the project runtime server.

Examples

Get the current time zone:

```plaintext
GetTimeZone()
```

Get the time zone name that corresponds to time zone index 24:

```plaintext
GetTimeZone(-24)
```

Get the time zone index that corresponds to Central Standard Time:

```plaintext
GetTimeZone("Central Standard Time")
```

**GetTimeZoneCount**

This function gets the number of available time zones on the local computer.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetTimeZone</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

Syntax

```plaintext
GetTimeZoneCount()
```

This function has no parameters.

Returned value

This function returns the number of available time zones on the local computer, depending on where the function is called from [i.e., the project runtime or the Viewer module / thin client].

On Windows Embedded Compact, this function always returns 1 because the operating system only supports the current time zone.

Notes

The list of available time zones varies by operating system version and configuration. If necessary, you can use this function in combination with the function `GetTimeZone` to generate a list that is specific to your computer.

If this function is called on Mobile Access, it is executed using the system time on the project runtime server.
GetUTC

This function gets the current Coordinated Universal Time (UTC) on the local computer.

Function | Group    | Execution | Windows | Embedded | Thin Client | Mobile Access
---|----------|-----------|---------|----------|-------------|---------------
GetUTC   | Date & Time | Synchronous | Supported | Supported | Supported | Supported (see "Notes" below)

Syntax

GetUTC()

This function has no parameters.

Returned value

This function returns a string that contains the current Coordinated Universal Time (UTC) on the local computer, depending on where the function is called from (i.e., the project runtime or the Viewer module / thin client). The time is formatted as **MM/DD/YYYY HH:MM:SS**.

If this function is called on Mobile Access, it is executed using the system time on the project runtime server.

Hour2Clock

Converts time in the **HH:MM:SS** format into seconds.

Function | Group    | Execution | Windows | Embedded | Thin Client | Mobile Access
---|----------|-----------|---------|----------|-------------|---------------
Hour2Clock | Date & Time | Synchronous | Supported | Supported | Supported | Supported

Syntax

Hour2Clock( **strTime** )

**strTime**

The number of hours, minutes, and seconds in **HH:MM:SS** format.

Returned value

Returns the number of seconds equivalent to the total number of hours, minutes, and seconds specified.

Examples

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Hour2Clock( &quot;01:00:00&quot; ) //</td>
<td>3600</td>
</tr>
<tr>
<td>Tag</td>
<td>Hour2Clock( &quot;10:01:01&quot; ) //</td>
<td>36061</td>
</tr>
</tbody>
</table>

SetSystemDate

Sets the date in the operating system’s clock.

Function | Group    | Execution | Windows | Embedded | Thin Client | Mobile Access
---|----------|-----------|---------|----------|-------------|---------------
SetSystemDate | Date & Time | Synchronous | Supported | Supported | Supported | Not supported
Appendix: Built-in Scripting Language

SetSystemDate (strDate)

Syntax

SetSystemDate (strDate)
strDate

The date (in MM/DD/YYYY format) in which to set the clock.

Returned value

This function does not return any value.

Notes

For this function to execute successfully and change the system settings, you must run InduSoft Web Studio with Administrator privileges. To run as an administrator once, right-click the program icon and then click Run as administrator on the shortcut menu. To always run as an administrator, open the program's Properties dialog box, click Compatibility, and then select Run this program as an administrator.

Examples

Set the system clock to 15 Apr 2002:

SetSystemDate("04/15/2002")

SetSystemTime

Sets the time in the operating system's clock.

Syntax

SetSystemTime (strTime)

SetSystemTime (strTime)
strTime

The time (in HH:MM:SS format) in which to set the clock.

Returned value

This function does not return any value.

Notes

For this function to execute successfully and change the system settings, you must run InduSoft Web Studio with Administrator privileges. To run as an administrator once, right-click the program icon and then click Run as administrator on the shortcut menu. To always run as an administrator, open the program's Properties dialog box, click Compatibility, and then select Run this program as an administrator.

Examples

Set the system clock to 3:45:18 PM:

SetSystemTime("15:45:18")
**SetTimeZone**

This function sets the time zone for the Viewer module / thin client, separate from the local computer's system settings.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetTimeZone</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot;)</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
SetTimeZone( optTimeZone )
```

```
SetTimeZone({   | optTimeZone;})
```

`optTimeZone`

A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>A time zone index (i), up to the maximum index returned by the function GetTimeZoneCount.</td>
</tr>
<tr>
<td>0</td>
<td>The current time zone in the Viewer module / thin client.</td>
</tr>
<tr>
<td>1</td>
<td>Coordinated Universal Time (UTC).</td>
</tr>
<tr>
<td>2</td>
<td>The current time zone in the local computer's system settings.</td>
</tr>
<tr>
<td>&quot;timezone&quot;</td>
<td>The full name of a time zone.</td>
</tr>
</tbody>
</table>

This parameter is optional; if you do not specify a tag or value, then the default value is 2.

**Returned value**

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>Invalid time zone name or index.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid function call; this function can only be called from the Viewer module / thin client. The project runtime will always use the local computer's system settings.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

**Notes**

The list of available time zones varies by operating system version and configuration. If necessary, you can use the functions GetTimeZoneCount and GetTimeZone to generate a list that is specific to your computer.

Also, this function is supported on Windows Embedded Standard but not on Windows Embedded Compact.

**Examples**

Reset the time zone to the local computer's system settings:

```
SetTimeZone()
```

Set the time zone to UTC:

```
SetTimeZone(1)
```
Set the time zone to index 24:

```
SetTimeZone(-24)
```

Set the time zone to Central Standard Time:

```
SetTimeZone("Central Standard Time")
```

### UTC2DateTime

This function converts a date and time from Coordinated Universal Time (UTC) to the specified time zone.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTC2DateTime</td>
<td>Date &amp; Time</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

#### Syntax

```
UTC2DateTime( strUTC, optTimeZone )
```

```
UTC2DateTime(strUTC| , optTimeZone )
```

**strUTC**

The date and time to be converted, in the format "MM/DD/YYYY HH:MM:SS".

**optTimeZone**

A time zone name (string) or index (integer).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>A time zone index (i), up to the maximum index returned by the function GetTimeZoneCount.</td>
</tr>
<tr>
<td></td>
<td>On Windows Embedded Compact, the only valid index is -1 (i.e., the current time zone).</td>
</tr>
<tr>
<td>0</td>
<td>The current time zone in either the project runtime or the Viewer module / thin client, depending on where the function is called from.</td>
</tr>
<tr>
<td></td>
<td>This may be different from the current time zone in the local computer's system settings, if the function SetTimeZone was executed previously.</td>
</tr>
<tr>
<td>1</td>
<td>Coordinated Universal Time (UTC).</td>
</tr>
<tr>
<td>2</td>
<td>The current time zone in the local computer's system settings. On Windows Embedded Compact, this value is invalid.</td>
</tr>
<tr>
<td>&quot;timezone&quot;</td>
<td>The full name of a time zone (e.g., &quot;Central Standard Time&quot;).</td>
</tr>
<tr>
<td></td>
<td>On Windows Embedded Compact, this value is invalid.</td>
</tr>
</tbody>
</table>

This parameter is optional; if you do not specify a tag or value, then the default value is **0**.

#### Returned value

If this function is executed successfully, it returns a string that contains the specified date and time converted from UTC to the specified time zone. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>Invalid time zone name or index.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid or improperly formatted date/time.</td>
</tr>
<tr>
<td>-3</td>
<td>The specified value is not a string.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

**Notes**

The list of available time zones varies by operating system version and configuration. If necessary, you can use the functions `GetTimeZoneCount` and `GetTimeZone` to generate a list that is specific to your computer.

If this function is called on Mobile Access, it is executed using the system time on the project runtime server.

**Examples**

Convert the specified date/time from UTC to the current time zone:

```plaintext
UTC2DateTime("07/15/2013 19:54:46")
```

Convert the specified date and time from UTC to Central Standard Time:

```plaintext
UTC2DateTime("07/15/2013 19:54:46","Central Standard Time")
```

Convert the current date and time from UTC to Central Standard Time:

```plaintext
UTC2DateTime(GetUTC(),"Central Standard Time")
```
Trigonometric functions

These functions are used to perform trigonometric operations (e.g., sine, cosine, tangent) on numeric values.

**ACos**
Calculates the Arc Cosine of a value.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACos</td>
<td>Trigonometric</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ACos( numValue )`

**numValue**
Numerical tag from which the Arc Cosine will be taken.

**Returned value**
Returns the Arc Cosine of numValue in radians.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>( \text{ACos}( 1 ) ) // Returned value = 0.000000</td>
</tr>
<tr>
<td>Tag</td>
<td>( \text{ACos}( 0 ) ) // Returned value = 1.570796</td>
</tr>
</tbody>
</table>

**ASin**
Calculates the Arc Sine of a value.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASin</td>
<td>Trigonometric</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ASin( numValue )`

**numValue**
Numerical tag from which the Arc Sine will be taken.

**Returned value**
Returns the Arc Sine of numValue in radians.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>( \text{ASin}( 1 ) ) // Returned value = 1.570796</td>
</tr>
<tr>
<td>Tag</td>
<td>( \text{ASin}( 0 ) ) // Returned value = 0.000000</td>
</tr>
</tbody>
</table>
**ATan**

Calculates the Arc Tangent of a value.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATan</td>
<td>Trigonometric</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
ATan( numValue )
```

**numValue**

Numerical tag from which the Arc Tangent will be taken.

**Returned value**

Returns the Arc Tangent of numValue in radians.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ATan( 1 ) // Returned value = 0.785398</td>
</tr>
<tr>
<td>Tag</td>
<td>ATan( 0 ) // Returned value = 1.570796</td>
</tr>
</tbody>
</table>

**Cos**

Calculates the Cosine of a value.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cos</td>
<td>Trigonometric</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
Cos( numAngle )
```

**numAngle**

The Angle (in radians) from which to calculate the Cosine.

**Returned value**

Returns the Cosine of numAngle.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Cos( 1.570796 ) // Returned value = 0.000000</td>
</tr>
<tr>
<td>Tag</td>
<td>Cos( 0 ) // Returned value = 1.000000</td>
</tr>
</tbody>
</table>

**Cot**

The function Cot calculates the cotangent of an angle.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tan</td>
<td>Trigonometric</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>
**Syntax**

\[\text{Cot}(\text{numAngle})\]

\[\text{Cot}(\text{numAngle})\]

**numAngle**

The angle in radians.

**Tip:** 2π radians is the same as 360 degrees.

**Returned value**

This function returns the cotangent of the specified angle.

The function cannot return the actual cotangent of π (i.e., \(\text{Cot}(\text{Pi}())\) or equivalent), because that is infinite. Instead, the function returns the largest number possible, given the limited precision of the value returned by the function \(\text{Pi}\).

**Examples**

Calculate the cotangent of 1 radian:

\[\text{Cot}(1)\]

Calculate the cotangent of π/2 radians:

\[\text{Cot}(\text{Pi}()/2)\]

Calculate the cotangent of the value of the project tag \(\text{MyAngle}\):

\[\text{Cot}(\text{MyAngle})\]

**Pi**

Calculates the value of \(\pi\).

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pi</td>
<td>Trigonometric</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

\[\text{Pi}()\]

This function takes no parameters. You must still include the parentheses, however, or it will be evaluated as a tag rather than a function.

**Returned value**

Returns the value of \(\pi\).

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>(\text{Pi}()) // Returned value = 3.141593</td>
</tr>
</tbody>
</table>
**Sin**

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sin</td>
<td>Trigonometric</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Description**

Calculates the Sine of a value.

**Syntax**

```plaintext
Sin( numAngle )
```

**numAngle**

The Angle (in radians) from which to calculate the Sine.

**Returned value**

Returns the Sine of numAngle.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>Sin( 0 )</code> // Returned value = 0.000000</td>
</tr>
<tr>
<td>Tag</td>
<td><code>Sin( 1.570796 )</code> // Returned value = 1.000000</td>
</tr>
</tbody>
</table>

**Tan**

The function **Tan** calculates the tangent of an angle.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tan</td>
<td>Trigonometric</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
Tan( numAngle )
```

**numAngle**

The angle in radians.

**Tip:** 2π radians is the same as 360 degrees.

**Returned value**

This function returns the tangent of the specified angle.

The function cannot return the actual tangent of π/2 (i.e., `Tan(Pi()/2)` or equivalent), because that is infinite. Instead, the function returns the largest number possible, given the limited precision of the value returned by the function `Pi`. 
Examples
Calculate the tangent of 1 radian:

\( \tan(1) \)

Calculate the tangent of \( \pi \) radians:

\( \tan(\pi()) \)

Calculate the tangent of the value of the project tag MyAngle:

\( \tan(MyAngle) \)
Screen functions

These functions are used to open and close project screens.

**Close**

Close is a built-in scripting function that closes an open project screen.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Screen</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

`Close(strScreen, optNumID)`

`Close(strScreen) | , optNumID)`

*strScreen*

The name of the screen (not including the `.scr` extension) to be closed.

*optNumID*

The specific ID or instance number of the screen to be closed, if there is more than one screen with the same name open. (The ID is assigned when the screen is opened with the function `Open`.)

This is an optional parameter. If no value is specified, the default ID is 0.

**Returned value**

This function does not return any value.

**Notes**

If the option *Enforce Web functionality equivalence* is selected in the project settings, this function cannot be called from *Global Procedures* or *Script worksheets*. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see *Preferences tab* on page 108.

The following limitations also apply to how the function can be used in project screens in Mobile Access:

- The function cannot be called from the sub-routine *Screen_OnClose* in the *Screen Script*; and
- You should avoid creating a script that both shows a dialog box (e.g., a message box that is shown using the VBScript function `MsgBox` or the IWS built-in function `ShowMessageBox`) and opens or closes a screen. The Mobile Access web interface can display only one screen at a time, and opening a new screen will automatically close the previous one. Depending on the order of execution, this can lead to unexpected behavior such as a dialog box being orphaned or not being shown at all.

Finally, in some cases, you do not need to call this function to close a screen because the screen will be closed automatically when another screen replaces it. For more information, see *Screen Attributes*.

**Examples**

Close the screen named "main":

```plaintext
Close( "main" )
```

Close the screen named "alarms":

```plaintext
Close( "alarms" )
```
Close the screen named "main" with ID 10:

```
Close( "main", 10 )
```

**Open**

The function **Open** opens a project screen on the client with some specified settings.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Screen</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

**Syntax**

```
Open( strScreenAndProperties, optNumX1, optNumY1, optNumX2, optNumY2, optNumResizeFlag, optNumID, optStrMnemonicList )
```

```
Open( strScreenAndProperties{   | , optNumX1{   | , optNumY1, optNumX2, optNumY2, { optNumResizeFlag | 0
| 1 } } }{   | , optNumID }{   | , optStrMnemonicList })
```

- **strScreenAndProperties**
  - The name of the screen to be opened. (The screen file extension *scr* is assumed, so you do not need to include it.)

If you specify only the screen name, then it will be opened with its default properties (as specified in **Screen Attributes**). You can, however, customize the properties as follows:

<table>
<thead>
<tr>
<th>Property</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Bar</td>
<td>EnableTitleBar: Enable</td>
</tr>
<tr>
<td>System Menu</td>
<td>SystemMenu: Enable</td>
</tr>
<tr>
<td>Maximize Box</td>
<td>Maximize: Enable</td>
</tr>
<tr>
<td>Minimize Box</td>
<td>Minimize: Enable</td>
</tr>
<tr>
<td>Style</td>
<td>Style: Overlapped</td>
</tr>
<tr>
<td>Border</td>
<td>Border: None</td>
</tr>
</tbody>
</table>

For more information, see "Examples" below.

- **optNumX1**
- **optNumY1**
- **optNumX2**
- **optNumY2**

The coordinates, in pixels, for the top-left (X1,Y1) and bottom-right (X2,Y2) corners of the screen.

These are optional parameters. If no values are specified, then the default screen size and location are used.

Please note the following special circumstances:

- You can open the screen at the mouse’s current position by using **Open("screen",1)**, or **Open("screen",1,-1,-1,-1,...)** if the parameters at the end are needed.
- If **optNumX1** equals **optNumX2** and **optNumY1** equals **optNumY2**, then the default screen size is used but the screen is centered at (X1,Y1).
- If **optNumX2** is less than **optNumX1** and/or **optNumY2** is less than **optNumY1**, or if all four parameters are set to −1, then the parameters are ignored and the default screen size and location are used.
**optNumResizeFlag**

Specifies whether objects in the screen will be resized when the screen is opened:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Screen objects will not be resized.</td>
</tr>
<tr>
<td>1</td>
<td>Screen objects will be automatically resized to fit the new dimensions of the screen, as specified by the coordinates described above. The resizing is done at the moment the screen is opened, so if the user changes the screen size after the screen is opened, then the objects will not be resized again.</td>
</tr>
</tbody>
</table>

This parameter is required if all four coordinates are specified.

**optNumID**

An ID or instance number to be assigned to the screen, because you can open multiple instances of the same screen file. (This ID is required when a screen is closed using the function `Close`.)

This is an optional parameter. If no value is specified, then the default ID is 0.

**optStrMnemonicList**

A string that describes how the custom properties of any generic objects or linked symbols in the screen will be completed when the screen is opened. This string has the following syntax...

```
#Label: Value
```

...where `Label` is the name of the property and `Value` is the tag, expression, or literal value that the property will receive. You can declare more than one mnemonic as long as the mnemonics are separated by spaces.

You can also specify a external text file that contains the mnemonics separated by either spaces or line returns. The file must have the extension `*.MNE`, and it must be saved in the Web sub-folder of your project folder. Please note that the file extension is assumed, so you do not need to include it in parameter, and if you want to save the file in a deeper sub-folder, then you need to specify a file path relative to the Web sub-folder.

Note: Mnemonics are not supported when opening project screens in Mobile Access.

For more information, see "Examples" below.

**Returned value**

This function will return one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Failure</td>
</tr>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Notes**

If the option **Enforce Web functionality equivalence** is selected in the project settings, this function cannot be called from **Global Procedures** or **Script worksheets**. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see **Preferences tab** on page 108.

The following limitations also apply to how the function can be used in project screens in Mobile Access:

- The function cannot be called from the sub-routine **Screen_OnClose** in the **Screen Script**; and
• You should avoid creating a script that both shows a dialog box (e.g., a message box that is shown using the VBScript function MsgBox or the IWS built-in function ShowMessageBox) and opens or closes a screen. The Mobile Access web interface can display only one screen at a time, and opening a new screen will automatically close the previous one. Depending on the order of execution, this can lead to unexpected behavior such as a dialog box being orphaned or not being shown at all.

Examples

Open the screen using the default screen size, location, and ID:

```
Open("main")
```

Open the screen at the mouse’s current position:

```
Open("main",1)
```

Open the screen at the mouse’s current position and assign it an ID of 10:

```
Open("main",1,-1,-1,-1,0,10)
```

Open the screen using the default screen size but centered at the coordinates (500,250), do not resize the screen objects, and assign the screen an ID of 10:

```
Open("main",500,250,500,250,0,10)
```

Open the screen using the default screen size and location, use the default ID, and replace the custom properties Mne1 and Mne2 with Tag1 and Tag2, respectively:

```
Open("main",-1,-1,-1,-1,0,0,"#Mne1:Tag1 #Mne2:Tag2")
```

Open the screen using the default screen size and location, use the default ID, and replace the custom properties with values defined in the mnemonics file located at: `projectname\Web\mnemonic.mne`

```
Open("main",-1,-1,-1,-1,0,0,"mnemonic")
```

Open the screen using the default screen size but centered at the coordinates (500,250), do not resize the screen objects, assign the screen an ID of 10, and replace the custom properties with values defined in the mnemonics file located at: `projectname\Web\mnemonic.mne`

```
Open("main",500,250,500,250,0,10,"mnemonic")
```

Open the screen using the default screen size and location, assign it an ID of 2, and replace the custom properties with values defined in the mnemonics file located at: `projectname\Web\Mnemonics\mnemonic.mne`

```
Open("main",-1,-1,-1,-1,0,2,"Mnemonics\mnemonic")
```

Open the screen using the default screen size and location, do not resize the screen objects, use the default ID, and customize the screen properties:

```
Open("main; EnableTitleBar: Enable; TitleBar: Main Screen; SystemMenu: Enable; Maximize: Disable; Minimize: Disable; Style: Popup; Border: Thin",-1,-1,-1,-1,0,0)
```
**OpenPrevious**

*OpenPrevious* is a built-in scripting function that re-opens the last screen to be closed.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenPrevious</td>
<td>Screen</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

*OpenPrevious*(optNumX1, optNumY1, optNumX2, optNumY2)

*OpenPrevious*({ | optNumX1, optNumY1, optNumX2, optNumY2 })

**optNumX1**

**optNumY1**

**optNumX2**

**optNumY2**

The coordinates, in pixels, for the upper-left (X1,Y1) and lower-right (X2,Y2) corners of the screen.

These are optional parameters. If no values are specified, then the default screen size and location are used. For more information, see Screen Attributes.

**Returned value**

This function will return one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Failure</td>
</tr>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Notes**

If the option **Enforce Web functionality equivalence** is selected in the project settings, this function cannot be called from **Global Procedures** or **Script worksheets**. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

The following limitations also apply to how the function can be used in project screens in Mobile Access:

- The function cannot be called from the sub-routine **Screen_OnClose** in the **Screen Script**; and
- You should avoid creating a script that both shows a dialog box (e.g., a message box that is shown using the VBScript function *MsgBox* or the IWS built-in function *ShowMessageBox*) and opens or closes a screen. The Mobile Access web interface can display only one screen at a time, and opening a new screen will automatically close the previous one. Depending on the order of execution, this can lead to unexpected behavior such as a dialog box being orphaned or not being shown at all.

**Examples**

Open the previous screen using its default size and location:

`OpenPrevious()`

Open the previous screen in the top-left corner of the display and sized to 800x600:

`OpenPrevious(0, 0, 800, 600)`
**ShowInplaceInput**

The function `ShowInplaceInput` shows a simple text input dialog at a specified location in the project client/viewer.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowInplaceInput</td>
<td>Screen</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
ShowInplaceInput("tagOutput",numStartXPos,numStartYPos,optNumMin,optNumMax,optNumEnablePasswordMode)
```

**tagOutput**

The name of a tag that will receive the input.

> **Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**numStartXPos**

The starting X position of the top-left corner of the input dialog — that is, the number of pixels between that corner and the left side of the display.

**numStartYPos**

The starting Y position of the top-left corner of the input dialog — that is, the number of pixels between that corner and the top of the display.

**optNumMin**

The minimum numeric value that will be accepted by the input dialog.

This is an optional parameter. If no value is specified, then the dialog will accept any value.

**optNumMax**

The maximum numeric value that will be accepted by the input dialog.

This is an optional parameter. If no value is specified, then the dialog will accept any value.

**optNumEnablePasswordMode**

An option to enable password mode, which obfuscates the operator's input as if it's a password:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Show input as plain text.</td>
</tr>
<tr>
<td>1</td>
<td>Obfuscate input.</td>
</tr>
</tbody>
</table>

This is an optional parameter. If no value is specified, then the default is 0.

**optNumShowOSVK**

An option to show the default Virtual Keyboard, which is configured in the project settings:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not show Virtual Keyboard.</td>
</tr>
<tr>
<td>1</td>
<td>Show Virtual Keyboard.</td>
</tr>
</tbody>
</table>

This is an optional parameter. If no value is specified, then the default is 0.
**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid tag specified for tagOutput.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-3</td>
<td>Viewer is not running.</td>
</tr>
</tbody>
</table>

**Notes**

If the option **Enforce Web functionality equivalence** is selected in the project settings, this function cannot be called from **Global Procedures** or **Script worksheets**. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see **Preferences tab** on page 108.

**Examples**

```plaintext
ShowInplaceInput("OperatorInput",50,50)
ShowInplaceInput("OperatorInput",50,50,1,100)
ShowInplaceInput("OperatorInput",50,50,1,100,0,1)
```

**ShowMessageBox**

The function **ShowMessageBox** shows a message box with either an **OK** button or **Yes / No** buttons.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShowMessageBox</td>
<td>Screen</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
ShowMessageBox(strMessage, optNumButtons, optStrTitle)
```

- **strMessage**
  
  The message body that will be displayed in the box.

- **optNumButtons**
  
  A numeric flag that specifies which kind of confirmation buttons to display in the message box:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><strong>OK</strong> button</td>
</tr>
<tr>
<td>4</td>
<td><strong>Yes / No</strong> buttons</td>
</tr>
</tbody>
</table>

**Tip:** To add an exclamation icon to the box — that is, to make it an alert or warning rather than a plain message — add 48 (vbExclamation) to this parameter. For more information, see "Examples" below.
This is an optional parameter. If no value is specified, then the default is 0.

**optStrTitle**

The title of the message.

This is an optional parameter. If no value is specified, then no title will be displayed.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Message box not displayed because the Viewer is not open; see &quot;Notes&quot; below.</td>
</tr>
<tr>
<td>1</td>
<td>User clicked <strong>OK</strong>.</td>
</tr>
<tr>
<td>6</td>
<td>User clicked <strong>Yes</strong>.</td>
</tr>
<tr>
<td>7</td>
<td>User clicked <strong>No</strong>.</td>
</tr>
</tbody>
</table>

**Notes**

Unlike other Screen functions, this function can be called from Global Procedures and Script worksheets, and when it is, the message box is displayed in the Viewer module on the project runtime server. If the Viewer module is not open — that is, if the Viewer task is not running — the message box cannot be displayed and the function returns 0.

When this function is used in project screens on Mobile Access, it has been enhanced to duplicate the functionality of the VBScript function *MsgBox*.

The following limitations also apply to how the function can be used in project screens in Mobile Access:

- The function cannot be called from the sub-routine *Screen_OnClose* in the Screen Script; and
- You should avoid creating a script that both shows a dialog box (e.g., a message box that is shown using the VBScript function *MsgBox* or the IWS built-in function *ShowMessageBox*) and opens or closes a screen. The Mobile Access web interface can display only one screen at a time, and opening a new screen will automatically close the previous one. Depending on the order of execution, this can lead to unexpected behavior such as a dialog box being orphaned or not being shown at all.

**Examples**

Display a plain message with an **OK** button:

```plaintext
ShowMessageBox("The action could not be completed.")
```

Display a question with **Yes** / **No** buttons:

```plaintext
ShowMessageBox("Continue with action?",4)
```

Display an alert with an **OK** button and a title:

```plaintext
ShowMessageBox("The action could not be completed.",0+48,"Alert")
```
Security functions

These functions are used to manage users and groups in the project’s security system.

**BlockUser**

BlockUser is a built-in scripting function that blocks an existing user from logging onto a project. This allows you to disable a user account without deleting it.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockUser</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

BlockUser(*strUserName*)

*strUserName*

The name of the user to block.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>User blocked successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>3</td>
<td>Specified user does not exist.</td>
</tr>
<tr>
<td>4</td>
<td>User currently logged on does not have the rights to block (i.e., user does not have Edit Security System).</td>
</tr>
<tr>
<td>5</td>
<td>The operation on the distributed security system failed.</td>
</tr>
<tr>
<td>6</td>
<td>User cannot be blocked.</td>
</tr>
<tr>
<td>7</td>
<td>The current Security Mode does not allow user to be blocked/unblocked.</td>
</tr>
<tr>
<td>8</td>
<td>Internal error.</td>
</tr>
</tbody>
</table>

**Notes**

If you use this function to block a user, then the option User is blocked is selected in the User Account dialog. For more information, see Creating and configuring users on page 486.

**Examples**

Block the user named Bob:

```javascript
BlockUser( "Bob" )
```

Block the user named in position 3 of the array badUsers:

```javascript
BlockUser( badUsers[3] )
```

Block the user that is currently logged on:

```javascript
BlockUser( UserName )
```
**CheckESign**

CheckESign is a built-in function that prompts the run-time user to electronically sign an event by entering their username and password. You can call this function to secure scripts and expressions, just as you can select the E-Sign option in object properties to secure screen objects and animations.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckESign</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

CheckESign(optNumLevel)

CheckESign({ | optNumLevel})

**optNumLevel**

The security level (from 0 to 255, Runtime only) to which the user must have access in order to sign. For more information, see About security access levels on page 465.

This parameter is optional; if no value is specified, the default value is 0.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Username and/or password not accepted, or the user does not have access to the specified level.</td>
</tr>
<tr>
<td>1</td>
<td>Username and password accepted. The event is saved in the event history with the user's signature.</td>
</tr>
</tbody>
</table>

**Notes**

Calling this function displays a dialog box on the client, where the user must enter their username and password.

Users and groups are managed in the project security system. For more information, see Project Security on page 463.

**Examples**

Prompt the user for their username and password — the user must have access to security level 20...

CheckESign(20)

**CheckSecurityLevel**

The function CheckSecurityLevel checks whether the current user has access to the specified security level.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckSecurityLevel</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

CheckSecurityLevel(numLevel, optNumType)

CheckSecurityLevel(numLevel | ,{optNumType | 0 | 1})

**numLevel**
The security level to be checked.

**optNumType**

The type of security level to check: 0 is Runtime, 1 is Development.

This is an optional parameter; if no value is specified, the default value is 0.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The current user does not have access to the specified security level.</td>
</tr>
<tr>
<td>1</td>
<td>The current user has access to the specified security level.</td>
</tr>
</tbody>
</table>

**Notes**

Each user can belong to multiple groups, and those groups typically have access to different security levels. Rather than check the groups to which a user belongs, and from that try to determine the security levels to which the user has access, you can use this function to check the security levels directly.

For more information about groups and security levels, see Group Account dialog on page 482.

**Examples**

The user “Bob” belongs to the groups “GroupA” and “GroupB”. GroupA has access to runtime levels 10 to 20, and GroupB has access to development levels 30 to 40. That means Bob has access to runtime security levels 10 to 20 and development security levels 30 to 40.

Given this, if Bob is the current user (i.e., the user who is currently logged onto the client where the function is executed) and this function is called to check for access to runtime security level 17...

```
CheckSecurityLevel(17)
```

...it returns a value of 1, meaning that Bob has access.

Similarly, if this function is called to check for access to development security level 33...

```
CheckSecurityLevel(33,1)
```

...it again returns a value of 1.

However, if this function is called to check for access to runtime security level 25...

```
CheckSecurityLevel(25,0)
```

...it returns a value of 0, meaning that Bob does not have access.

**CreateUser**

The function `CreateUser` creates a new user in your project’s security system.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateUser</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
CreateUser(optStrUserName, optStrGroup, optStrPassw, optStrUserFullName)

CreateUser({ | optStrUserName, optStrGroup{ | , optStrPassw{ | , optStrUserFullName } }})
```
**optStrUserName**

The name of the user to be created, and the group(s) to which the user will belong. You can specify multiple groups by separating them with a comma.

These are optional parameters; if no values are specified, a dialog box will be displayed on the client so that the current user can provide the information.

**optStrGroup**

The user’s password.

This is an optional parameter; if no value is specified, the user will not have a password. (To specify one later, either call the function `SetPassword` or edit the user’s settings in the project security system.)

**optStrUserFullName**

The full name of the user.

This is an optional parameter; if no value is specified, the user will not have a full name. (To specify one later, edit the user’s settings in the project security system.)

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1000</td>
<td>Could not display dialog box. The function should executed on the client.</td>
</tr>
<tr>
<td>-1</td>
<td>Internal error. Please contact Support.</td>
</tr>
<tr>
<td>0</td>
<td>New user created successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>3</td>
<td>User name already exists.</td>
</tr>
<tr>
<td>4</td>
<td>Group does not exist.</td>
</tr>
<tr>
<td>5</td>
<td>Failed to save to configuration file.</td>
</tr>
<tr>
<td>6</td>
<td>Invalid user.</td>
</tr>
<tr>
<td>7</td>
<td>User full name already exists.</td>
</tr>
<tr>
<td>8</td>
<td>Reentrant function call not allowed.</td>
</tr>
<tr>
<td>9</td>
<td>User clicked <strong>Cancel</strong> on the Create User dialog box.</td>
</tr>
<tr>
<td>10</td>
<td>Invalid password, check the minimum password size specified for the group.</td>
</tr>
<tr>
<td>11</td>
<td>Invalid group. (Group may not have <strong>Runtime group</strong> option selected.)</td>
</tr>
<tr>
<td>12</td>
<td>Create User dialog box is already displayed, cannot display another dialog box. (For example, if the user clicked <strong>OK</strong> without providing all of the required information.)</td>
</tr>
<tr>
<td>13</td>
<td>Current user does not have sufficient privileges to create users (i.e., the option <strong>Edit Security System</strong> is not selected in the Security System settings).</td>
</tr>
<tr>
<td>14</td>
<td>The current Security Mode does not allow a user to be created.</td>
</tr>
</tbody>
</table>

**Notes**

Users created with this function have the **Runtime user** option selected in the **User Account** dialog. For more information, see **Creating and configuring users** on page 486.
Examples

Display the Create User dialog box on the client, to get the information from the current user:

CreateUser()

Create a user named "Bob" in the group "Admin", with the password "Chocolate":

CreateUser("Bob","Admin","Chocolate")

Create a user named "Albert" (full name "Albert Jones") in the group "Engineering", with the password "EMC2":

CreateUser("Albert","Engineering","EMC2","Albert Jones")

ExportSecuritySystem

This function exports the security system configuration to an encrypted file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportSecuritySystem</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

ExportSecuritySystem( strFileName, strPassword )

strFileName

The complete file path and name where you want to save the configuration file.

strPassword

The main password for the security system. This same password will be used to protect the exported file.

Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>0</td>
<td>Couldn't write security data.</td>
</tr>
<tr>
<td>1</td>
<td>File exported successfully.</td>
</tr>
</tbody>
</table>

Notes

This function can be called only from scripts executed on the project server. None of the connected clients — not even the client running on the same computer as the project server, because it runs as a separate process on that computer — have the necessary access to the security system. Therefore, generally speaking:

- It can be called from the Startup Script (which is executed when the project itself is run), Script Groups (which are continuously executed by the Background Task), and any Global Procedures called by them; and
- It cannot be called from the Graphics Script (which is executed separately by each client), Screen Scripts (which are attached to individual screens), and Command animations.
Appendix: Built-in Scripting Language

For more information, see VBScript Interfaces in the Software on page 944.
To work around this limitation, create a Script Group that actually calls the function, configure an appropriate tag/expression trigger in the Execution box of the Script worksheet, and then create a project screen that somehow changes the value of that tag/expression. Therefore, when a user on a connected client uses the screen to change the value, the Script Group will be executed on the project server.

Examples

ExportSecuritySystem( "C:\security.txt" )

ExportSecuritySystem( "C:\security.txt", "mypassword" )

GetSecuritySystemStatus

This function gets the status of the security system and its connection to the authentication server, when the security mode is either Distributed–Client or Domain (LDAP).

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSecuritySystemStatus</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

GetSecuritySystemStatus({ | { optNumType | 0 | 1 | 2 })

GetSecuritySystemStatus()  GetSecuritySystemStatus( optNumType )

optNumType

The type of action to take to update the status.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Perform a fast check using either Ping or Bind (depending on the server settings), but take no other actions.</td>
</tr>
<tr>
<td>1</td>
<td>Force reload of users and groups from the authentication server.</td>
</tr>
<tr>
<td>2</td>
<td>Clear cached users and groups.</td>
</tr>
</tbody>
</table>

This is an optional parameter; if no value is specified, then the default is 0.

Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Security Mode is Distributed–Client</th>
<th>Security Mode is Domain (LDAP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No cache</td>
<td>Connection timeout</td>
</tr>
<tr>
<td>1</td>
<td>Updated cache</td>
<td>Bind timeout</td>
</tr>
<tr>
<td>2</td>
<td>outdated local cache</td>
<td>Query timeout</td>
</tr>
<tr>
<td>3</td>
<td>Outdated server cache</td>
<td>Disconnected</td>
</tr>
<tr>
<td>4</td>
<td>Disconnected from server</td>
<td>Connected</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>No users or groups returned by query</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>Invalid user or group</td>
</tr>
</tbody>
</table>
Notes
This function returns the same value that is sent to the project tag configured in the Status Tag box, in the security system server settings. However, this function returns the value immediately, while the project tag configured in the Status Tag box is only updated periodically (according to Synchronization Period for Distributed-Client or Retry Interval for Domain (LDAP)). As such, there may be times when this function’s returned value and the value of the project tag do not match.

Also, there are other actions besides calling this function that update the status:

- When a user logs on to the project. Specifically, if the user logs on via the built-in LogOn dialog (invoked by either calling the LogOn function or selecting the LogOn menu command in the Viewer), then the status is updated before the dialog is displayed.
- When the authentication server is offline and the retry interval (configured in the security system server settings) has elapsed.
- When the security system settings are opened in the development application.

Whenever the status is updated, the new value is immediately sent to the project tag configured in the Status Tag box.

Examples
Get the status of the security system:

GetSecuritySystemStatus()

Force the security system to reload all users and groups from the authentication server:

GetSecuritySystemStatus( 1 )

GetUserFullName
This function gets the full name (if any) of a specified user in the project security system.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetUserFullName</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax
GetUserFullName(strUserName, "tagUserFullName")

strUserName
The name of a user in the project security system.

tagUserFullName
The name of a tag (String type) that will receive the full name of the specified user. If the specified user does not have a full name defined, then the tag will receive an empty string ("").

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

Returned value
This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Specified user does not exist.</td>
</tr>
<tr>
<td>1</td>
<td>Success: specified user exists.</td>
</tr>
</tbody>
</table>
Examples

Get the full name of the currently logged user (via the system tag `UserName`):

```
getUserFullName( UserName, "UserFullName" )
```

Get the full name of the user "engineer1":

```
getUserFullName( "engineer1", "UserFullName" )
```

**GetUserNames**

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetUserNames</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
getUserNames( "tagUsers", optNumUserType, "opttagGroups" )
```

**tagUsers**

Name of the array tag that will receive users.

**optNumUserType**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Return all users</td>
</tr>
<tr>
<td>1</td>
<td>Only users created during runtime</td>
</tr>
<tr>
<td>2</td>
<td>Only users created using the development environment</td>
</tr>
</tbody>
</table>

**opttagGroups**

Name of the array tag that will receive the group for each specific user.

**Returned value**

The number of users, or a negative number that can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>-2</td>
<td><code>tagUsers</code> is invalid</td>
</tr>
<tr>
<td>-3</td>
<td><code>optNumUserType</code> is invalid</td>
</tr>
<tr>
<td>-4</td>
<td><code>opttagGroups</code> is invalid</td>
</tr>
<tr>
<td>-5</td>
<td>Error, function cannot be called in the Thin Client</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfUsers</td>
<td><code>getUserNames( &quot;UsersArray&quot; )</code> // Retrieves all users, stores names in the <code>UsersArray</code> tag and the number of users in the <code>NumberOfUsers</code> tag.</td>
</tr>
<tr>
<td>NumberOfUsers</td>
<td><code>getUserNames(&quot;UsersArray&quot;, 1)</code> // Retrieves all users created during runtime, stores names in the <code>UsersArray</code> tag and the number of users in the <code>NumberOfUsers</code> tag.</td>
</tr>
<tr>
<td>NumberOfUsers</td>
<td><code>getUserNames(&quot;UsersArray&quot;, 2)</code> // Retrieves all users created in the development environment, stores names in the <code>UsersArray</code> tag and the number of users in the <code>NumberOfUsers</code> tag.</td>
</tr>
</tbody>
</table>
### Tag Name

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfUsers</td>
<td><code>GetStringNames(&quot;UsersArray&quot;, 2, &quot;Groups&quot;)  // Retrieves all users created in the development environment, stores names in the UsersArray tag and the number of users in the NumberOfUsers tag. The group name per each user is stored in the Groups tag.</code></td>
</tr>
</tbody>
</table>

### `GetUserPwdAging`

The function `GetUserPwdAging` gets the age of the password for a specified user — that is, the time remaining until the password expires, or if it has expired, the time since it expired.

#### Syntax

```plaintext
GetUserPwdAging(strUser)
```

`strUser`  
The name of the user.

#### Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>less than 0</td>
<td>Number of hours since the password expired.</td>
</tr>
<tr>
<td>0</td>
<td>The specified user is not logged on.</td>
</tr>
<tr>
<td>greater than 0</td>
<td>Number of hours until the password expires.</td>
</tr>
</tbody>
</table>

**Note:** If the function is not executed correctly (e.g., if the specified user name is invalid), or if the specified user is not logged on, then the function also returns BAD quality.

#### Examples

Get the age of the password for the user "John":

```plaintext
GetUserPwdAging("John")
```

Get the age of the password for the user who is currently logged on:

```plaintext
GetUserPwdAging(UserName)
```

### `GetUserState`

Use to see the current status of a selected user.

#### Syntax

```plaintext
GetUserState(strUserName)
```

`strUserName`
The name of the user

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>Specified user does not exist.</td>
</tr>
<tr>
<td>-2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>0</td>
<td>Specified user is unblocked.</td>
</tr>
<tr>
<td>1</td>
<td>Specified user is blocked.</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetUserState( &quot;Bob&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>GetUserState( &quot;Albert&quot; )</td>
</tr>
</tbody>
</table>

**ImportSecuritySystem**

This function imports a security system configuration from an external file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImportSecuritySystem</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
ImportSecuritySystem( strSecuritySystemPassword, strFileName, strFilePassword, optNumMode )
```

```plaintext
ImportSecuritySystem( strSecuritySystemPassword, strFileName, strFilePassword| , , optNumMode | 0 | 1 | 2 )
```

**strSecuritySystemPassword**

The main password for the project’s current security system configuration. (The security system must be enabled.)

**strFileName**

The complete file path and name of the configuration file that you want to import. (The file must have been previously exported from a IWS project using either the Security System configuration tool or the function ExportSecuritySystem.)

**strFilePassword**

The password for the specified configuration file.

**optNumMode**

A numeric flag indicating how the imported settings should be handled:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Append the imported settings to the current settings. In the event of a conflict, replace with the imported settings.</td>
</tr>
<tr>
<td>1</td>
<td>Append the imported settings to the current settings. In the event of a conflict, keep the current settings.</td>
</tr>
</tbody>
</table>
**Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Completely replace the current settings with the imported settings.</td>
</tr>
</tbody>
</table>

This parameter is optional; if no value is specified, then the default value is 0.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>0</td>
<td>Couldn't read security data.</td>
</tr>
<tr>
<td>1</td>
<td>File imported successfully.</td>
</tr>
</tbody>
</table>

**Notes**

This function can be called only from scripts executed on the project server. None of the connected clients — not even the client running on the same computer as the project server, because it runs as a separate process on that computer — have the necessary access to the security system. Therefore, generally speaking:

- It can be called from the Startup Script (which is executed when the project itself is run), Script Groups (which are continuously executed by the Background Task), and any Global Procedures called by them; and

- It cannot be called from the Graphics Script (which is executed separately by each client), Screen Scripts (which are attached to individual screens), and Command animations.

For more information, see VBScript Interfaces in the Software on page 944.

To work around this limitation, create a Script Group that actually calls the function, configure an appropriate tag/expression trigger in the *Execution* box of the Script worksheet, and then create a project screen that somehow changes the value of that tag/expression. Therefore, when a user on a connected client uses the screen to change the value, the Script Group will be executed on the project server.

**Examples**

```vbs
ImportSecuritySystem( "curr3ntPa55w0rd", "C:\security.txt", "1mp0rtPa55w0rd" )
```

```vbs
ImportSecuritySystem( "curr3ntPa55w0rd", "C:\security.txt", "1mp0rtPa55w0rd", 2 )
```

**RemoveUser**

The function `RemoveUser` removes a specified user from your project’s security system.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveUser</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```vbs
RemoveUser (strUserName)
```

`strUserName`  
The name of the user to be removed.
Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The specified user was successfully removed.</td>
</tr>
<tr>
<td>1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>3</td>
<td>The current user does not have sufficient privileges to remove other users (i.e., the user does not have the option Edit Security System selected in the Security System settings).</td>
</tr>
<tr>
<td>4</td>
<td>The specified user cannot be removed.</td>
</tr>
<tr>
<td>5</td>
<td>The specified user does not exist.</td>
</tr>
<tr>
<td>6</td>
<td>Component-level failure (e.g., the LDAP server returned an error).</td>
</tr>
<tr>
<td>7</td>
<td>Failed to save changes to the configuration file.</td>
</tr>
<tr>
<td>8</td>
<td>The current security mode does not allow the specified user to be removed.</td>
</tr>
</tbody>
</table>

Examples

Remove the user named "Bob":

RemoveUser("Bob")

Remove the user specified by the tag InvalidUser:

RemoveUser(InvalidUser)

Note: You can use this function to remove only those users who were created during project run time using the function CreateUser. You cannot use this function to remove users who were created using the Security System configuration tools in the project development environment. For more information, see Project Security on page 463.

SetPassword

The function SetPassword sets a new password for a specified user in your project’s security system.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetPassword</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

SetPassword(optStrUserName, optStrNewPassword, optStrConfirmPassword, optStrCurrentPassword)

SetPassword({ | optStrUserName| | , optStrNewPassword| | , optStrConfirmPassword| | , optStrCurrentPassword } } )

**optStrUserName**

The name of the user.

This is an optional parameter; if no value is specified, a dialog box will be displayed on the client so that the current user can select the name.

**optStrNewPassword**
The new password for the specified user.
This is an optional parameter; if no value is specified, a dialog box will be displayed on the client so that the current user can type the password.

**optStrConfirmPassword**

The new password again, to confirm that it has been typed correctly.
This is an optional parameter; if no value is specified, a dialog box will be displayed on the client so that the current user can type the password.

**optStrCurrentPassword**

The current password for the specified user, to authorize the change.
This is an optional parameter; if no value is specified, a dialog box will be displayed on the client so that the current user can type the password.

### Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1000</td>
<td>Could not display dialog box, because the function was called on the project runtime server. The function should be called on the client.</td>
</tr>
<tr>
<td>-1</td>
<td>Internal error. Please contact Customer Support.</td>
</tr>
<tr>
<td>0</td>
<td>Password set successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>3</td>
<td>The specified user does not exist.</td>
</tr>
<tr>
<td>4</td>
<td>Reentrant call not allowed.</td>
</tr>
<tr>
<td>5</td>
<td>User clicked <strong>Cancel</strong> in the <strong>Set Password</strong> dialog box.</td>
</tr>
<tr>
<td>6</td>
<td>The specified group does not exist.</td>
</tr>
<tr>
<td>7</td>
<td>The specified password is too weak.</td>
</tr>
<tr>
<td>8</td>
<td>Invalid password.</td>
</tr>
<tr>
<td>9</td>
<td>Invalid user.</td>
</tr>
<tr>
<td>10</td>
<td>The current user does not have sufficient privileges to set user passwords (i.e., the current user does not have the option <strong>Edit Security System</strong> selected in the Security System settings).</td>
</tr>
<tr>
<td>11</td>
<td>Server offline.</td>
</tr>
<tr>
<td>12</td>
<td>Communication error (e.g., the function was called on remote client that cannot communicate with the LDAP server).</td>
</tr>
<tr>
<td>13</td>
<td>Confirmation does not match new password.</td>
</tr>
<tr>
<td>14</td>
<td><strong>Set Password</strong> dialog box is already displayed, cannot display another dialog box. (For example, if the user clicked <strong>OK</strong> without providing all of the required information.)</td>
</tr>
<tr>
<td>15</td>
<td>The current security mode does not allow user passwords to be changed.</td>
</tr>
</tbody>
</table>

### Notes

If you do not want the dialog box to be displayed on the client, you must provide all four parameters to the function. Otherwise, the dialog box will be displayed in order to get the remaining information from the current user.
Examples
Display the Set Password dialog box on the client, to get the information from the current user:

SetPassword()

Change the password for user "Admin", but display the dialog box in order to get the new password and authorization:

SetPassword("Admin")

Change the password for user "admin" to the value of the tag newPassword, but display the dialog box in order to get authorization:

SetPassword("Admin",newPassword,newPassword)

Change the password for user 'admin' to the value of the tag newPassword, without displaying the dialog box:

SetPassword("Admin",newPassword,newPassword,"DLfVU89Y")

UnblockUser
This function unblocks a blocked user in the security system.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnblockUser</td>
<td>Security</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

UnblockUser( strUserName )

strUserName
The name of the user to unblock.

Returned value
This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>User unblocked successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>3</td>
<td>Specified user does not exist.</td>
</tr>
<tr>
<td>4</td>
<td>User currently logged on does not have the rights to unblock (i.e., user does not have Edit Security System).</td>
</tr>
<tr>
<td>5</td>
<td>The operation on the distributed security system failed.</td>
</tr>
<tr>
<td>6</td>
<td>Specified user cannot be blocked.</td>
</tr>
<tr>
<td>7</td>
<td>The current Security Mode does not allow user to be blocked/unblocked.</td>
</tr>
<tr>
<td>8</td>
<td>Internal error.</td>
</tr>
</tbody>
</table>
Notes
If you use this function to unblock a user, then the option User is blocked is cleared in the User Account dialog. For more information, see Creating and configuring users on page 486.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>UnblockUser( &quot;Bob&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>UnblockUser( &quot;Albert&quot; )</td>
</tr>
</tbody>
</table>
Module Activity functions

These functions are used to manage a project’s various runtime modules — such as background tasks, the data server, and the project viewer — as well as those modules’ interactions with the operating system.

**AppActivate**

AppActivate is a built-in scripting function that activates (i.e., brings to the front) another application window that is already open.

- **Function**: Module Activity
- **Execution Group**: Asynchronous
- **Windows**: Supported
- **Embedded**: Supported
- **Thin Client**: Supported
- **Mobile Access**: Not supported

**Syntax**

```plaintext
AppActivate(strAppTitle | | optNumActiv | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | optNumTimeout | )
```

**strAppTitle**

The full title (as shown in the title bar) of the application window.

**optNumActiv**

Controls how the specified window is to be activated:

<table>
<thead>
<tr>
<th>Value</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SW_HIDE</td>
<td>Hides the currently active window and then activates the specified window.</td>
</tr>
<tr>
<td>1</td>
<td>SW_SHOWNORMAL</td>
<td>Activates and displays the specified window. If the window is minimized or maximized, then it is restored to its original size and position. You should use this command when displaying a window for the first time.</td>
</tr>
<tr>
<td>2</td>
<td>SW_SHOWMINIMIZED</td>
<td>Activates the specified window and then minimizes it.</td>
</tr>
<tr>
<td>3</td>
<td>SW_SHOWMAXIMIZED</td>
<td>Activates the specified window and then maximizes it.</td>
</tr>
<tr>
<td>4</td>
<td>SW_SHOWNOACTIVATE</td>
<td>Displays the specified window, but does not activate it. If the window is minimized or maximized, then it remains in that state.</td>
</tr>
<tr>
<td>5</td>
<td>SW_SHOW</td>
<td>Activates and displays the specified window in its current size and position. This is similar to SW_SHOWNORMAL except that if the window is minimized or maximized, then it remains in that state.</td>
</tr>
<tr>
<td>6</td>
<td>SW_MINIMIZE</td>
<td>Minimizes the specified window and then activates the next open window.</td>
</tr>
<tr>
<td>7</td>
<td>SW_SHOWMINNOACTIVATE</td>
<td>Displays the specified window as a minimized window, but does not activate it.</td>
</tr>
<tr>
<td>8</td>
<td>SW_SHOWNA</td>
<td>Displays the specified window in its current size and position, but does not activate it. This is similar to SW_SHOWNOACTIVATE except that if the window is minimized or maximized, then it remains in that state.</td>
</tr>
<tr>
<td>Value</td>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>9</td>
<td>SW_RESTORE</td>
<td>Activates and displays the specified window. If the window is minimized or maximized, then it is restored to its original size and position. You should use this command when restoring a minimized window.</td>
</tr>
</tbody>
</table>

This is an optional parameter. If no value is specified, then the default command is \texttt{SW\_RESTORE}.

**optNumTimeout**

The timeout period (in milliseconds) for the function to be successfully executed. If, for whatever reason, the function is not executed in this period, then it is aborted.

This is an optional parameter. If no value is specified, then the default timeout is five seconds (or 5000 milliseconds).

**Returned value**

This function will return one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ERROR: The specified application window was not activated or otherwise did not respond within the timeout period.</td>
</tr>
<tr>
<td>1</td>
<td>SUCCESS: The specified application window was successfully activated.</td>
</tr>
</tbody>
</table>

**Notes**

\texttt{AppActivate} is similar to the function \texttt{ShowWindow} in the Microsoft Windows API, and it allows many of the same options. For more information, please refer to the Windows API documentation.

**Examples**

Show the Microsoft Word document named \texttt{test.doc}:

\begin{verbatim}
AppActivate( "test.doc — Microsoft Word", 5 )
\end{verbatim}

**AppIsRunning**

\texttt{AppIsRunning} is a built-in scripting function that verifies another application window is open and running.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppIsRunning</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

\begin{verbatim}
AppIsRunning( \texttt{strAppTitle}, \texttt{optNumTimeout} )
\end{verbatim}

**strAppTitle**

The full title (as shown in the title bar) of the application window.

**optNumTimeout**

The timeout period (in milliseconds) for the function to be successfully executed. If, for whatever reason, the function is not executed in this period, then it is aborted.

This is an optional parameter. If no value is specified, then the default timeout is five seconds (or 5000 milliseconds).

**Returned value**

This function will return one of the following values:
## AppIsRunning

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ERROR: The specified application window is not open or otherwise did not respond within the timeout period.</td>
</tr>
<tr>
<td>1</td>
<td>SUCCESS: The specified application window is open and running.</td>
</tr>
</tbody>
</table>

### Notes

AppIsRunning is similar to the function IsWindow in the Microsoft Windows API. For more information, please refer to the Windows API documentation.

### Examples

Verify the Microsoft Word document named test.doc is open and running:

```powershell
AppIsRunning( "test.doc — Microsoft Word" )
```

## AppPostMessage

AppPostMessage is a built-in scripting function that sends a Windows system message to another application window.

### Syntax

```powershell
AppPostMessage(strAppTitle, strMessage, numWParam, numLParam|, optNumTimeout)
```

- **strAppTitle**
  The full title (as shown in the title bar) of the application window.

- **strMessage**
  The name or code of the system message.

```powershell
Note: The CLOSE, MINIMIZE, MAXIMIZE and RESTORE messages can be given as string values enclosed in quotes. All other message codes must be given as numeric values.
```

- **numWParam**
  Additional message-specific information.

- **numLParam**
  Additional message-specific information.

- **optNumTimeout**
  The timeout period (in milliseconds) for the function to be successfully executed. If, for whatever reason, the function is not executed in this period, then it is aborted.

  This is an optional parameter. If no value is specified, then the default timeout is five seconds (or 5000 milliseconds).

### Returned value

This function will return one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ERROR: The system message was not sent, or the specified application window did not respond, within the timeout period.</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUCCESS: The system message was successfully sent.</td>
</tr>
</tbody>
</table>

### Notes

`AppPostMessage` is similar to the function `PostMessage` in the Microsoft Windows API, and it allows many of the same options. For more information, including a list of available system messages, please refer to the Windows API documentation.

### Examples

Close the Calculator application:

```plaintext
AppPostMessage( "Calculator", "CLOSE", 3, 1 )
```

#### AppSendKeys

Sends keyboard commands to the active application.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AppSendKeys</code></td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

#### Syntax

```plaintext
AppSendKeys( strKeys1, strKeys2, ... , strKeysN )
```

**strKeys (1-N)**

String tags containing the keyboard commands to be used.

#### Returned value

No returned values.

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>AppSendKeys( &quot;S&quot;, &quot;t&quot;, &quot;u&quot;, &quot;d&quot;, &quot;i&quot;, &quot;o&quot;, &quot;&lt;ENTER&gt;&quot; )</code></td>
</tr>
<tr>
<td></td>
<td><code>AppSendKeys( &quot;&lt;Alt&gt;F&quot; )</code></td>
</tr>
</tbody>
</table>

**Note:** You can specify `<ALT>`, `<CTRL>`, or `<SHIFT>` in the text to send a code equal to the Alt, Ctrl, or Shift keyboard commands. To send the `<` character, specify `<<` in the text.

#### CleanReadQueue

Removes all reading messages from the communications module.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CleanReadQueue</code></td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

#### Syntax

```plaintext
CleanReadQueue()
```

This function takes no parameters.
Returned value
No returned values.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CleanReadQueue()</td>
</tr>
</tbody>
</table>

Note: You should not use this function in new projects, but it is still valid for projects built using earlier versions of IWS.

CloseSplashWindow
Closes the IWS splash screen.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CloseSplashWindow</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

Syntax
CloseSplashWindow()
This function takes no parameters.

Returned value
No returned values.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CloseSplashWindow()</td>
</tr>
</tbody>
</table>

DisableMath
DisableMath is a built-in scripting function that pauses the execution of all Math worksheets.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisableMath</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Syntax
DisableMath()
This function takes no parameters.

Returned value
There are no returned values for this function.

Notes
To resume the execution of Math worksheets, call the EnableMath function.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DisableMath()</td>
</tr>
</tbody>
</table>
**EnableMath**

EnableMath is a built-in scripting function that resumes the execution of all Math worksheets.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnableMath</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

EnableMath()

This function takes no parameters.

**Returned value**

There are no returned values for this function.

**Notes**

In most cases, execution was paused by calling the `DisableMath` function.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EnableMath()</td>
</tr>
</tbody>
</table>

**EndTask**

The function EndTask stops a specified run-time module that is currently running.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndTask</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Syntax**

EndTask(*strTask*)

**EndTask**(*strTask*)

**strTask**

The name of the task to stop (must be one of the following):

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGTask</td>
<td>Background Tasks</td>
</tr>
<tr>
<td>Viewer</td>
<td>Viewer</td>
</tr>
<tr>
<td>DBSpy</td>
<td>Database Spy</td>
</tr>
<tr>
<td>LogWin</td>
<td>LogWin</td>
</tr>
<tr>
<td>Driverdrivername</td>
<td>Driver</td>
</tr>
<tr>
<td>UniDDECl</td>
<td>DDE Client</td>
</tr>
<tr>
<td>UniNDDE</td>
<td>DDE Server</td>
</tr>
<tr>
<td>UniODBC</td>
<td>ODBC</td>
</tr>
<tr>
<td>TCPSserver</td>
<td>TCP/IP Server (i.e., the project runtime server)</td>
</tr>
<tr>
<td>TCPClient</td>
<td>TCP/IP Client</td>
</tr>
<tr>
<td>OPCCClient</td>
<td>OPC Client</td>
</tr>
</tbody>
</table>
Returned value
No returned values.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EndTask(&quot;Viewer&quot;)</td>
</tr>
</tbody>
</table>

**Note:** To close a driver, you must use the following syntax:

```plaintext
EndTask("Driverdrivername")
```

Where `drivername` is the name of the driver's *.dll* file. For example:

```plaintext
EndTask("DriverMODBU")
```

**ExitWindows**

Exits the Windows operating system in a specified manner.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExitWindows</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ExitWindows( numExitCode )`

`numExitCode`

A numeric code specifying how Windows should be exited:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Restart</td>
</tr>
<tr>
<td>1</td>
<td>Log off</td>
</tr>
<tr>
<td>2</td>
<td>Shut down</td>
</tr>
</tbody>
</table>

Returned value
No returned values.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ExitWindows( 1 )</td>
</tr>
</tbody>
</table>

**IsScreenOpen**

The function `IsScreenOpen` that a project screen is open on a client.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsScreenOpen</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

IsScreenOpen(strScreen, optNumID)

IsScreenOpen(strScreen{   | , optNumID })

strScreen
    The name of the project screen to be verified.

optNumID
    The specific ID number of the screen. (This number is assigned when the screen is opened using the function Open.)
    This parameter is optional; if no value is specified, the default value is 0.

Returned value
This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Screen is not open.</td>
</tr>
<tr>
<td>1</td>
<td>Screen is open.</td>
</tr>
</tbody>
</table>

Notes
If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

Examples
Is the screen named "main" open?

    IsScreenOpen("main")

Is the screen named "main" with ID 10 open?

    IsScreenOpen("main",10)

IsTaskRunning
Verifies that a background task is running.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsTaskRunning</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

Syntax

IsTaskRunning(strTask)

strTask
    The name of the task to be verified.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Task status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Task is not running.</td>
</tr>
<tr>
<td>1</td>
<td>Task is running.</td>
</tr>
</tbody>
</table>
Examples
Is the project viewer (Viewer.exe) running?

IsTaskRunning( "Viewer" )

IsViewerInFocus
Verifies that the project viewer (Viewer.exe) is in focus on the display.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsViewerInFocus</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax
IsViewerInFocus()
This function takes no parameters.

Returned value

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Viewer is not in focus.</td>
</tr>
<tr>
<td>1</td>
<td>Viewer is in focus.</td>
</tr>
</tbody>
</table>

Examples

IsViewerInFocus()

KeyPad
Displays a Virtual Keyboard to prompt the runtime user to enter a tag value.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPad</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

KeyPad( "strTagName", optStrKeyboardName, optNumIsPassword, optStrHint, optNumMin, optNumMax, optNumESign, optStrConfirmation )

strTagName
The name of the tag to which the Virtual Keyboard will write.
This name must be enclosed in quotes; if it is not, then the project will try to get the contents of the tag.

optStrKeyboardName
Type of Virtual Keyboard that will be launched (e.g., AlphaNumeric, EnhKeypad, or Keypad). If this parameter is omitted, then the default Virtual Keyboard will be launched.

optNumIsPassword
If this parameter is set with any value different from 0 (zero), the characters typed in the Virtual Keyboard will appear as asterisks (*). This option is useful when the user is typing a password.

optStrHint
The value specified for this parameter is displayed in the title bar of the Virtual Keyboard window, if the Show Hint option is selected in the project settings.
**optNumMin, optNumMax**

Minimum and maximum numeric values for the Tag when using the Keypad keyboard type, if the *Show MIN/MAX fields* option is selected in the project settings. (These values are ignored for all other keyboard types.) These parameters are optional, but you must specify both to have them implemented. If you specify only one parameter — for example, Min but not Max — then it will be ignored.

**optNumESign**

If this parameter is set with any value different from 0 (zero), then the user will be prompted to log on to complete the input and the action will be noted in the log.

**optStrConfirmation**

The value specified for this parameter is displayed in a confirmation dialog that the user must acknowledge to complete the input.

> **Note:** Confirmation cannot be automated or bypassed; only an actual key press or mouse click by the user will acknowledge the dialog.

This is an optional parameter. If no value is specified, then no dialog is displayed.

### Returned value

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Error</td>
</tr>
<tr>
<td>2</td>
<td>Tag does not exist</td>
</tr>
<tr>
<td>3</td>
<td>Reentrant error, function is already executing</td>
</tr>
<tr>
<td>4</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>5</td>
<td>Internal error, contact Technical Support for more information</td>
</tr>
</tbody>
</table>

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>KeyPad( &quot;tagA&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>KeyPad( &quot;tagA&quot;, &quot;EnhKeypad&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>KeyPad( &quot;tagA&quot;, &quot;EnhKeypad&quot;, 1 )</td>
</tr>
<tr>
<td>Tag</td>
<td>KeyPad( &quot;tagA&quot;, &quot;EnhKeypad&quot;, 1, &quot;My Input&quot;, 0, 100 )</td>
</tr>
</tbody>
</table>

**LogOff**

This function logs off the current user and then logs on the default user (typically "guest").

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogOff</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

LogOff()

This function takes no parameters.

### Returned value

This function returns the following possible values:
### Examples

**LogOff()**

**LogOn**

This function either logs on a specified user or displays a *Log On* dialog.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogOn</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

LogOn(*optStrUsername*, *optStrPassword*)

LogOn({   | *optStrUsername*, *optStrPassword* })*

*optStrUsername*

The name of the user to log on.

*optStrPassword*

The specified user’s password.

*optStrUsername* and *optStrPassword* are optional parameters. If they are not specified, then the project will instead display a *Log On* dialog, to prompt the station’s current operator — whoever it is — to log on.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error (e.g., username or password is invalid) or cancellation.</td>
</tr>
<tr>
<td>1</td>
<td>Success.</td>
</tr>
</tbody>
</table>

**Notes**

The following limitations also apply to how the function can be used in project screens in Mobile Access:

- The function cannot be called from the sub-routine **Screen_OnClose** in the **Screen Script**; and
- You should avoid creating a script that both shows a dialog box (e.g., a message box that is shown using the VBScript function **MsgBox** or the IWS built-in function **ShowMessageBox**) and opens or closes a screen. The Mobile Access web interface can display only one screen at a time, and opening a new screen will automatically close the previous one. Depending on the order of execution, this can lead to unexpected behavior such as a dialog box being orphaned or not being shown at all.

**Examples**

Display a *Log On* dialog:

LogOn()
Log on username Albert with password EMC2:

```
LogOn( "Albert", "EMC2" )
```

Math

Executes the specified Math worksheet.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Syntax

```
Math( numWorksheet )
```

**numWorksheet**
The number of the math worksheet to be executed.

Returned value
No returned values.

Examples

```
<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Math( 6 )</td>
</tr>
</tbody>
</table>
```

⚠️ Caution: Running a Math worksheet from inside another module will pause that module until the Math worksheet finishes. Consequently, use this function only when absolutely necessary to avoid decreasing the performance of the other modules.

PostKey

The function PostKey posts virtual-key codes to the currently displayed project screen. In other words, it simulates keyboard input.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostKey</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

```
PostKey(numKeydownOrKeyup, numwParam, numlParam)
```

```
PostKey({numKeydownOrKeyup | 0 | 1}, numwParam, numlParam)
```

**numKeydownOrKeyup**
A numerical flag with the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>KeyDown event</td>
</tr>
<tr>
<td>1</td>
<td>KeyUp event</td>
</tr>
</tbody>
</table>

In most cases, this value should be 0 to indicate that the key is being pressed.
numwParam
The virtual-key code to be posted, in hexadecimal format.
For a list of codes, go to: msdn.microsoft.com/library/dd375731.aspx

numlParam
Additional message information.
In most cases, this value should be 0 to indicate that there is no additional information.

Notes
If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

Returned value
This function does not return any value.

Examples
Post the virtual-key code for the Home key:

PostKey(0,0x24,0)

Recipe
Executes the specified Recipe worksheet.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recipe</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Syntax
Recipe( strFunction )

strFunction
String tag specifying the operation to be performed and the Recipe worksheet to be used in the [Operation]:[Recipe sheet] format.

Operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td>Saves data to a data file.</td>
</tr>
<tr>
<td>Load</td>
<td>Loads data from a data file.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes a data file.</td>
</tr>
<tr>
<td>Init</td>
<td>Initializes a data file with a value of 0 in all the tags.</td>
</tr>
</tbody>
</table>

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>If the tag is numeric</td>
</tr>
<tr>
<td>2</td>
<td>Expression does not contain &quot;=&quot;</td>
</tr>
<tr>
<td>3</td>
<td>Previous command to the invalid &quot;=&quot;</td>
</tr>
</tbody>
</table>
4 Task not found by the system
5 Disk error

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>Recipe( &quot;Save:Recipe1&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>Recipe( &quot;Load:Recipe5&quot; )</td>
</tr>
</tbody>
</table>

⚠️ **Note:**

- You must be running the Background Task to execute the recipe functions.
- When this function is called on a Thin Client, the command is sent to the Server (via TCP/IP) and the Recipe task on the Server executes the command. Therefore, be aware that tags configured with a Scope of Local rather than Server will still be updated on the Server.

**Report**

Executes the specified Report worksheet and sends the output to hard disk, printer, or PDF.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
Report( "strFunction" , optNumOrientation )
```

**strFunction**

String specifying the operation to perform and the Report worksheet to output, using the syntax "Operation:Report worksheet", where...

- **Operation** is either Disk (saves data file to hard disk), Prn (sends report to default printer), or Pdf (generates a PDF file of the report); and
- **Report worksheet** is the name of the Report worksheet file (*.rep) to output.

**optNumOrientation**

Set the paper orientation as follows: 0 (default) is Portrait, 1 is Landscape. This parameter is ignored if the **Operation** is other than Prn.

⚠️ **Note:** Some features of this function are not supported when running the project on a Windows Embedded device: it cannot generate PDFs; it cannot change paper orientation using the **optNumOrientation** parameter; and it does not support Report worksheets in RTF format.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td><strong>strFunction</strong> is configured with a numeric value (invalid)</td>
</tr>
<tr>
<td>2</td>
<td><strong>strFunction</strong> does not contain &quot;::&quot; (invalid)</td>
</tr>
<tr>
<td>3</td>
<td><strong>strFunction</strong> contains an invalid output type before the &quot;:&quot;</td>
</tr>
<tr>
<td>4</td>
<td>Background Task is not running (see Tip below)</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Disk error (e.g., disk full, read-only file cannot be overwritten, or invalid path)</td>
</tr>
<tr>
<td>6</td>
<td>Specified Report worksheet does not exist</td>
</tr>
</tbody>
</table>

**Tip:** The Background Task must be running in order to execute this function. Otherwise, the operation will not be executed and the function will return the value 4 indicating error. For more information, see Execution Tasks.

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Report( &quot;Disk:Report1.rep&quot; )</td>
</tr>
<tr>
<td></td>
<td>Report( &quot;Prn:Report2.rep&quot;, 0 )</td>
</tr>
<tr>
<td></td>
<td>Report( &quot;Prn:Report3.rep&quot;, 1 )</td>
</tr>
<tr>
<td></td>
<td>Report( &quot;Pdf:Report1.rep&quot; )</td>
</tr>
</tbody>
</table>

### RunGlobalProcedureAsync

This function executes a global procedure asynchronously, in its own thread, so that it does not slow down or interfere with other running scripts. The procedure is run on the project server, but it can be called by any local or remote client.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunGlobalProcedureAsync</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported; see note</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

RunGlobalProcedureAsync(strProcedureName{ | , optStrArgument1,..., optStrArgumentN})

**strProcedureName**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder) to run asynchronously.

**optStrArgument1, ..., optStrArgumentN**

Values that are passed to the procedure’s parameters. Arguments must be passed as strings.

**Returned value**

If the procedure is successfully executed, then this function will return a thread ID that can be used with the RunGlobalProcedureAsyncGetStatus function. Otherwise, this function will return an error code:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Function is not supported on Viewer / Web Thin Client.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters. You must specify at least the procedure name.</td>
</tr>
<tr>
<td>-3</td>
<td>Maximum number of threads exceeded. See note.</td>
</tr>
<tr>
<td>-4</td>
<td>Failed to compile VBScript parameters for execution.</td>
</tr>
<tr>
<td>-5</td>
<td>Failed to start the thread execution.</td>
</tr>
</tbody>
</table>
Value | Description
--- | ---
-100 | Internal error. Please contact technical support.

**Notes**

It is very important to note that this function can only be called by background tasks (e.g., Math, Script, Scheduler) on the project server. It cannot be directly called from any project client, even if the project client is running on the same workstation or device as the project server, because the client process is single-threaded. To indirectly call the function from a project client, configure a Math or Script worksheet to execute on a tag/expression trigger, and then configure a project screen to activate the trigger when needed. For example, configure the worksheet to execute when the value of `MyTag` is 1, and then configure a Button screen object to toggle the value of `MyTag` when pressed.

Also, the maximum number of VBScript threads that can be executed asynchronously is configured by manually editing the project file (i.e., `projectname.APP`) to change the following setting:

```
[Script]
MaxAsyncThreads=8
```

The default number of threads is 8, but the only real limit is determined by the available system resources. Increasing the number of threads may decrease runtime performance.

**Examples**

Given the following procedure that is defined in the Procedures folder...

```vbnet
Function AddMe(intNumber)
    If intNumber >= 6 Then
        AddMe = 0
    Else
        AddMe = intNumber + 2
    End If
End Function
```

...the procedure is run by calling the `RunGlobalProcedureAsync` function...

```vbnet
RunGlobalProcedureAsync( "AddMe", "2" )
```

...and the function returns a thread ID that can be used with the `RunGlobalProcedureAsyncGetStatus` function.

**RunGlobalProcedureAsyncGetStatus**

This function gets the status of one or more global procedures that were run asynchronously by calling the `RunGlobalProcedureAsync` function. Each procedure is run in its own thread, so that it does not slow down or interfere with other threads.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunGlobalProcedureAsyncGetStatus()</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported; see note</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```vbnet
RunGlobalProcedureAsyncGetStatus({ | optNumThreadID | "optTagThreadIDs","optTagStatus","optTagParameters" })
```

```vbnet
RunGlobalProcedureAsyncGetStatus( )
RunGlobalProcedureAsyncGetStatus( optNumThreadID )
```
RunGlobalProcedureAsyncGetStatus( "optTagThreadIDs", "optTagStatus", "optTagParameters" )

**optNumThreadID**

The thread ID returned by the RunGlobalProcedureAsync function, if the procedure was successfully executed.

**optTagThreadIDs**

The name of an Array tag that will receive the thread IDs of all currently running and recently completed threads.

![Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.]

**optTagStatus**

The name of an Array tag that will receive the statuses of all currently running and recently completed threads.

![Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.]

**optTagParameters**

The name of an Array tag that will receive the parameters of all currently running and recently completed threads.

![Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.]

**Returned value**

If the function succeeds, then the possible returned values depend on how the function was called:

- If the function was called with no parameters...

  RunGlobalProcedureAsyncGetStatus( )

  ...then the returned value is the total number of threads that are currently running.

- If the function was called with only the optNumThreadID parameter...

  RunGlobalProcedureAsyncGetStatus( optNumThreadID )

  ...then the returned value is either 0, indicating that the thread is still running, or the value that was returned by the procedure.

- If the function was called with the Array tags...

  RunGlobalProcedureAsyncGetStatus( "optTagThreadIDs", "optTagStatus", "optTagParameters" )

  ...then the tags will receive the appropriate values for all currently running and recently completed threads.

If the function fails, then it returns one of the following errors:
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Function is not supported on Viewer / Web Thin Client.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid thread ID.</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid optTagThreadIDs.</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid optTagStatus.</td>
</tr>
<tr>
<td>-5</td>
<td>Invalid optTagParameters.</td>
</tr>
<tr>
<td>-100</td>
<td>Internal error. Please contact technical support.</td>
</tr>
</tbody>
</table>

### Notes

It is very important to note that this function can only be called by background tasks (e.g., Math, Script, Scheduler) on the project server. It cannot be directly called from any project client, even if the project client is running on the same workstation or device as the project server, because the client process is single-threaded. To indirectly call the function from a project client, configure a Math or Script worksheet to execute on a tag/expression trigger, and then configure a project screen to activate the trigger when needed. For example, configure the worksheet to execute when the value of `MyTag` is 1, and then configure a Button screen object to toggle the value of `MyTag` when pressed.

Also, when the call to `RunGlobalProcedureAsync` succeeds, it returns an ID for the thread created and starts running the procedure in that thread. The status of the thread is stored in an internal buffer and can be retrieved using the `RunGlobalProcedureAsyncGetStatus` function. The buffer gets cleared when:

- The `RunGlobalProcedureAsyncGetStatus` function has been called and the thread status is different from 0 (thread is running); or
- The maximum buffer size has been exceeded, the thread is no longer running, and a call to start a new thread has been made.

### RunGlobalProcedureOnFalse

This function runs a global procedure when the value of a specified project tag or expression becomes FALSE.

#### Syntax

```plaintext
RunGlobalProcedureOnFalse("strCondition", strProcedureOnFalse)
```

**strCondition**

A project tag or expression that can be evaluated as either FALSE (zero) or TRUE (non-zero).

**Note:** The condition should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the condition instead.

**strProcedureOnFalse**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder of your project) to run when the value of the specified tag/expression becomes FALSE.

### Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

## Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

### Notes

Once this function is called, it is continuously executed by the project runtime client (i.e., the Viewer) until either it or the project runtime server is stopped. That means every time the value of the condition becomes FALSE, the procedure is run. However, the procedure is run only once when the value of the condition becomes FALSE; it is not repeatedly run while the value of the condition is FALSE. For the procedure to run again, the value of the condition must become TRUE and then FALSE again.

Also, the function can be called more than once, so that the same procedure can be run by different triggers. The project runtime client manages the execution of all instances of the function.

The procedure is run on the client where this function was called. To run a procedure on the project server, use the function `RunGlobalProcedureOnServer`.

The value of `strCondition` is passed to the procedure as an argument, so the procedure should be written to receive it. For example:

```vbnet
Sub UsingOnFalse(strCondition)
    
End Sub
```

Please note that you do not actually have to use the argument in your procedure, only that you should write the procedure to receive it.

No other arguments can be passed to the procedure.

To ensure compatibility with previous versions of InduSoft Web Studio, passing the argument is disabled by default in existing projects and enabled by default in new projects. To change this for your project, open your project file (`projectname.APP`) in a text editor and then edit the following property:

```text
[Options]
EnableTagNameOnRunGlobalProcedureOnTag=<0 (disabled) / 1 (enabled)>
```

### Examples

When the value of `TagOnFalse` becomes FALSE, run the procedure `UsingOnFalse`:

```vbnet
RunGlobalProcedureOnFalse("TagOnFalse","UsingOnFalse")
```

### RunGlobalProcedureOnServer

The function `RunGlobalProcedureOnServer` runs a specified VBScript procedure, as defined in the Procedures folder in the Project Explorer. The procedure is run on the project runtime server, but it can be triggered by any client that calls this function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunGlobalProcedureOnServer</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

### Syntax

```vbnet
RunGlobalProcedureOnServer(strProcedureName, optStrArgument1, ..., optStrArgumentN)
```

```vbnet
RunGlobalProcedureOnServer(strProcedureName{ | , optStrArgument1, ..., optStrArgumentN})
```

`strProcedureName`
The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder) to run on the project runtime server.

`optStrArgument1,...,optStrArgumentN`

Values that are passed to the procedure's parameters. Arguments must be passed as strings, but the procedure will interpret them as the correct data types. For more information, see "Examples" below.

**Returned value**

This function returns whatever value that is returned by the procedure.

**Notes**

If the option **Enforce Web functionality equivalence** is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

Also, you cannot call this function in a procedure that was itself run by calling the function `RunGlobalProcedureOnServer`. If you attempt to do so, then the function will return an error. This is to prevent a possible memory leak caused by nested or recursive function calls.

You can still call other procedures directly, as you normally would in VBScript, or you can use the function `Eval` in VBScript to dynamically determine the procedure you are calling.

**Examples**

Given the following procedure that is defined in the Procedures folder...

```vbnet
Function AddMe(intNumber)
    If intNumber >= 6 Then
        AddMe = 0
    Else
        AddMe = intNumber + 2
    End If
End Function
```

...the procedure is run by calling the function `RunGlobalProcedureOnServer`...

```vbnet
RunGlobalProcedureOnServer("AddMe","2")
```

...and it returns a value of 4.

**RunGlobalProcedureOnTrigger**

This function runs a global procedure when the value or quality of a specified tag changes.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunGlobalProcedureOnTrigger</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```vbnet
RunGlobalProcedureOnTrigger("strTagName",strProcedureOnTrigger)
```

`strTagName`

The name of a project tag.
**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**strProcedureOnTrigger**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder of your project) to run when the value or quality of the specified tag changes.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Notes**

Once this function is called, it is continuously executed by the project runtime client (i.e., the Viewer) until either it or the project runtime server is stopped. That means every time the value or quality of the specified tag changes, the procedure is run. Also, the function can be called more than once, so that the same procedure can be run by different triggers. The project runtime client manages the execution of all instances of the function.

**Tip:** The procedure is run on the client where the function was called. To run a procedure on the server, use the function `RunGlobalProcedureOnServer`.

The value or quality of the specified tag is passed to the procedure as an argument, so the procedure should be written to receive it. For example:

```
Sub MyProcedure(strTrigger)
  .
  .
End Sub
```

In practice, this means either...

```
strTrigger = $tagname->Value
```

...or...

```
strTrigger = $tagname->Quality
```

...depending on which one changed. You can then use the value or quality in your procedure.

Please note that you do not actually have to use the argument, only that you should write the procedure to receive it.

No other arguments can be passed to the procedure.

To ensure compatibility with previous versions of InduSoft Web Studio, passing the argument is disabled by default in existing projects and enabled by default in new projects. To change this for your project, open your project file (`projectname.APP`) in a text editor and then edit the following property:

```
[Options]
EnableTagNameOnRunGlobalProcedureOnTag=<0 (disabled) / 1 (enabled)>
```
Examples
When the value or quality of the tag MyInteger changes, run the procedure MyProcedure:

RunGlobalProcedureOnTrigger("MyInteger","MyProcedure")

The equivalent of MyInteger->Value or MyInteger->Quality, depending on which one changed, is passed to the procedure as an argument.

RunGlobalProcedureOnTrue
This function runs a global procedure when the value of a specified project tag or expression becomes TRUE.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunGlobalProcedureOnTrue</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

RunGlobalProcedureOnTrue("strCondition",strProcedureOnTrue)

strCondition
A project tag or expression that can be evaluated as either FALSE (zero) or TRUE (non-zero).

\[\text{Note}: \text{The condition should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the condition instead.}\]

strProcedureOnTrue
The name of the procedure (i.e., a VBScript function or sub-routine defined in the Procedures folder of your project) to run when the value of the specified tag/expression becomes TRUE.

Returned value
This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

Notes
Once this function is called, it is continuously executed by the project runtime client (i.e., the Viewer) until either it or the project runtime server is stopped. That means every time the value of the condition becomes TRUE, the procedure is run. However, the procedure is run only once when the value of the condition becomes TRUE; it is not repeatedly run while the value of the condition is TRUE. For the procedure to run again, the value of the condition must become FALSE and then TRUE again.

Also, the function can be called more than once, so that the same procedure can be run by different triggers. The project runtime client manages the execution of all instances of the function.

The procedure is run on the client where this function was called. To run a procedure on the project server, use the function RunGlobalProcedureOnServer.

The value of strCondition is passed to the procedure as an argument, so the procedure should be written to receive it. For example:

\[
\text{Sub UsingOnTrue(strCondition)}
\]
Appendix: Built-in Scripting Language

End Sub

Please note that you do not actually have to use the argument in your procedure, only that you should write the procedure to receive it.

No other arguments can be passed to the procedure.

To ensure compatibility with previous versions of InduSoft Web Studio, passing the argument is disabled by default in existing projects and enabled by default in new projects. To change this for your project, open your project file (projectname.APP) in a text editor and then edit the following property:

[Options]
EnableTagNameOnRunGlobalProcedureOnTag=<0 (disabled) / 1 (enabled)>

Examples
When the value of TagOnTrue becomes TRUE, run the procedure UsingOnTrue:

RunGlobalProcedureOnTrue("TagOnTrue","UsingOnTrue")

RunVBScript
Executes a statement in VBScript language.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunVBScript</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

RunVBScript ( strScript, "optTagReturnError" )

strScript
Script statement that must be executed by the function.

optTagReturnError
Name of the tag that will receive the error (if any) generated by the statement (e.g., "Division by zero"). The tag name must be configured between double-quotes and it must be a String tag.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagResult</td>
<td>RunVBScript(&quot;MsgBox(Time)&quot;) // Executes the MsgBox function from VBScript and displays the current time.</td>
</tr>
<tr>
<td></td>
<td>RunVBScript(TagStatement) // Executes the statement configured in the value of the string tag TagStatement</td>
</tr>
<tr>
<td></td>
<td>RunVBScript(&quot;$TagC=$TagA/$TagB&quot;, &quot;TagError&quot;) // Writes in TagC the result of TagA divided by TagB. The error generated by the operation (if any) is written to the string tag TagError (e.g., &quot;Division by zero&quot;).</td>
</tr>
</tbody>
</table>
Tip: This function is useful to execute VBScript statements from interfaces that support the built-in language only (e.g., Scheduler groups). You can also call VBScript functions created in the Global Procedures.

Note: The runtime station must support the VBScript statements configured in this function in order to execute them.

SecureViewerReload

SecureViewerReload is a built-in scripting function that closes the Secure Viewer program and then reloads it with a new configuration file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecureViewerReload</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Not Supported</td>
<td>Not Supported</td>
<td>Secure Viewer only</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

SecureViewerReload(strFileName)

strFileName
The file path of an INI file (*.ini) that describes the new configuration. (If the file is located in the same folder as Viewer.exe, then only the file name is needed.) The file should be structured the same and contain all of the same settings as the default configuration file (Viewer.ini).

This parameter must specify either the name of a String tag or a text string enclosed in quotes.

Returned value
This function does not return any value.

Examples

SecureViewerReload( configFile1 )

SecureViewerReload( "C:\Program Files\Secure Viewer\Bin\Config1.ini" )

SendKeyObject

The function SendKeyObject sends a key event code to objects in the currently displayed project screen. You can use this function to trigger Command animations on those objects.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendKeyObject</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

SendKeyObject(numEvent, strMainKey, numShift, numCtrl, numAlt, strTargetScreen, optNumID)

numEvent
A numeric value that indicates the type of key event to send to the screen. The following values are accepted:
Appendix: Built-in Scripting Language

Value | Description
--- | ---
0 | On Down
1 | On While
2 | On Up

For more information about these key events, see Command animation on page 234.

**Note:** If the "On While" event is specified, the "On While" script on the Command animation is executed just once for each time this function is executed. It is not continuously executed as if the key is pressed and held down, because this function does not have a parameter for specifying duration. If you want to cause that sort of behavior, you can include this function in an appropriately configured FOR loop.

`strMainKey`

The key to be sent to the screen. The following values are accepted:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;A&quot; ... &quot;Z&quot;</td>
<td>alphabetic characters A through Z</td>
</tr>
<tr>
<td>&quot;+&quot;</td>
<td>plus symbol</td>
</tr>
<tr>
<td>&quot;+&quot;</td>
<td>minus symbol</td>
</tr>
<tr>
<td>&quot;×&quot;</td>
<td>multiply symbol</td>
</tr>
<tr>
<td>&quot;/&quot;</td>
<td>divide symbol</td>
</tr>
<tr>
<td>&quot;LEFT&quot;</td>
<td>left arrow (←)</td>
</tr>
<tr>
<td>&quot;UP&quot;</td>
<td>up arrow (↑)</td>
</tr>
<tr>
<td>&quot;RIGHT&quot;</td>
<td>right arrow (→)</td>
</tr>
<tr>
<td>&quot;DOWN&quot;</td>
<td>down arrow (↓)</td>
</tr>
<tr>
<td>&quot;HOME&quot;</td>
<td>Home key</td>
</tr>
<tr>
<td>&quot;END&quot;</td>
<td>End key</td>
</tr>
<tr>
<td>&quot;PAGEUP&quot;</td>
<td>Page Up key</td>
</tr>
<tr>
<td>&quot;PAGEDOWN&quot;</td>
<td>Page Down key</td>
</tr>
<tr>
<td>&quot;INSERT&quot;</td>
<td>Insert key</td>
</tr>
<tr>
<td>&quot;DELETE&quot;</td>
<td>Delete key</td>
</tr>
<tr>
<td>&quot;SPACE&quot;</td>
<td>Space key</td>
</tr>
<tr>
<td>&quot;RETURN&quot;</td>
<td>Return key</td>
</tr>
<tr>
<td>&quot;BACKSPACE&quot;</td>
<td>Backspace key (if different from Delete key)</td>
</tr>
<tr>
<td>&quot;ESCAPE&quot;</td>
<td>Escape key</td>
</tr>
<tr>
<td>&quot;F1&quot; ... &quot;F20&quot;</td>
<td>function keys F1 through F20</td>
</tr>
</tbody>
</table>

The key must be enclosed in quotes, as shown.

`numShift`

A numeric value that indicates whether to include Shift with the specified key (e.g., Shift+R). The following values are accepted: 0 is no, 1 is yes.

This parameter is optional; if no value is specified, the default value is 0.
**numCtrl**

A numeric value that indicates whether to include Ctrl with the specified key (e.g., Ctrl+R). The following values are accepted: 0 is no, 1 is yes.

This parameter is optional; if no value is specified, the default value is 0.

**numAlt**

A numeric value that indicates whether to include Alt with the specified key (e.g., Alt+R). The following values are accepted: 0 is no, 1 is yes.

This parameter is optional; if no value is specified, the default value is 0.

**strTargetScreen**

The name of the screen that will receive the key event code.

This parameter is optional; if no value is specified, the currently active screen is used.

**optNumID**

The specific ID number of the screen. (The ID number is assigned when the screen is opened with the function Open.)

This parameter is optional; if no value is specified, the default value is 0.

**Returned value**

This function does not return any value.

**Notes**

If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

Also, the parameters numShift, numCtrl, numAlt and strTargetScreen are all optional, but if you configure one of them, you must configure the others as well.

**Examples**

Send R to the currently active screen:

```
SendKeyObject(0,"R")
```

Send Ctrl+Shift+R to the screen named "main" with ID 10:

```
SendKeyObject(0,"R",1,1,0,"main",10)
```

**SetAppPath**

Sets the new file path for the project folder. After this function is executed, IWS will look for all of the project files (i.e., screens, alarms, trends, database, events) in this folder.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetAppPath</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

SetAppPath(strPath)

strPath

The file path.
### Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Failed to set path.</td>
</tr>
<tr>
<td>1</td>
<td>Succeeded in setting path</td>
</tr>
</tbody>
</table>

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SetAppPath( &quot;C:\Studio&quot; )</td>
</tr>
</tbody>
</table>

> **Note:** If the computer is on a network, you can use the //IP address or host name/Path syntax to define a location on another node of the network.

### SetViewerInFocus

SetViewerInFocus is a built-in scripting function that moves the project viewer in front of all other open windows and then maximizes it to fill the display.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetViewerInFocus</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

### Syntax

SetViewerInFocus()

There are no parameters.

### Returned value

This function does not return any value.

### Notes

Beginning with Windows XP, system security features prevent program windows from moving themselves in front of others without user input. As such, when you call this function, the Viewer program will request the user's attention by blinking in the Windows taskbar. (Some anti-virus software may also flag this as suspicious behavior.) Only when the user selects the program will the program window move to the front.

To work around this, you must call SetViewerInFocus at least once in your project's Startup Script. Allow twenty seconds more for your project to finish starting up, and then after that, any additional calls of this function should work normally.

### Examples

SetViewerInFocus()

### SetViewerPos

Sets the height, width, and position of the project viewer or thin client.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetViewerPos</td>
<td>Module Activity</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

### Syntax

SetViewerPos( numLeft, numTop, optNumWidth, optNumHeight )
numLeft
A numeric flag that specifies the left-side position of the Viewer in pixels.

numTop
A numeric flag that specifies the top-side position of the Viewer in pixels.

optNumWidth
Optional numeric tag specifying the Viewer width in pixels.

optNumHeight
Optional numeric tag containing the Viewer height in pixels.

Returned value

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SetViewerPos( 50, 50, 640, 480 )</td>
</tr>
</tbody>
</table>

Note: When you omit the optional parameters (numWidth and numHeight), IWS gets size of the Viewer window from the project resolution.

ShutDown
This function stops all execution tasks and runtime modules, effectively shutting down the project.

Syntax

ShutDown()

Retrunned value
This function does not return any value.

Notes
This function only stops the project runtime server. It does not close the development environment on the server, if it happens to be open.

If this function is called from a screen object or script on a client, it only stops the Viewer module on that client. On Mobile Access, it returns the user to the logon screen.

Also, when this function is used in project screens in Mobile Access, it cannot be called from the sub-routine Screen_OnClose in the Screen Script.
**StartTask**

The function `StartTask` starts a specified run-time module that is not currently running.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartTask</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Syntax**

`StartTask(strTask)`

**StartTask**

**strTask**

The name of the task to start (must be one of the following tasks):

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGTask</td>
<td>Background Tasks</td>
</tr>
<tr>
<td>Viewer</td>
<td>Viewer</td>
</tr>
<tr>
<td>DBSpy</td>
<td>Database Spy</td>
</tr>
<tr>
<td>LogWin</td>
<td>LogWin</td>
</tr>
<tr>
<td>Driver</td>
<td>Driver</td>
</tr>
<tr>
<td>UniDDECl</td>
<td>DDE Client</td>
</tr>
<tr>
<td>UniNDDE</td>
<td>DDE Server</td>
</tr>
<tr>
<td>UniODBC</td>
<td>ODBC</td>
</tr>
<tr>
<td>TCPServer</td>
<td>TCP/IP Server (i.e., the project runtime server)</td>
</tr>
<tr>
<td>TCPClient</td>
<td>TCP/IP Client</td>
</tr>
<tr>
<td>OPCClient</td>
<td>OPC Client</td>
</tr>
</tbody>
</table>

**Returned value**

No returned values.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>StartTask(&quot;Viewer&quot;)</code></td>
</tr>
</tbody>
</table>

**Note:** The `StartTask("Driver")` function starts all drivers that are configured in the project. To start a specific driver, use the function `WinExec`.

**ViewerPostMessage**

The function `ViewerPostMessage` posts a Windows System Message to the specified project screen.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewerPostMessage</td>
<td>Module Activity</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

ViewerPostMessage(strScrTitle, numMessage, numwParam, numlParam, optNumID)

strScrTitle
The name of the screen to which the message will be posted.

numMessage
The number of the Windows System Message to be posted.

numwParam
Additional, message-specific information that is passed to wParam of the Windows System Message.

numlParam
Additional, message-specific information that is passed to lParam of the Windows System Message.

optNumID
The specific ID number of the screen. (This number is assigned when the screen is opened using the function Open.)

This parameter is optional; if no value is specified, the default value is 0.

Returned value
This function does not return any value.

Notes
This function emulates the PostMessage function in Microsoft Windows. For more information, including a complete list of available Windows System Messages, go to: msdn.microsoft.com/library/ms644944

If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

Examples
Send message 16 to the screen named "main" with ID 10:

ViewerPostMessage("main",16,3,1,10)

WinExec
WinExec is a built-in scripting function that executes a Windows command as if it was entered at the command prompt.

Syntax

WinExec(strCommand, |, { optNumState | 0 | 1 | 2 | 3 | 4 | 7 } |, { optNumSync | 0 | 1 }, "optTagReturnOrHandle" )

strCommand
The command to be executed.

**optNumState**

The initial state of the program (if any) that is run by the command:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Hides the program and gives control to another one.</td>
</tr>
<tr>
<td>1</td>
<td>Activates and displays the program.</td>
</tr>
<tr>
<td>2</td>
<td>Activates the program and displays it as an icon.</td>
</tr>
<tr>
<td>3</td>
<td>Activates the program and maximizes it.</td>
</tr>
<tr>
<td>4</td>
<td>Shows the program at its recent size. The program is still active.</td>
</tr>
<tr>
<td>7</td>
<td>Shows the program as an icon. The program is still active.</td>
</tr>
</tbody>
</table>

This is an optional parameter. If no value is specified, then the default value is 1.

*Note:* This parameter is not supported on Windows Embedded target systems; regardless of what value is actually specified, the function is executed as if the default value is specified.

**optNumSync**

A setting that specifies whether the command will execute synchronously or asynchronously:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Execute asynchronously; the function will return immediately.</td>
</tr>
<tr>
<td>1</td>
<td>Execute synchronously; the function will return when the command has finished executing.</td>
</tr>
</tbody>
</table>

This is an optional parameter. If no value is specified, then the default value is 0.

*Tip:* To verify that a command executed asynchronously has finished, use the optTagReturnOrHandle parameter below and the `WinExecIsRunning` function.

**optTagReturnOrHandle**

The name of a project tag that will store feedback about the execution of the command:

- If the command is executed asynchronously, then the tag will receive a handle that can used with the `WinExecIsRunning` function to determine whether the command is still running.
- If the command is executed synchronously, then the tag will receive the command’s exit code. (This is separate from the function’s own returned value.)

This is an optional parameter, but given its nature, there is no default value.

*Note:* The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Command was not executed successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Command was executed successfully.</td>
</tr>
</tbody>
</table>
Please note that this indicates only whether the command started its execution successfully, particularly if it is executed asynchronously. It does not indicate when or how the command finished its execution.

**Examples**

Start Notepad, and then immediately continue to the next line of the script:

```plaintext
WinExec( "C:\Windows\System32\notepad.exe", 4 )
```

Start MS Paint, and then immediately continue to the next line of the script:

```plaintext
WinExec( "C:\Windows\System32\mspaint.exe" )
```

Call a batch file, execute it in hidden mode, wait until it’s finished before continuing, and then store the exit code in the tag `return`:

```plaintext
WinExec( "CMD /C call C:\Temp\MyBatch.bat", 0, 1, "return" )
```

Call a VBScript file, execute it in hidden mode, and then immediately continue, storing the handle in the tag `handle`:

```plaintext
WinExec( "CMD /C call C:\Temp\MyScript.vbs", 0, 0, "handle" )
```

---

**Note:** Calling VBScript files is not supported on Windows Embedded target systems.

---

**WinExecIsRunning**

This function indicates whether a Windows program previously executed by the function `WinExec` is still running.

### Syntax

```plaintext
WinExecIsRunning( numHandle{   |,"optTagReturn" } )
```

- `numHandle`: The handle number that was stored in the tag configured in the parameter `optStrReturnOrHandle` of the function `WinExec`.
- `optTagReturn`: A tag that will receive the completion code returned by the previously executed program.

---

**Returned value**

This function returns the following possible values:
### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Program is still running.</td>
</tr>
<tr>
<td>0</td>
<td>Execution has been completed successfully (i.e., program has stopped).</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid handle number. Check the value of numHandle.</td>
</tr>
<tr>
<td>-3</td>
<td>Execution has been completed, but the return tag is invalid. Check the value of optTagReturn.</td>
</tr>
</tbody>
</table>

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyTag</td>
<td>WinExecIsRunning(numHandle)</td>
</tr>
<tr>
<td>MyTag</td>
<td>WinExecIsRunning(numHandle, &quot;return&quot;)</td>
</tr>
</tbody>
</table>
File functions

These functions are used to read from, write to, print, move, and delete external files.

**DeleteOlderFiles**

Deletes files that are older than a date matching the configured mask from the configured path.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteOlderFiles</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
DeleteOlderFiles( strPath, strMask, strDate )
```

**strPath**
The path to the files that will be deleted.

**strMask**
The mask of the files to be deleted.

**strDate**
The cut-off date. Any files older than this date will be deleted.

*Note:* This parameter must be configured using the date format specified for the project (such as MDY or DMY) with the appropriate separator (/ ; ; , and so forth.).

**Returned value**

Returns the number of files deleted.

**Examples**

```
Tag Name | Expression
---------|-------------
Tag      | DeleteOlderFiles( "C:\Studio\Project\HST\", ".hst", "04/12/2002" )
```

**DirCreate**

Creates the specified directory.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirCreate</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
DirCreate( strDirectory, optBooFullPath )
```

**strDirectory**
The name and file path of the directory to create.

**optBooEmptyOnly**
Optional flag. If omitted or if this parameter has the value 0, the directory is created only if all previous directories exist. If this parameter has the value different from 0, the full path specified in the strDirectory parameter is created.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Invalid parameters</td>
</tr>
<tr>
<td>0</td>
<td>Failed to create the directory (e.g., Drive does not exist.)</td>
</tr>
<tr>
<td>1</td>
<td>Directory created successfully.</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>DirCreate(&quot;C:\Studio\Temp&quot;)</td>
<td>// The Temp folder is created in the C:\Studio path (only if the C:\Studio path already exists).</td>
</tr>
<tr>
<td>Tag</td>
<td>DirCreate(&quot;C:\Studio\Temp&quot;,1)</td>
<td>// The C:\Studio\Temp full path is created.</td>
</tr>
</tbody>
</table>

DirDelete

Deletes the specified directory.

Syntax

DirDelete( strDirectory, optBooEmptyOnly )

strDirectory

The name and file path of the directory to delete.

Tip: This parameter supports wildcards ( * and ? ).

optBooEmptyOnly

Optional flag. If this parameter has a value of 1, then the directory is deleted only if it is empty. By default — that is, if the parameter is omitted or has a value of 0 — the directory is deleted whether it is empty or not.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Attempted to delete a non-empty directory when this action is not allowed (i.e., optBooEmptyOnly does not equal 0).</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid parameters.</td>
</tr>
<tr>
<td>0</td>
<td>Failed to delete the directory (i.e., directory does not exist).</td>
</tr>
<tr>
<td>1</td>
<td>Directory deleted successfully.</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>DirDelete( &quot;C:\Studio\Temp&quot; )</td>
<td>// The Temp folder from C:\Studio is deleted.</td>
</tr>
</tbody>
</table>
**DirDelete**

The Temp folder from C:\Studio is deleted only if it is empty.

**DirLength**

Returns the size of a specific directory.

**Syntax**

```
DirLength(strPath)
```

**strPath**

The path of the directory that will be checked.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Directory does not exist.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid parameters</td>
</tr>
<tr>
<td>n</td>
<td>Size (in bytes) of the files and sub-folders in the directory</td>
</tr>
</tbody>
</table>

**Notes**

⚠ **Caution:** This function executes synchronously, which means that the project pauses while it waits for the function to return. As such, if the specified directory is unusually large, then the project could be paused for several seconds while size of the directory is calculated.

**Examples**

```
Tag | Expression
---|-------------
Tag | DirLength("C:\Studio") // Returns the size (in bytes) of all files and sub-folders from C:\Studio.
```

**DirRename**

 Renames directories.

**Syntax**

```
DirRename(strPath, strDirectoryFrom, strDirectoryTo)
```

**strPath**

The path of the directory that will be renamed.

⚠ **Tip:** This function supports wildcard ( * and ? ).
The original name of the directory that will be renamed.

**strDirectoryTo**

The target name used to rename the original directory.

### Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Invalid parameters</td>
</tr>
<tr>
<td>0</td>
<td>Failed to rename the directory (e.g., strDirectoryFrom does not exist.)</td>
</tr>
<tr>
<td>1</td>
<td>Directory renamed successfully</td>
</tr>
</tbody>
</table>

### Examples

<table>
<thead>
<tr>
<th>Tag</th>
<th>Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td></td>
<td>DirRename(&quot;C:\Studio&quot;,&quot;Temp&quot;, &quot;New&quot;) // C:\Studio\Temp is renamed to C:\Studio\New.</td>
</tr>
</tbody>
</table>

### FileCopy

Copies the file(s) configured in the strSourceFile parameter to the path/file configured in the strTargetFile parameter.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileCopy</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

#### Syntax

FileCopy(strSourceFile, strTargetFile[, optNumTimeOut])

**strSourceFile**

The file path and name the file(s) to be copied.

**Tip:** This function supports wildcards (* and ?).

**strTargetFile**

The file path where the file(s) are to be copied.

**optNumTimeOut**

Numerical tag containing an integer to set the timeout time for the operation.

If you use the optNumTimeOut parameter, the function returns the value -1 after the specified timeout time and the scan continues. Though the function returns a -1, it does not cancel the copying procedure. Instead, it creates an internal process to finish the copying procedure.

### Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Timeout time expired.</td>
</tr>
<tr>
<td>0</td>
<td>Failed to copy file(s).</td>
</tr>
<tr>
<td>1</td>
<td>File(s) copied successfully.</td>
</tr>
</tbody>
</table>

### Notes

**Caution:** This function executes synchronously, which means that the project pauses while it waits for the function to return. As such, if the function is called to copy files from or to another volume across a slow network, then the project could be paused for long time.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>FileCopy( &quot;C:\Studio\Project\HST*.hst&quot;, &quot;C:\Temp\Hst&quot;, 1000 )</td>
</tr>
<tr>
<td>Tag</td>
<td>FileCopy( &quot;C:\Studio\Project opert.txt&quot;, &quot;C:\Temp\Tuesday_Report.txt&quot;, 500 )</td>
</tr>
</tbody>
</table>

**FileDelete**

Deletes the specified file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileDelete</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

FileDelete(\textit{strFile})

\textit{strFile}

The file path and name of the file to delete.

**Returned value**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Failed to delete file</td>
</tr>
<tr>
<td>Real</td>
<td>Returns the size of the file deleted</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>FileDelete( &quot;C:\Studio\Readme.txt&quot; )</td>
</tr>
</tbody>
</table>

**FileLength**

Gets the size of a file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileLength</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

FileLength(\textit{strFile})

\textit{strFile}

The file path and name of the file.

**Returned value**

Returns the size of the specified file in bytes.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>FileLength( &quot;C:\Readme.txt&quot; )</td>
</tr>
</tbody>
</table>
**FileReadFields**

The function **FileReadFields** reads values contained in the fields of a CSV file, and then it writes those values to a series of project tags or array elements.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileReadFields</td>
<td>cFile</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
FileReadFields(strFilename, numOffset, strStartTagName, numNumberOfTags)
```

**strFilename**
The file path and/or name of the CSV file. If the file is located inside your project folder, you can specify either just the file name or the file path relative to that folder. If the file is located outside your project folder, you must specify the absolute file path.

**numOffset**
The number of bytes to skip in the CSV file before reading values. To read from the start of the file, `numOffset` should be 0.

You can use this parameter to start reading from any position in the file, as long as you know how many bytes to skip. In most cases, you will simply take the value returned by the previous execution of this function (see "Returned value" below) and use it to resume reading where you previously stopped.

However, if you already know the structure of the file and where you want to start in it, you can do that. For example, if you know that each line of the file is exactly 100 bytes and you want to read from the start of the fifth line, `numOffset` should be 400.

**Note:** The number of bytes per character in a file depends on the text encoding (i.e., ANSI, UTF-8, UTF-16, or other), the byte order, and the language or character set.

**strStartTagName**
The name of the first project tag or array element in the series that will receive the values read from the CSV file. Project tags should be sequentially numbered with a numerical suffix, and the series is determined by incrementing that suffix (e.g., `MyTag1`, `MyTag2`, `MyTag3`, and so on). Array elements are handled similarly: the series is determined by incrementing the array index (e.g., `MyArray[1]`, `MyArray[2]`, `MyArray[3]`, and so on). You do not need to begin the numbering with 1.

**Note:** If the tag name is not enclosed in quotes, the function will try to use the value of specified tag.

**numNumberOfTags**
The number of project tags or array elements in the series that will receive values read from the CSV file. For example, if `strStartTagName` is `MyTag4` and `numNumberOfTags` is 5, five values will be read from the file and then written to the tags `MyTag4`, `MyTag5`, `MyTag6`, `MyTag7`, and `MyTag8`.

**Returned value**
If this function is successfully executed, it returns the position of the last byte read from the CSV file (including 0 if no bytes were read), which can be used in turn as the offset for the next batch of fields to be read.

If this function fails, it returns a negative value.
Notes

"CSV" is an abbreviation of "comma-separated values", and in most cases, a CSV file is simply a plain text file that uses commas (,) to delimit its data fields. Only comma delimiters — as opposed to tabs, spaces, pipes, or other characters — are supported in CSV files. Each field in the file contains exactly one value, even if the value is empty, and each project tag or array element will receive exactly one value read from the file.

When this function is executed, it will read at most one line of data. If a line feed is found before reading the specified number of values, execution of the function will be aborted without reading any values.

Examples

FileReadFields("C:\FieldFiles\FieldFile01.csv",400,"IntValueTag003",5)

FileReadFields("FieldFile02.csv",0,"IntValueTag[0]",10)

FileReadFields("FieldFile03.csv",0,"IntValueTag[IndexTag]",7)

FileReadMessage

The function FileReadMessage reads a message (i.e., a string of characters) from a text file, and then it writes that message to a project tag.

Syntax

FileReadMessage(strFilename, numOffset, strMessageTag, numCharsToRead)

FileReadMessage(strFilename, numOffset, strMessageTag, { numCharsToRead | 0 })

strFilename

The file path and/or name of the text file. If the file is located inside your project folder, you can specify either just the file name or the file path relative to that folder. If the file is located outside your project folder, you must specify the absolute file path.

numOffset

The number of bytes to skip in the text file before reading characters. To read from the start of the file, numOffset should be 0.

You can use this parameter to start reading from any position in the file, as long as you know how many bytes to skip. In most cases, you will simply take the value returned by the previous execution of this function (see "Returned value" below) and use it to resume reading where you previously stopped.

However, if you already know the structure of the file and where you want to start in it, you can do that. For example, if you know that each line of the file is exactly 100 bytes and you want to read from the start of the fifth line, numOffset should be 400.

Note: The number of bytes per character in a file depends on the text encoding (i.e., ANSI, UTF-8, UTF-16, or other), the byte order, and the language or character set.

strMessageTag

The name of the project tag (String type) that will receive the message read from the text file.
numCharsToRead
The number of characters to read from the text file, starting from the position specified by numOffset. If this value is 0, all characters up to the next line feed (LF) will be read.

Returned value
If this function is successfully executed, it returns the position of the last byte read from the text file (including 0 if no bytes were read), which be used in turn as the offset for the next message to be read. If this function fails, it returns a negative value.

Notes
This function can be used to read from any plain text file, as long as the correct file extension is specified in strFilename.

Examples
FileReadMessage("C:\Data\Messages01.txt",0,"MsgTag",0)
FileReadMessage("Messages02.txt",0,"MsgTag",140)

FileRename
FileRename is a built-in scripting function that renames a specified file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileRename</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax
FileRename(strOldName, strNewName)
strOldName
The path and old name of the file.
strNewName
The path and new name of the file.

Returned value
This function does not return any value.

Examples
FileRename( "C:\readme.txt", "C:\readthis.txt" )

FileWrite
FileWrite is a built-in function that writes a string to a specified text file. If the file does not exist, it will be created when the function is executed.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileWrite</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

\[ \text{FileWrite}(\text{strFileName}, \text{strWriteText}, \text{optNumAppend}) \]

\[ \text{FileWrite}(\text{strFileName, strWriteText} | , \text{optNumAppend}) \]

**strFileName**

The name of the text file.

By default, the file must be located in your project folder (i.e., the folder that contains the file `projectname.app`), on the computer that host the project runtime server. If the file is or should be located in another folder, specify the complete file path.

**strWriteText**

The text to be written to the file.

**optNumAppend**

A numerical flag indicating how the text should be written to the file:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Create a new ASCII file with the specified file name. If the file already exists, overwrite it.</td>
</tr>
<tr>
<td>1</td>
<td>Append to an existing ASCII file with the specified file name. If the file does not exist, create it.</td>
</tr>
<tr>
<td>2</td>
<td>Create a new Unicode (UTF-16LE) file with the specified file name. If the file already exists, overwrite it.</td>
</tr>
<tr>
<td>3</td>
<td>Append to an existing Unicode (UTF-16LE) file with the specified file name. If the file does not exist, create it.</td>
</tr>
</tbody>
</table>

This parameter is optional; if no value is specified, or if the specified value is not one of the accepted values described in the table above, the default value is 0.

**Returned value**

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Function successfully executed.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid parameter(s).</td>
</tr>
<tr>
<td>-2</td>
<td>Failed to open file. Either the drive is write-protected or the file name is invalid.</td>
</tr>
</tbody>
</table>

**Notes**

If this function is called in a project screen on a thin client, it will still write to the specified file in the project folder on the computer that hosts the project runtime server. You cannot use this function to write to a file on the thin client.

**Examples**

Create a new ASCII file in the project folder:

```
FileWrite("est.txt","This is a test.")
```

Append to an existing ASCII file in the project folder:

```
FileWrite("est.txt","This is a test.",1)
```
Append to an existing Unicode file in the Documents folder:

```plaintext
FileWrite("C:\Users\MyUser\Documents\est.txt","This is a test.",3)
```

**FileWriteFields**

The function **FileWriteFields** reads values contained in a series of project tags or array elements, and then it writes those values to the fields of a CSV file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileWriteFields</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
FileWriteFields(strFilename, numOffset, strStartTagName, numNumberOfTags)
```

**strFilename**

The file path and/or name of the CSV file to which the values will be written. If the file is located inside your project folder, you can specify either just the file name or the file path relative to that folder. If the file is located outside your project folder, you must specify the absolute file path.

**numOffset**

The number of bytes to skip in the CSV file before writing values. To write to the start of the file, `numOffset` should be 0.

You can use this parameter to start writing to any position in the file, as long as you know how many bytes to skip. In most cases, you will simply take the value returned by the previous execution of this function (see "Returned value" below) and use it to resume writing where you previously stopped.

However, if you already know the structure of the file and where you want to start in it, you can do that. For example, if you know that each line of the file is exactly 100 bytes and you want to write to the start of the fifth line, `numOffset` should be 400.

You can also specify a value of -1, which will automatically append the values to the end of the file.

**strStartTagName**

The name of the first project tag or array element in the series from which the function will read the values to be written to the CSV file. Project tags should be sequentially numbered with a numerical suffix, and the series is determined by incrementing that suffix (e.g., `MyTag1`, `MyTag2`, `MyTag3`, and so on). Array elements are handled similarly: the series is determined by incrementing the array index (e.g., `MyArray[1]`, `MyArray[2]`, `MyArray[3]`, and so on). You do not need to begin the numbering with 1.

**numNumberOfTags**

**Note:** The number of bytes per character in a file depends on the text encoding (i.e., ANSI, UTF-8, UTF-16, or other), the byte order, and the language or character set.

**Note:** If the tag name is not enclosed in quotes, the function will try to use the value of specified tag.
The number of project tags or array elements in the series to read. For example, if 
strStartTagName is MyTag4 and numNumberOfTags is 5, values will be read from the tags
MyTag4, MyTag5, MyTag6, MyTag7, and MyTag8 and then written to the CSV file.

Returned value

If this function is successfully executed, it returns the position of the last byte written to the CSV file
(including 0 if no bytes were written), which can be used in turn as the offset for the next batch of values to be
written.

If this function fails, it returns a negative value.

Notes

"CSV" is an abbreviation of "comma-separated values", and in most cases, a CSV file is simply a plain text file
that uses commas (,) to delimit its data fields. Only comma delimiters — as opposed to tabs, spaces, pipes, or
other characters — are supported in CSV files. Each field in the file can contain exactly one value, even if the
value is empty.

When this function is executed, it will write at most one line of data. If a line feed is found before writing the
specified number of tags, execution of the function will be aborted without writing any values.

Examples

FileWriteFields("C:\FieldFiles\FieldFile01.csv",400,"IntValueTag003",5)

FileWriteFields("FieldFile02.csv",-1,"IntValueTag[0]",10)

FileWriteFields("FieldFile03.csv",0,"IntValueTag[IndexTag]",7)

FileWriteMessage

The function FileWriteMessage reads a message (i.e., a string) from a project tag, and then it writes that
message to a text file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileWriteMessage</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

FileWriteMessage(strFilename,numOffset,strMessage,numAddLineFeed)

FileWriteMessage({strFilename, | numOffset | -1},strMessage, | numAddLineFeed | 0 | 1})

strFilename

The file path and/or name of the text file. If the file is located inside your project folder, you can
specify either just the file name or the file path relative to that folder. If the file is located outside
your project folder, you must specify the absolute file path.

numOffset

The number of bytes to skip in the text file before writing the message. To write to the start of
the file, numOffset should be 0.

You can use this parameter to start reading from any position in the file, as long as you know
how many bytes to skip. In most cases, you will simply take the value returned by the previous
execution of this function (see "Returned value" below) and use it to resume reading where you
previously stopped.
However, if you already know the structure of the file and where you want to start in it, you can do that. For example, if you know that each line of the file is exactly 100 bytes and you want to read from the start of the fifth line, numOffset should be 400.

You can also specify a value of -1, which will automatically append the message to the end of the file.

Note: The number of bytes per character in a file depends on the text encoding (i.e., ANSI, UTF-8, UTF-16, or other), the byte order, and the language or character set.

strMessage

The message to be written to the text file.

numAddLineFeed

A boolean value specifying whether to add a line feed (LF) to the end of the message. If this value is 1, a line feed is added. If this value is 0, a line feed is not added.

Returned value

If this function is successfully executed, it returns the position of the last byte written to the text file (including 0 if no bytes were written), which can be used in turn as the offset for the next message to be written.

If this function fails, it returns a negative value.

Notes

This function can be used to write to any plain text file, as long as the correct file extension is specified in strFilename.

Examples

FileWriteMessage("C:\Data\Messages01.txt",0,MsgTag,1)

FileWriteMessage("Messages02.txt",-1,"Append this text.",0)

FindFile

FindFile is a built-in scripting function that searches for all files that match a given search string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindFile</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported (see &quot;Notes&quot;)</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

FindFile(strFile{ | ,"optTagFilesFound"{ | ,optNumTimeout }})

FindFile( strFile, optTagFilesFound, optNumTimeout )

strFile

The name of the file(s) to search for.

You may use wildcards (*) to find multiple files. For example, *.gif to find all GIF files or log*.txt to find all log files in a sequence (e.g., log001.txt, log002.txt, log003.txt).
By default, the function only searches the project folder, but you may specify a file path (either relative or absolute) to search elsewhere. For example, if strFile is defined as...

```
\\volume name or IP address\Logs\log*.txt
```

...then the function will search the Logs directory on the specified network volume.

**optTagFilesFound**

An array (of String type) that will receive the names of the matching files. The array name must be enclosed in quotes; if it is not, then the function will try to get the contents of the array.

This is an optional parameter. If no value is specified, then the file names will not be saved and the function will only return the number of files found. For more information, see "Returned value" below.

The specified array will receive only the file names and not their paths, even if you define strFile to search outside the project folder. Also, the file names will be saved starting at array position 1 (e.g., MyArray[1]).

**optNumTimeout**

The timeout period (in milliseconds) for the function to be successfully executed.

This is an optional parameter. If no value is specified, then the project will continue searching until it has completely searched the specified directory.

**Note:** optNumTimeout is not supported if the target system / project server is a Windows Embedded device.

**Returned value**

This function will return one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Function timed out.</td>
</tr>
<tr>
<td>0</td>
<td>No matching files found.</td>
</tr>
<tr>
<td>n</td>
<td>Number of matching files found.</td>
</tr>
</tbody>
</table>

**Notes**

This function can be called from scripts executed on project clients (e.g., Graphics Script, Screen Scripts, Command animations), but the function itself is always executed on the project server. By default, it only searches the project folder on the server, and if strFile includes a relative file path, then it is relative to that folder.

Furthermore, because the function is executed synchronously on the project server, if strFile is poorly defined and/or optNumTimeout is not used, then the entire project — both the server and the clients — may hang while it searches for the files.

Finally, optNumTimeout is not supported if the target system / project server is a Windows Embedded device.

**Examples**

Find all text files in the Server's project folder:

```
FindFile( "*.txt" )
```

Find all Microsoft Word files in the project folder and then send the names of the matching files to StringArray, within a timeout period of 1000 milliseconds:

```
FindFile( "*.doc", "StringArray", 1000 )
```
**FindPath**

Verifies whether a directory exists.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindPath</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`FindPath(strPathName)`

*strPathName*

The file path for which to search.

**Returned value**

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Path not found</td>
</tr>
<tr>
<td>1</td>
<td>Path found</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>FindPath(&quot;C:\Windows&quot;)</td>
</tr>
</tbody>
</table>

**GetFileAttributes**

Reads the attributes of a specified file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFileAttributes</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`GetFileAttributes(strFile)`

*strFile*

The file path and name of the file from which to read the attributes.

**Returned value**

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>Read only</td>
</tr>
<tr>
<td>2</td>
<td>Hidden</td>
</tr>
<tr>
<td>4</td>
<td>System</td>
</tr>
<tr>
<td>16</td>
<td>Directory</td>
</tr>
<tr>
<td>32</td>
<td>Archive</td>
</tr>
<tr>
<td>128</td>
<td>Normal</td>
</tr>
<tr>
<td>256</td>
<td>Temporary</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetFileAttributes(&quot;C:\Readme.txt&quot;)</td>
</tr>
</tbody>
</table>
**GetFileTime**

Reads the time and date the file was last modified.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFileTime</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetFileTime( *strFileName*, *numFormat* )

strFileName

The file path and name of the file to be read.

numFormat

A numeric flag specifying the format of the returned data:

- 0: Returns the date and time from the file.
- 1: Returns only the file date.
- 2: Returns only the file time.

**Returned value**

Returns the date and or time the file was last modified.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetFileTime( &quot;C:\Readme.txt&quot; )</td>
</tr>
</tbody>
</table>

**GetHstInfo**

Returns the Start Time, End Time, and Duration of the specified history (*.HST) file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHstInfo</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetHstInfo( *strFileName*, *numInfoType* )

strFileName

The file path and name of the history file to be read.

numFormat

A numeric flag specifying the type of information to be returned:

- 0: Returns the Start Time of the file.
- 1: Returns the End Time of the file.
- 2: Returns the Duration (in hours) of the file.

**Returned value**

If the file cannot be read or the specified information cannot be returned, then an error is generated:

- -1 Failed to retrieve the Start Time; verify the history file exists and is valid.
- -2 Failed to retrieve the End Time; verify the history file exists and is valid.
- -3 Internal program error; please contact Technical Support.
The Studio TCP/IP server returned a Time that is incompatible with the format specified in the project screen or Web page. Please use the Verify Project tool to update the project and try again.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetHstInfo( &quot;batch&quot;, 0 )</td>
</tr>
<tr>
<td>Tag</td>
<td>GetHstInfo( &quot;hst/02060801.hst&quot;, 1 )</td>
</tr>
<tr>
<td>Tag</td>
<td>GetHstInfo( &quot;C:\batch.bat&quot;, 2 )</td>
</tr>
</tbody>
</table>

GetLine

Gets a specified line or search string from a text file and then stores the line in a String tag.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLine</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

GetLine( strFileName, Search, "tagStore", optNumCase, "optOverflowTag", optRunFromServer )

strFileName

A string value enclosed in quotes, or the name of a String tag that contains the value, specifying the name of the file to be searched. The name can be a fully qualified file path (e.g., C:\File.txt) or a simple file name (e.g., File.txt). In the latter case, the project will search for the file in the following paths:

• Local Station: The project will search for the file in the project folder and Web sub-folder.
• Thin Client: If the parameter optRunFromServer is set to 0, the path where the file will be searched is undetermined. If is set to 1, it will search for the file in the URL typed in the Browser, and if the file is not found, in the Backup URL.

Note: For Web-enabled projects, we recommend setting the optRunFromServer parameter to 1 and placing your files in the project’s Web sub-folder.

Search

There are two options for this parameter, based on the data type of the value you give it:

• If it is a string value or tag, then the function will search the text file for the first occurrence of the string and then copy the entire line that contains the occurrence to the tag specified by tagStore. Additional occurrences are counted (see Returned Values below) but not copied.
• If it is a numeric value or tag, then the function will go to that line number in the text file and then copy the line to the tag specified by tagStore. The first line of the file is line 0.

tagStore

Name of the String tag receiving the contents of the line pointed to by the function. This name must be enclosed in quotes; if it isn’t, then the function will use the contents of the tag rather than its name.

optNumCase

Optional numeric tag specifying whether the search is case-sensitive.

• 0: Not case-sensitive
• 1: Case-sensitive
**optOverflowTag**

*Optional* numeric tag receiving the result of overflow verification.

- 0: OK
- 1: Overflow

**optRunFromServer**

*Optional* numeric tag ignored when the function is called on local stations. On Thin Clients, this parameter indicates the following:

- 0: Retrieves the file from the Thin Client machine (do not use this value with non-fully qualified names)
- 1: Retrieves the file from the Web Server. If the file name is not a URL, then the function will ignore the project path and search for the file in the URL where the screen files are located.

**Returned value**

If the function is successfully executed, then it returns the total number lines in which the search string was found. Otherwise, the function returns one of the following errors:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>String was not found in the target file</td>
</tr>
<tr>
<td>-1</td>
<td>File not found</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid <code>strFileName</code> parameter</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid <code>strSeqChar</code> parameter</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid <code>strStoreTag</code> parameter</td>
</tr>
<tr>
<td>-5</td>
<td>Invalid <code>optNumCase</code> parameter</td>
</tr>
<tr>
<td>-6</td>
<td>Invalid <code>optNumOverflowTag</code> parameter</td>
</tr>
<tr>
<td>-7</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>-8</td>
<td>Invalid line number</td>
</tr>
</tbody>
</table>

**Notes**

**Note:** This function only supports ASCII and UTF-16LE text encoding. (UTF-16LE is the Unicode implementation that is natively supported by Windows.) If you use this function to get text from a UTF-8 or UTF-16BE encoded file, then you may see some invalid characters.

**Important:** This function can only read up to 509 characters in a single function call. If a line has more than 509 characters (i.e., 507 alphanumeric + CR + LF), then the function will read it as two or more lines. This will also increase the effective line count for the purposes of the Search parameter. So, for line 100 that has 1024 characters (i.e., 1022 alphanumeric + CR + LF), the function must be called three times:

```plaintext
GetLine( "C:\FileName.txt", 100, "strTagStore[1]" )  //Reads the first 509 characters
GetLine( "C:\FileName.txt", 101, "strTagStore[2]" )  //Reads the second 509 characters
GetLine( "C:\FileName.txt", 102, "strTagStore[3]" )  //Reads the last 6 characters
```

After this, line 101 of the source file is actually counted by the function as line 103. Therefore, to avoid unnecessarily complicated line counting, you should make sure the source file is limited to 509 characters per line.
Examples

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
<th>// Comment</th>
</tr>
</thead>
</table>
| Tag | GetLine( "C:\TechRef v61.doc", "Studio version 6.1", "ReturnedLine" )  
// Gets the first occurrence of "Studio version 6.1". | |
| Tag | GetLine( "C:\Readme.txt", 1, "ReturnedLine", 0, "Overflow" )  
// Gets the second line of the file. | |

**Hst2Txt**

Hst2Txt is a built-in function that exports Trend historical data from InduSoft Web Studio's proprietary binary format to a text file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hst2Txt</td>
<td>File</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

Hst2Txt(\texttt{strStartDate, strStartTime, numDuration, numGroup, strTargetFile, optStrSeparator, optNumMilliseconds})

\begin{align*}
\text{Hst2Txt(} & \texttt{strStartDate, strStartTime, numDuration, numGroup, strTargetFile, } \\
& \texttt{optStrSeparator, optNumMilliseconds}) \\
\end{align*}

- **strStartDate**
  
  The start date (e.g., 04/14/2002) of the data to be exported.

- **strStartTime**
  
  The start time (e.g., 06:30:00) of the data to be exported.

- **numDuration**
  
  The duration (in hours) of the data to be exported, starting from the specified start date and time.

- **numGroup**
  
  The trend group or worksheet number. For more information, see Trend worksheet on page 305.

- **strTargetFile**
  
  The name of the text file to which the data will be exported.

  By default, the file must be located in your project folder (i.e., the folder that contains the file \texttt{projectname.APP}), on the computer that host the project runtime server. If the file is or should be located in another folder, specify the complete file path.

- **optStrSeparator**
  
  The character that will be used to separate the values, a.k.a. the delimiter.

  This parameter is optional; if no value is specified, a TAB character (\texttt{\textbackslash t}) will be used, making the file a tab-delimited text file.

- **optNumMilliseconds**
  
  A numerical flag indicating whether to show millisecond-precision in the timestamp on each entry:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not show milliseconds.</td>
</tr>
<tr>
<td>1</td>
<td>Show milliseconds.</td>
</tr>
</tbody>
</table>

---

Page 782
This parameter is optional; if no value is specified, the default value is 0.

**optStrFormat**

The date format that will be used in the timestamp on each entry:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMY</td>
<td>Day, Month, Year</td>
</tr>
<tr>
<td>MDY</td>
<td>Month, Day, Year</td>
</tr>
<tr>
<td>YMD</td>
<td>Year, Month, Day</td>
</tr>
</tbody>
</table>

This parameter is optional; if no value is specified, the default value is DMY.

**optNumInterval**

The interval between entries to be exported. Only entries at this interval are exported to the specified file. For example, if optNumInterval is 10, only every tenth entry is exported.

This parameter is optional; if no value is specified, every entry is exported.

**Returned value**

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Error. Previous execution of the HST2TXT has not yet been completed.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
<tr>
<td>-1</td>
<td>InStudiot.dll not found in the program folder.</td>
</tr>
<tr>
<td>-2</td>
<td>DLL functions not found.</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

**Notes**

If this function is called in a project screen on a thin client, it will still write to the specified file in the project folder on the computer that hosts the project runtime server. You cannot use this function to write to a file on the thin client.

To export to a comma-separated values (CSV) file that can be easily imported into Microsoft Excel and other spreadsheet applications, specify a file name with the CSV suffix (e.g., data.csv) and a comma separator.

**Examples**

Export 0.1 hour (i.e., six minutes) of historical data from Trend group/worksheet 3 to a slash-delimited text file:

```
Hst2Txt("04/14/2002","06:30:00",0.1,3,"data.txt","\")
```

Export 0.1 hour (i.e., six minutes) of historical data from Trend group/worksheet 3 to a CSV file using the MDY date format:

```
Hst2Txt("04/14/2002","06:30:00",0.1,3,"data.csv",","",0,"MDY")
```

Export every tenth entry in 0.1 hour (i.e., six minutes) of historical data from Trend group/worksheet 3 to a CSV file using the MDY date format:

```
Hst2Txt("04/14/2002","06:30:00",0.1,3,"data.csv",","",0,"MDY",10)
```
**HST2TXTIsRunning**

Returns the status of the HST2TXT function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>HST2TXTIsRunning</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`HST2TXTIsRunning()`

This function takes no parameters.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HST2TXT is still running.</td>
</tr>
<tr>
<td>-1</td>
<td>Last conversion process was executed properly.</td>
</tr>
<tr>
<td>-2</td>
<td>Reserved.</td>
</tr>
<tr>
<td>-3</td>
<td>File not found. There are no history files in the configured time interval for the group specified.</td>
</tr>
<tr>
<td>-4</td>
<td>Cannot open history file.</td>
</tr>
<tr>
<td>-5</td>
<td>Cannot create/open ASCII file.</td>
</tr>
<tr>
<td>-6</td>
<td>Cannot read file information from history file.</td>
</tr>
<tr>
<td>-7</td>
<td>Invalid file type.</td>
</tr>
<tr>
<td>-8</td>
<td>Cannot read header information from history file.</td>
</tr>
<tr>
<td>-9</td>
<td>Invalid number of tag in the header information (0 &gt; nTags &gt; 250).</td>
</tr>
<tr>
<td>-10</td>
<td>Cannot create Header file (.hdr).</td>
</tr>
<tr>
<td>-20</td>
<td>InStudiot.dll was not found.</td>
</tr>
<tr>
<td>-30</td>
<td>Cannot access DLL function.</td>
</tr>
</tbody>
</table>

**ImportXML**

The function ImportXML is used to import Studio XML files into your project during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImportXML</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported (see “Notes” below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ImportXML(strXMLFile, optStrDestFile, optNumFileType, optNumReplaceDuplicate)`

**strXMLFile**

The file path and name of the XML file to be imported. If you specify only a file name with no path, the runtime will automatically look in the Screen and Web sub-folders of your project folder.

**optStrDestFile**
The new name of the project file, if any.
This parameter is optional; if you do not specify a tag or value, then the default value is "screen".

**optNumFileType**
A numeric flag indicating the type of Studio XML file to be imported. At this time, only Studio XML Screen files are supported.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Studio XML Screen file</td>
</tr>
</tbody>
</table>

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

**optNumReplaceDuplicate**
A numeric flag indicating whether the imported XML file should replace an existing project file of the same name. For example, Objects.xml and Objects.scr would be duplicates.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not replace duplicate project file.</td>
</tr>
<tr>
<td>1</td>
<td>Replace duplicate project file.</td>
</tr>
</tbody>
</table>

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

**Returned value**
This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Invalid license; this function is not available in Demo Mode.</td>
</tr>
<tr>
<td>-1</td>
<td>Cannot execute this function on the project client (i.e., Viewer). Execute the function on the project server.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>2</td>
<td>Invalid parameter data type.</td>
</tr>
<tr>
<td>3</td>
<td>Cannot replace duplicate file.</td>
</tr>
<tr>
<td>4</td>
<td>Failed to load import module. ImportXML does not work when your project is running as a Windows service. For more information, see Run a project as a Windows service on page 114.</td>
</tr>
<tr>
<td>5</td>
<td>File type (optNumFileType) not supported.</td>
</tr>
<tr>
<td>6</td>
<td>Failed to create destination file.</td>
</tr>
<tr>
<td>7</td>
<td>Internal error.</td>
</tr>
<tr>
<td>8</td>
<td>Failed to save imported screen.</td>
</tr>
<tr>
<td>9</td>
<td>Failed to retrieve screen file.</td>
</tr>
<tr>
<td>10</td>
<td>Internal XML file error.</td>
</tr>
</tbody>
</table>

**Notes**
This function can be executed only on the project server, where the InduSoft Web Studio development environment must be installed and running with at least a Runtime license. You can use the functions `StartTask`, `EndTask`, and `IsTaskRunning` with the parameter "Studio" to programmatically control the development environment.
For more information about Studio XML Screen files and how they are created, see Import a Studio XML Screen on page 167.

Examples

Look for `Screen1.xml` in the Screen and Web sub-folders, and then import it to create `Screen1.scr` if it does not already exist:

```plaintext
ImportXML("Screen1.xml")
```

Look for `Screen1.xml` at the specified file path, and then import it to create `Screen1.scr` if it does not already exist:

```plaintext
ImportXML("C:\Users\username\Documents\InduSoft Web Studio v7.1 Projects\SMA_Project\Screen\Screen1.xml")
```

Look for `Screen1.xml` at the specified file path, and then import it to create `Screen9.scr` if it does not already exist:

```plaintext
ImportXML("C:\Users\username\Documents\InduSoft Web Studio v7.1 Projects\SMA_Project\Screen\Screen1.xml","Screen9.scr")
```

Look for `Screen1.xml` at the specified file path, and then import it to replace `Screen9.scr`:

```plaintext
ImportXML("C:\Users\username\Documents\InduSoft Web Studio v7.1 Projects\SMA_Project\Screen\Screen1.xml","Screen9.scr",0,1)
```

LookupContains

This function verifies that an external file contains the specified keyword in its key column.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>LookupContains</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

LookupContains( `strKey` )

`strKey` - The keyword to look for in the file's keywords column.

Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Specified keyword not found.</td>
</tr>
<tr>
<td>1</td>
<td>Specified keyword found.</td>
</tr>
</tbody>
</table>

Notes

The external file must already be loaded by calling the function LookupLoad.

Examples

```plaintext
LookupContains( "customer167" )
```
**LookupGet**

This function gets a value from an external file by cross-referencing from a specified keyword.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>LookupGet</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`LookupGet(strKey)`

*strKey*

The keyword to look for in the file's keywords column.

**Returned value**

This function returns (as a string) the cross-referenced value from the file's specified values column.

If no value is found, then this function returns strKey.

**Notes**

The external file must already be loaded by calling the function `LookupLoad`.

**Examples**

`LookupGet("customer167")`

---

**LookupLoad**

This function loads an external file — typically, a delimited text file — that can be used to look up table values. One column of the file is designated as the keywords column, and another column is designated as the values column.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>LookupLoad</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`LookupLoad(strFileName, numColKey, numColValue, strDelimiters)`

*strFileName*

The file path and name of the external file.

*numColKey*

The number of the column/field that contains the keywords.

*numColValue*

The number of the column/field that contains the desired values.

*strDelimiters*

The delimiter that separates the columns/fields.

**Returned value**

This function returns the number of rows/lines in the specified file.

If the specified file cannot be found, then this function returns a negative number as an error code.

**Notes**

This function only loads the specified file; it doesn’t do anything with the file. To use the file, call the `LookupContains` and `LookupGet` functions.
Also, to load another file, simply call this function again. Only one file can be loaded at a time, however; the new file replaces the old in the project’s memory.

Examples

\texttt{LookupLoad( "C:\Temp\customerlist.csv", 1, 4, ",", )}

\textbf{PDFCreate}

Creates a PDF file from the specified source file.

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Function & Group & Execution & Windows & Embedded & Thin Client & Mobile Access \\
\hline
PDFCreate & File & Synchronous & Supported & Not supported & Supported & Not supported \\
\hline
\end{tabular}
\end{center}

Syntax

\texttt{PDFCreate\( (strSourceFile\{ | , optStrPdfFile \})\)}

\texttt{strSourceFile}

String specifying the file path and name of the desired source file (\texttt{*.doc}, \texttt{*.txt}, or \texttt{*.rtf}). If a complete path is not specified, then the function will look for the source file in the project folder.

\texttt{optStrPdfFile}

\textit{Optional} string specifying the file path and name of the created PDF file. If a file path is not specified, then the PDF file will be saved in the same location as the source file. If this parameter is omitted — that is, if no file path or name is specified at all — then the PDF file will be saved in the same location and with the same name as the source file. Only a new extension is added. For example, \texttt{\textbackslash path\MyDocument.rtf} becomes \texttt{\textbackslash path\MyDocument.pdf}.

\textbf{Note:} When entering the file name without a path, a leading backlash (\textbackslash) is optional.

Returned value

\begin{center}
\begin{tabular}{|c|c|}
\hline
Value & Description \\
\hline
0 & Success \\
1 & Error in PDF profile information \\
3 & Error saving PDF file \\
4 & Job canceled \\
101 & Error initializing PDF resource \\
102 & Specified source file not found \\
103 & Error generating PDF file \\
104 & Wrong number of parameters \\
105 & Wrong parameter type \\
\hline
\end{tabular}
\end{center}

\textbf{Note:} This function only supports the execution of one job at a time. If more than one user or command attempts to call the function at the same time, then the function will fail and return a value of \texttt{101}. 

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFCreate( &quot;C:\Report1.rtf&quot; )</td>
<td></td>
</tr>
<tr>
<td>PDFCreate( &quot;C:\Report2.doc&quot;, &quot;C:\Converted1.pdf&quot; )</td>
<td></td>
</tr>
<tr>
<td>PDFCreate( &quot;C:\Report3.txt&quot;, &quot;C:\Data\Converted1.pdf&quot; )</td>
<td></td>
</tr>
</tbody>
</table>

Print
Prints a text file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print</td>
<td>File</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

Print(strFilePath| , optNumOrientation )

strFilePath
Path and name of the text file that will be printed.

optNumOrientation
Set the paper orientation as follows:
- 0 (default) = Portrait
- 1 = Landscape

Returned value
No returned values.

Note: The optNumOrientation parameter is not supported when running the project on a Windows Embedded target system.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print(&quot;C:\ReadMe.txt&quot;)</td>
<td></td>
</tr>
<tr>
<td>Print(&quot;C:\ReadMe.txt&quot;, 1)</td>
<td></td>
</tr>
<tr>
<td>Print(TagFileName, 0)</td>
<td></td>
</tr>
</tbody>
</table>

Note: This function can be used to print the contents of text files only. Information in any other format (e.g., pictures, binary files, etc.) cannot be printed with this function.

RDFileN
Launches a File Browser window allowing you to select a file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDFileN</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

Syntax

\[ \text{RDFileN( "tagSelectedFile", strSearchPath, strMask, optNumChangeDir )} \]

**tagSelectedFile**
Name of the string tag receiving the name and path of a selected file. The tag name must be enclosed in quotes, or the project will try to get the contents of the tag. Moreover, it must be a valid tag name — it cannot be a VBScript variable name, for example.

**strSearchPath**
The file path (directory) to search.

**strMask**
The mask used to filter the files.

**optNumChangeDir**
Optional numeric tag that indicates whether the operator will be able to change the browsing directory. If this parameter is omitted or set TRUE (1), then the window opened by this function will allow the operator to navigate to different directories. If it is set FALSE (0), then the window will be restricted to the directory specified by strSearchPath.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>One of the parameters is not a string</td>
</tr>
<tr>
<td>2</td>
<td>Parameter 1 contains an invalid tag name</td>
</tr>
<tr>
<td>3</td>
<td>The user canceled the operation</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>RDFileN( &quot;FileName&quot;, &quot;C:\Studio&quot;, &quot;*.doc&quot;, 1 )</td>
</tr>
</tbody>
</table>

### WebGetFile

Downloads a file from a specified address and then saves it locally.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebGetFile</td>
<td>File</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

WebGetFile\( (strURL, strLocalPath) \)

**strURL**
The URL (i.e., the Web address) of the file you want to download.

**strLocalPath**
The complete local file path where you want to save the file.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Timeout</td>
</tr>
<tr>
<td>0</td>
<td>File not found</td>
</tr>
<tr>
<td>1</td>
<td>File saved successfully</td>
</tr>
</tbody>
</table>
## Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>WebGetFile( myURL, myFilePath )</code></td>
</tr>
</tbody>
</table>
Graphic functions

These functions are used to manipulate and print project screens.

AutoFormat

Automatically formats a real number to a preset number of decimal places, according to the virtual table of settings created by the SetDecimalPoints function. (This is similar to the Format function, except that you do not need to specify the number of decimal places.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoFormat</td>
<td>Graphic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

AutoFormat(numValue)

numValue

The real number to be formatted.

Returned value

This function returns a formatted string.

Examples

In the following examples, the SetDecimalPoints function has already been used to set 3 decimal places for values greater than equal to 1.5 and 1 decimal place for values less than or equal to −3.

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>AutoFormat( 1.543210 ) // Returned value = &quot;1.543&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>AutoFormat( −3.123456 ) // Returned value = &quot;−3.1&quot;</td>
</tr>
</tbody>
</table>

ExecuteCustomCall

The function ExecuteCustomCall invokes custom code in the ActiveX container that contains the ISSymbol control. For example, you can use this function to invoke custom JavaScript on web thin clients.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecuteCustomCall</td>
<td>Graphic</td>
<td>Asynchronous</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

ExecuteCustomCall(optParam1, optParam2, ..., optParamN)

ExecuteCustomCall({ | optParam1{ | , optParam2{ | ,optParamN}}})

optParam1

optParam2

optParamN

Parameters to be passed through the ISSymbol control to the ActiveX container. See "Notes" below.

Returned value

This function returns one of the following possible values:
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Function is not supported. This error will be returned if the function was called by any of the Background Task worksheets (e.g., Math, Script, Scheduler) on the project server. The function only works on web thin clients, and it only affects the web thin client on which it was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

### Notes

This function can be used to invoke custom code in any ActiveX container that can add custom events to a control and receive VBArray objects. The following example only shows how this is done when the ActiveX container is Internet Explorer and the ISSymbol control is being used to drive a web thin client.

When you save a project screen as HTML, the resulting webpage by default is simply a frame for the ISSymbol control. The control is embedded in the webpage as an object, similar to a Flash player or a Java applet, and then it is configured to access your IWS project over the network. For more information, see About web thin clients on page 533.

There is nothing, however, that prevents you from customizing the webpage, as long as the object configuration is maintained. And as part of customizing the webpage, you might insert the following piece of JavaScript code:

```html
<script language="JavaScript"> <!--
    ISSymbol1.attachEvent("OnCustomCall",
        function (parameters) {
            var params = (new VBArray(parameters)).toArray();
            ...
        });
//--></script>
```

This code uses the method `attachEvent` to add the event `OnCustomCall` to the ISSymbol control, and then it binds a function to that event. The first line of the function sets up an array that will receive the parameters passed to the function from `ExecuteCustomCall`. (The parameters are passed as a VBArray object, from your IWS project through the ISSymbol control to the JavaScript function, and the first line of the function converts that VBArray object to a standard JavaScript array.) The rest of the function can be whatever you want it to be.

Once this code is inserted in the webpage, calling `ExecuteCustomCall` in your IWS project will cause the ISSymbol control to fire the event and execute the function in the web browser.

#### Note

The method `attachEvent` is supported only in Internet Explorer 5 through Internet Explorer 10. It will be deprecated in Internet Explorer 11. Instead, you may use the DOM-standard methods `createEvent`, `initEvent`, and `addEventListener` to do the same thing. For more information, go to: msdn.microsoft.com/library/ms536343.aspx

If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

### GetScrInfo

The function `GetScrInfo` gets information about an open project screen.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetScrInfo</td>
<td>Graphic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

```
GetScrInfo(strScreenName,"tagResult",optNumResultType,optNumID)
```

```
GetScrInfo(strScreenName,"tagResult"{   |
| , optNumResultType | 0 | 1 | 2 | 3 | 4 } | , optNumID } )
```

**strScreenName**
The name of the screen about which you want to get information.

**tagResult**
The name of the project tag that will receive the information.

> **Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**optNumResultType**
A numeric flag specifying the type of information to be retrieved by the function:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Default value. Writes the TOP, LEFT, BOTTOM and RIGHT screen coordinates to each consecutive position of the array tag specified for tagResult.</td>
</tr>
<tr>
<td>1</td>
<td>Writes the TOP screen coordinate to the tag specified for tagResult.</td>
</tr>
<tr>
<td>2</td>
<td>Writes the LEFT screen coordinate to the tag specified for tagResult.</td>
</tr>
<tr>
<td>3</td>
<td>Writes the BOTTOM screen coordinate to the tag specified for tagResult.</td>
</tr>
<tr>
<td>4</td>
<td>Writes the RIGHT screen coordinate to the tag specified for tagResult.</td>
</tr>
</tbody>
</table>

This is an optional parameter; the default value is 0.

**optNumID**
The specific instance number of the screen. (The ID is assigned when the screen is opened with the function `Open`.) This is an optional parameter; the default ID is 0.

**Returned value**
This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Invalid tag specified for tagResult.</td>
</tr>
<tr>
<td>-3</td>
<td>optNumResultType is 0, but an array tag is not specified for tagResult.</td>
</tr>
<tr>
<td>-2</td>
<td>Memory allocation error.</td>
</tr>
<tr>
<td>-1</td>
<td>String not specified for strScreenName and/or tagResult.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

**Notes**
If the option **Enforce Web functionality equivalence** is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.
Examples
Retrieve the TOP, LEFT, BOTTOM and RIGHT coordinates of the "main" screen and then write them to the first four positions of the array tag TagXY:

```
GetScrInfo("main","TagXY[0]")
```

Retrieve the BOTTOM coordinate of the "main" screen and then write it to TagXY:

```
GetScrInfo("main","TagXY",3)
```

Retrieve the LEFT coordinate of the "main" screen with ID 10 and then write it to TagXY:

```
GetScrInfo("main","TagXY",2,10)
```

PrintSetup
The function PrintSetup displays a standard print setup dialog box on the client, in which the user can select and configure a printer for printing project screens.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrintSetup</td>
<td>Graphic</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax
PrintSetup()

This function takes no parameters.

Returned value
This function does not return any value.

Notes
If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

PrintWindow
The function PrintWindow prints a screenshot of a project screen. The screen does not need to be open and active; the function can print a screen running in the background or even closed screen file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrintWindow</td>
<td>Graphic</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax
PrintWindow(strScreenName, optNumOrientation, optNumID, optStrMnemonicList)

PrintWindow(strScreenName| [ [ optNumOrientation | 0 | 1]] | , optNumID| [ , optStrMnemonicList ] ] )}
**strScreenName**

The name of the screen to be printed. If this parameter is omitted, then the currently active screen will be printed. (This parameter must be omitted when executing the function on a Windows Embedded target system.)

**optNumOrientation**

A numeric flag specifying the print orientation:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Portrait</td>
</tr>
<tr>
<td>1</td>
<td>Landscape</td>
</tr>
</tbody>
</table>

This parameter is optional; if no value is specified, the default value is 0.

**Note:** This parameter is not supported in projects running on Windows Embedded target systems.

**optNumID**

The specific ID number of the screen. (This number is assigned when the screen is opened using the function `Open`.)

This parameter is optional; if no value is specified, the default value is 0.

**optStrMnemonicList**

A string that describes how the custom properties of any generic objects or linked symbols in the screen will be completed when the screen is printed. This string has the following syntax...

```
#Label:Value
```

...where `Label` is the name of the property and `Value` is the tag, expression or literal value that the property will receive. You can declare two or more mnemonics, as long as they are separated by spaces. See the Examples section below for an example.

**Note:** This parameter does not work for a screen that is already open, because if the screen has been opened, the custom properties have received their values.

**Returned value**

This function does not return any value.

**Notes**

If the option `Enforce Web functionality equivalence` is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

**Examples**

Print the currently active screen in portrait orientation:

```
PrintWindow()
```
Print the screen named "Main" in landscape orientation:

   PrintWindow("Main",1)

Print the screen specified by the tag MyScreenName:

   PrintWindow(MyScreenName)

Print the screen named "Main" with ID 10:

   PrintWindow("Main",1,10)

Print the screen named "Main", replacing the custom properties Mne1 and Mne2 with the values of Tag1 and Tag2, respectively:

   PrintWindow("Main",1,0,"#Mne1:Tag1 #Mne2:Tag2")

Tip: You can use this function to print graphical reports that include Alarm/Event Control and Trend Control objects.

ResetDecimalPointsTable

Resets the virtual table of settings created by the SetDecimalPoints function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResetDecimalPointsTable</td>
<td>Graphic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

ResetDecimalPointsTable()

This function takes no parameters.

Returned value

This function does not return any value.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ResetDecimalPointsTable() // Resets the virtual table of settings.</td>
</tr>
</tbody>
</table>

RGBColor

Returns the number of the color defined by the RGB (Red, Green, Blue) codes.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGBColor</td>
<td>Graphic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Syntax

RGBColor(numRed, numGreen, numBlue)

numRed
Red code from the RGB code.

numGreen
Green code from the RGB code.

numBlue
Blue code from the RGB code.

**Returned value**
This function returns the number of the color defined by the RGB (Red, Green, Blue) codes.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagColor</td>
<td><code>RGBColor(51,153,102)</code> // This function returns the value 13434828, which is the color code for Sea Green.</td>
</tr>
<tr>
<td>TagColor</td>
<td><code>RGBColor(TagRed,TagGreen,TagBlue)</code> // This function returns the color code of the RGB values set in the tags TagRed, TagGreen and TagBlue, respectively.</td>
</tr>
</tbody>
</table>

**Tip:** See the list of RGB Codes and Color values for the most used colors in the Color Interface section.

**RGBComponent**
RGBComponent is a built-in scripting function that gets the level of a color component (red, green, or blue) in a specified color.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGBComponent</td>
<td>Graphic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
RGBComponent(numColor, {numComponent | 0 | 1 | 2})
```

**numColor**
The decimal code for a 24-bit RGB color, which can be any integer value between 0 and 16777215. (This color model is also known as "Truecolor" or "millions of colors.")

**numComponent**
The color component for which you want to get the level: 0 is red, 1 is green, and 2 is blue.

**Returned value**
This function returns an integer value between 0 or 255, which represents the level of the color component in the specified color.

**Notes**
For a list of frequently used RGB color codes and their equivalent "plain English" names, see Color Interface.

**Examples**
Get the level of red in color code 13434828 (i.e., sea green):

```
RGBComponent( 13434828, 0 )
```
Get the level of the component specified by `TagComponent` in the color specified by `TagCode`:

```
RGBComponent( TagCode, TagComponent )
```

### SaveScreenShot

`SaveScreenShot` is a built-in function that takes a screen shot of a project screen and then saves it as an image file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaveScreenShot</td>
<td>Graphic</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

#### Syntax

```
SaveScreenShot( optStrScreenName, optStrOutputFile, optNumFormat )
```

```
SaveScreenShot({   | optStrScreenName|   |optStrOutputFile|   |, { optNumFormat | 0 | 1 | 2 | 3 | 4 | 5 } } {   | optStrOutputFile|   |, { optNumFormat | 0 | 1 | 2 | 3 | 4 | 5 } } )
```

#### optStrScreenName

The file path and name of a project screen file (*.scr). If no file path is specified, the file must be located in the Screen sub-folder of the project folder (e.g., `C:\Documents\InduSoft Web Studio v7.1 Projects\projectname\Screen\screenname.scr`).

For projects running on Windows (using InduSoft Web Studio) and Windows Embedded Standard (using EmbeddedView), the screen may be either open or closed. For projects running on Windows Embedded Compact (using CEView), the screen must be open.

This parameter is optional; if no value (or "") is specified, the currently open and active screen is used.

#### optStrOutputFile

The name of the output file. If no file path is specified, the file is saved in the Web sub-folder of the project folder (e.g., `C:\Documents\InduSoft Web Studio v7.1 Projects\projectname\Web\screenname.jpg`).

This parameter is optional; if no value is specified, either the value of `optStrScreenName` or simply `Screenshot.jpg` is used.

#### optNumFormat

The format of the image file:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BMP</td>
</tr>
<tr>
<td>1</td>
<td>JPG</td>
</tr>
<tr>
<td>2</td>
<td>PNG</td>
</tr>
<tr>
<td>3</td>
<td>GIF</td>
</tr>
<tr>
<td>4</td>
<td>TIFF</td>
</tr>
<tr>
<td>5</td>
<td>Auto</td>
</tr>
</tbody>
</table>

This parameter is optional; if no value is specified, the default is 1 (JPG).

#### Returned value

This function returns the following possible values:
### Value
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success.</td>
</tr>
<tr>
<td>-1</td>
<td>Wrong number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>Wrong parameter types.</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid directory.</td>
</tr>
<tr>
<td>-4</td>
<td>Second parameter cannot be empty.</td>
</tr>
<tr>
<td>-5</td>
<td>Wrong format / invalid option for third parameter.</td>
</tr>
<tr>
<td>-6</td>
<td>Failed to save file.</td>
</tr>
<tr>
<td>-7</td>
<td>Failed to create compatible bitmap.</td>
</tr>
<tr>
<td>-10000</td>
<td>Project is not running.</td>
</tr>
</tbody>
</table>

#### Notes
For this function to be executed properly, the project must be running and the Viewer task must be started. The function can be called by background tasks (i.e., Script and Math worksheets) or in the Database Spy, but it will fail if the Viewer task is not also started. This is because the Viewer task is needed to render the screen. For more information about starting run-time tasks, see Execution Tasks dialog on page 112.

#### Examples
Take a screen shot of the current screen, and then save it as ScreenShot.jpg:

```plaintext
SaveScreenShot()
```

Take a screen shot of main.scr, and then save it as main.jpg:

```plaintext
SaveScreenShot("main.scr")
```

Take a screen shot of the current screen, and then save it as a bitmap with the name of the currently logged user:

```plaintext
SaveScreenShot("", UserName, 0)
```

### SetDecimalPoints
Sets the number of decimal places to be displayed, for a specified range of real numbers. This setting will be used by all screen objects and animations that have the Auto Format option enabled, as well as by the AutoFormat function.

#### Syntax
```plaintext
SetDecimalPoints(numBaseValue, numDecimalPoints)
```

- **numBaseValue**
  The base value of the range of real numbers. For negative values, the range includes all real numbers less than or equal to that value. For positive values, the range includes all real numbers greater than or equal to that number. (You can set the other limit of the range by calling the function again with a new set of parameters.)

- **numDecimalPoints**
The number of decimal places to be displayed, for the range of real numbers specified by `numBaseValue`.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error</td>
</tr>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Notes**

If you call this function more than once with different parameters for each call, then you can build a virtual table of format settings. You can set a different number of decimal places for each range of real numbers, and all of the settings are saved for the duration of runtime or until you reset the table using the `ResetDecimalPointsTable` function.

Note: This formatting does not change the actual value of any tag or expression. It only changes how the value is displayed by on-screen objects.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>SetDecimalPoints( 1.5, 3 )</code> // Displays 3 decimal places for all real numbers greater than or equal to 1.5.</td>
</tr>
<tr>
<td></td>
<td><code>SetDecimalPoints( -3, 1 )</code> // Displays 1 decimal place for all real numbers less than or equal to -3.</td>
</tr>
</tbody>
</table>

**SetDisplayUnit**

Finds all tags and all Grid object and Trend Control object values that have a specific engineering unit (as stored in the `Unit` tag field), and then sets the `DisplayUnit`, `UnitDiv`, and `UnitAdd` fields on those tags.

**Syntax**

```
SetDisplayUnit(strUnitOrigin, strDisplayUnit, numDiv, numAdd)
```

*strUnitOrigin*

The engineering unit to be matched.

*strDisplayUnit*

The new value for the `DisplayUnit` tag field.

*numDiv*

The new value for the `UnitDiv` tag field.

*numAdd*

The new value for the `UnitAdd` tag field.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>-1</td>
<td>Wrong number of parameters</td>
</tr>
<tr>
<td>-2</td>
<td><code>strUnitOrigin</code> parameter is empty</td>
</tr>
<tr>
<td>-3</td>
<td><code>numDiv</code> parameter is invalid (equal to 0)</td>
</tr>
</tbody>
</table>
Notes
This function only affects how the tag values are displayed on screen; it does not change the actual tag values in any way.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SetDisplayUnit( &quot;C&quot;, &quot;F&quot;, 0.555556, 32 ) // For all tags and object values with a Unit of &quot;C&quot;, the DisplayUnit tag field is set to &quot;F&quot;, the UnitDiv tag field is set to 0.555556, and the UnitAdd tag field is set to 32.</td>
</tr>
</tbody>
</table>

SetTagDisplayUnit
Sets the DisplayUnit, UnitDiv, and UnitAdd properties on a specific tag.

Syntax
SetTagDisplayUnit(strTagName, strDisplayUnit, numDiv, numAdd)

strTagName
The name of the specific tag on which the DisplayUnit, UnitDiv and UnitAdd tag fields will be set.

**Note:** If this parameter is given a tag, then that tag should contain the name of the tag on which the tag fields will be set.

strDisplayUnit
The new value for the DisplayUnit tag field.

numDiv
The new value for the UnitDiv tag field.

numAdd
The new value for the UnitAdd tag field.

Returned value

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success.</td>
</tr>
<tr>
<td>-1</td>
<td>Wrong number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>Specified tag doesn't exist.</td>
</tr>
<tr>
<td>-3</td>
<td>numDiv parameter is invalid (equal to 0).</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SetTagDisplayUnit( &quot;TagTemp&quot;, &quot;F&quot;, 0.555556, 32 ) // For the tag &quot;TagTemp&quot;, the DisplayUnit tag field is set to &quot;F&quot;, the UnitDiv tag field is set to 0.555556, and the UnitAdd tag field is set to 32.</td>
</tr>
</tbody>
</table>
Translation functions

These functions are used to access the translation tool during runtime.

**Ext**

Translates specified text using the active translation file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ext</td>
<td>Translation</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported (see &quot;Noted&quot; below)</td>
</tr>
</tbody>
</table>

**Syntax**

`Ext(strText)`

**strText**

The text to be translated.

**Returned value**

Returns the text translation using the active translation file.

**Notes**

If this function is called on Mobile Access, it will not fail but the translation will not be performed.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>Ext(&quot;Start&quot;)</code> // Returned value if translating to Portuguese = &quot;Iniciar&quot;</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td><code>Ext(&quot;Stop&quot;)</code> // Returned value if translating to German = &quot;Anschlag&quot;</td>
<td></td>
</tr>
</tbody>
</table>

**SetLanguage**

The function `SetLanguage` sets the project translation to one of the languages configured in the Translation Table.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetLanguage</td>
<td>Translation</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`SetLanguage(numLanguageID)`

**numLanguageID**

The locale ID for the language that you want to set as the current translation. The language must already be configured in the Translation Table.

**Returned value**

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Success</td>
</tr>
</tbody>
</table>

### Notes

Locale IDs are the proprietary language/region codes used in Microsoft Windows. For more information, go to: [msdn.microsoft.com/library/ms912047](https://msdn.microsoft.com/library/ms912047)

### Examples

Set the language to "French – France":

**SetLanguage(1036)**

Set the language to "Portuguese – Brazil":

**SetLanguage(1046)**

**SetTranslationFile**

Sets the active translation file and translates all enabled text within the project.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetTranslationFile</td>
<td>Translation</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
SetTranslationFile(strFileName{   | , optStrColumnName})
```

- **strFileName**
  
  The name of a translation file

- **optStrColumnName**
  
  The name of the column from the translation file, which must be used to translate the texts in the project. When omitted, the second column from the translation file will be used by default.

### Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>-3</td>
<td>Translation file could not be found or opened.</td>
</tr>
</tbody>
</table>

### Examples

**Tag**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SetTranslationFile( &quot;Portuguese.tra&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>SetTranslationFile( &quot;German.tra&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>SetTranslationFile( &quot;Mytranslation.csv&quot;, &quot;Portuguese&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>SetTranslationFile( &quot;Mytranslation.csv&quot;, &quot;German&quot; )</td>
</tr>
</tbody>
</table>

**Note:** You must enable the Translation option from the **Project Settings** dialog for this function to work.
Caution: You must have a translation file in the Translation Tool.

TranslationLoad

This function loads an external translation file and either replaces or appends to the current translation.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TranslationLoad</td>
<td>Translation</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

TranslationLoad( strFileName, numAppend )

strFileName

The file path and name of the translation file (*.TRA or *.CSV) to be loaded. If you specify a relative path, the function will look first in the project folder and then in the Web sub-folder. If you are running your application on the Web, you must place the file in the Web sub-folder and specify a relative path.

numAppend

A numeric flag that indicates whether the loaded translation file replaces or appends to the existing contents of the Translation Table. Specify one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Completely replace the contents Translation Table and replace with the contents of the loaded translation file.</td>
</tr>
<tr>
<td>1</td>
<td>Keep the contents of the Translation Table, and then append the contents of the loaded translation file only if they are not already present.</td>
</tr>
<tr>
<td>2</td>
<td>Keep the contents of the Translation Table, and then append the contents of the loaded translation file, replacing any duplicates.</td>
</tr>
<tr>
<td>3</td>
<td>Reset the Translation Table to its initial contents when the project was run. If this value is specified, strFileName is ignored.</td>
</tr>
</tbody>
</table>

Returned value

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>Failed to load the translation file. The file format is invalid or you do not have the necessary privileges.</td>
</tr>
<tr>
<td>-2</td>
<td>The specified file cannot be found.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid parameters.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>
Examples

TranslationLoad( "alternate.tra", 0 )

TranslationLoad( "Web\pt-BR.csv", 1 )

TranslationLoad( ",", 3 )

TranslationLookupClose

The function TranslationLookupClose closes a lookup map that was previously loaded into memory by the function TranslationLookupLoad.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TranslationLookupClose</td>
<td>Translation</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

TranslationLookupClose(numTranslationLookupID)

numTranslationLookupID

The ID number of the lookup map that you want to close, as it was returned by the function TranslationLookupLoad. Valid values range from 0 to 255.

Returned value

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Lookup map successfully closed.</td>
</tr>
<tr>
<td>-1</td>
<td>Function executed from the Viewer module, or other internal error.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid parameter(s).</td>
</tr>
<tr>
<td>-3</td>
<td>Other error.</td>
</tr>
</tbody>
</table>

Notes

Closing a lookup map frees the system resources used by that map. You should always close a lookup map when you are done with it.

Examples

Close the lookup map specified by the ID number stored in the tag MyLookupID:

TranslationLookupClose(MyLookupID)

TranslationLookupGet

The function TranslationLookupGet gets the translation of a specified source text, using a lookup map that was previously loaded into memory by the function TranslationLookupLoad.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TranslationLookupGet</td>
<td>Translation</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

TranslationLookupGet(numTranslationLookupID, strSource)

numTranslationLookupID

The ID number of the lookup map that you want to use, as it was returned by the function TranslationLookupLoad. Valid values range from 0 to 255.

strSource

The source text for which you want to get the translation.

Returned value

If this function is successfully executed, it returns the translation of the specified source text. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strSource</td>
<td>Source text returned untranslated, because it was not found in the lookup map.</td>
</tr>
<tr>
<td>-1</td>
<td>Function executed from the Viewer module, or other internal error.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid parameter(s).</td>
</tr>
<tr>
<td>-3</td>
<td>Other error.</td>
</tr>
</tbody>
</table>

Examples

Get the translation of "Tank #1", using the lookup map specified by the ID number stored in the tag MyLookupID:

TranslationLookupGet(MyLookupID, "Tank #1")

Load a lookup map for translating from English (United States) to Portuguese (Brazil), and then use it to get the translation of "Tank #1":

TranslationLookupGet(TranslationLookupLoad(1033, 1046), "Tank #1")

TranslationLookupLoad

The function TranslationLookupLoad creates a lookup map from the Translation Table for the specified source and target languages, and then it loads the map into memory so that it can be used by the function TranslationLookupGet.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TranslationLookupLoad</td>
<td>Translation</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

TranslationLookupLoad(numSourceLanguageID, numTargetLanguageID)

numSourceLanguageID

numTargetLanguageID

The locale IDs for the source and target language, respectively. Both languages must already be configured in the Translation Table.
Returned value

If this function is successfully executed, it returns an ID number (from 0 to 255) for the lookup map. You can then use the ID number to call the functions `TranslationLookupGet` and `TranslationLookupClose`.

Otherwise, this function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Function executed from the Viewer module, or other internal error.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid parameter(s).</td>
</tr>
<tr>
<td>-3</td>
<td>Cannot have more than 256 lookup maps loaded in memory at the same time.</td>
</tr>
<tr>
<td>-4</td>
<td>The source language is the same as the target language, or the specified languages have not been configured in the Translation Table.</td>
</tr>
</tbody>
</table>

Notes

When you call this function, you can specify any two languages that have been configured in the Translation Table, even if they were both originally configured as target languages. When the lookup map is created, it automatically cross-references the specified languages. For example, if you configured the Translation Table for your project to include both an English-to-Spanish translation and an English-to-Portuguese translation, you can then specify Spanish and Portuguese when you call this function.

You can have up to 256 lookup maps loaded in memory at the same time, but each map uses system resources and that can affect run-time performance. You should use the function `TranslationLookupClose` to close a map when you are done with it.

Locale IDs are proprietary language/region codes used in Microsoft Windows. For more information, go to: [msdn.microsoft.com/library/ms912047](msdn.microsoft.com/library/ms912047)

Examples

Load a lookup map for translating from English (United States) to Portuguese (Brazil):

```plaintext
TranslationLookupLoad(1033,1046)
```
Multimedia functions

These functions are used to play external audio and video files.

**Play**

Plays a specified WAV audio file.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Play</td>
<td>Multimedia</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Note:** For this function to work on a Thin Client, the target WAV file must be located in the same file path on the remote station.

**Description**

Plays a specified WAV audio file.

**Syntax**

\[
\text{Play}(\text{strFileName} \ |
\left\{\begin{array}{c}
\text{optNumSynchronous} \\
\end{array}\right\})
\]

**strFileName**

The file path and name of the WAV file to play.

**optNumSynchronous**

A numeric flag specifying whether the function executes synchronously or asynchronously:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Asynchronous (i.e., the project continues without waiting for the function to return)</td>
</tr>
<tr>
<td>1</td>
<td>Synchronous (i.e., the project pauses while it waits for the function to return)</td>
</tr>
</tbody>
</table>

This is an optional parameter; if no value is specified, then the default is 0.

**Returned value**

This function does not return any value.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>\text{Play}(\ &quot;C:\ Sounds\ Wav\ alarm.wav&quot; )</td>
</tr>
</tbody>
</table>
System Info functions

These functions are used to get information about the computer that is running the project (either server or client, depending on the function), as well as to change some project settings on that computer.

**DBVersion**

*DbVersion* is a built-in scripting function that gets the version number of your project tags database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBVersion</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

*DBVersion()*

This function takes no parameters.

**Returned value**

This function returns a numerical value equal to the version number of the database.

**Notes**

This function only applies to the native database within your project. There currently is no function to get the version number of an external or historical database.

**Examples**

*DBVersion()*

**GetAppHorizontalResolution**

Gets the default horizontal screen resolution (in pixels) of the project.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAppHorizontalResolution()</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

*GetAppHorizontalResolution()*

This function takes no parameters.

**Returned value**

Returns the default value as it is stored in the project file, but does not test the Windows configuration.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetAppHorizontalResolution()</td>
<td>640</td>
</tr>
<tr>
<td>Tag</td>
<td>GetAppHorizontalResolution()</td>
<td>800</td>
</tr>
</tbody>
</table>
**GetAppPath**

Returns the file path of the project folder.

**Function**
- Group: System Info
- Execution: Synchronous
- Windows: Supported
- Embedded: Supported
- Thin Client: Executed on Server
- Mobile Access: Supported

**Syntax**

`GetAppPath()`

This function takes no parameters.

**Returned value**

Returns the file path as a string.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>GetAppPath()</code> // Returned value = &quot;C:\DemoProject&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td><code>GetAppPath()</code> // Returned value = &quot;C:\Studio\Projects\project_name&quot;</td>
</tr>
</tbody>
</table>

**Note:** This function must return the current path of the project, including the "\" at the end of the path.

**GetAppVerticalResolution**

Gets the default vertical screen resolution (in pixels) of the project.

**Function**
- Group: System Info
- Execution: Synchronous
- Windows: Supported
- Embedded: Not supported
- Thin Client: Supported
- Mobile Access: Supported

**Syntax**

`GetAppVerticalResolution()`

This function takes no parameters.

**Returned value**

Returns the default value as it is stored in the project file, but does not test the Windows configuration.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>GetAppVerticalResolution()</code> // Returned value = 480</td>
</tr>
<tr>
<td>Tag</td>
<td><code>GetAppVerticalResolution()</code> // Returned value = 600</td>
</tr>
</tbody>
</table>

**GetComputerIP**

Returns the first IP Address of the local computer.

**Function**
- Group: System Info
- Execution: Synchronous
- Windows: Supported
- Embedded: Supported
- Thin Client: Supported
- Mobile Access: Not supported
**Syntax**

GetComputerIP()
This function takes no parameters.

**Returned value**

Returns the first IP Address of the local station as a string.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetComputerIP() // Returned value = &quot;192.168.0.1&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>GetComputerIP() // Returned value = &quot;248.12.2.78&quot;</td>
</tr>
</tbody>
</table>

**GetComputerName**

Returns the local computer name.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetComputerName()</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetComputerName()
This function takes no parameters.

**Returned value**

Returns the local computer name as a string.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetComputerName() // Returned value = &quot;Terminal53&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>GetComputerName() // Returned value = &quot;BobsComputer&quot;</td>
</tr>
</tbody>
</table>

**GetCursorX**

Gets the X-coordinate of the mouse cursor on the screen.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCursorX</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetCursorX()
This function takes no parameters.

**Returned value**

This function returns the X-coordinate of the cursor on the screen, or −1 if an error occurs.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GetCursorX( ) // Returned value = 1024</td>
</tr>
</tbody>
</table>
**GetCursorY**

Gets the Y-coordinate of the mouse cursor on the screen.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCursorY</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetCursorY()

This function takes no parameters.

**Returned value**

This function returns the Y-coordinate of the cursor on the screen, or −1 if an error occurs.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GetCursorY() // Returned value = 768</td>
</tr>
</tbody>
</table>

**GetDisplayHorizontalResolution**

Gets the horizontal resolution (in pixels) of the display connected to the local station.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDisplayHorizontalResolution</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetDisplayHorizontalResolution()

This function takes no parameters.

**Returned value**

This function returns the horizontal resolution of the display as an integer.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GetDisplayHorizontalResolution() // Returned value = 1024</td>
</tr>
</tbody>
</table>

**GetDisplayVerticalResolution**

Gets the vertical resolution (in pixels) of the display connected to the local station.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDisplayVerticalResolution</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetDisplayVerticalResolution()

This function takes no parameters.

**Returned value**

This function returns the vertical resolution of the display as an integer.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GetDisplayVerticalResolution( ) // Returned value = 768</td>
</tr>
</tbody>
</table>

GetFreeMemoryCE

Returns the free memory available in a Windows Embedded device.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFreeMemoryCE</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Not supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

GetFreeMemoryCE({ | optNumType })

optNumType

A numeric flag that specifies which type of free memory the project should retrieve from the device:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Total free program memory</td>
</tr>
</tbody>
</table>

This is an optional parameter; if no value is specified, then the default is 0.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0</td>
<td>Size of free memory in bytes.</td>
</tr>
<tr>
<td>-1</td>
<td>Coredll.dll file not found.</td>
</tr>
<tr>
<td>-2</td>
<td>GetMemoryCE function not found.</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid optional parameter.</td>
</tr>
<tr>
<td>-4</td>
<td>Type of memory unavailable.</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GetFreeMemoryCE()</td>
</tr>
</tbody>
</table>

GetHardKeyModel

Returns the model name of your Hardkey.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHardKeyModel</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Syntax

GetHardKeyModel()

This function takes no parameters.

Returned value

If the Hardkey is installed, then the function returns a string with the Hardkey model name.

If the Hardkey is not installed, then the function returns 0.
**Notes**
You must attach the Hardkey before executing this function, or the function will not execute properly.

**Examples**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetHardKeyModel()  // Returned value = &quot;Local Interface&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>GetHardKeyModel()  // Returned value = &quot;Advanced Server&quot;</td>
</tr>
</tbody>
</table>

**GetHardKeySN**

Returns the serial number of the Hardkey.

**Syntax**

GetHardKeySN()

This function takes no parameters.

**Returned value**

If the Hardkey is installed, then the function returns a string with the Hardkey serial number.

If the Hardkey is not installed, then the function returns 0.

**Notes**
You must attach the Hardkey before executing this function, or the function will not execute properly.

**Examples**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetHardkeySN()  // Returned value = 120.745</td>
</tr>
<tr>
<td>Tag</td>
<td>GetHardkeySN()  // Returned value = 224.941</td>
</tr>
</tbody>
</table>

**GetIPAll**

Returns the number of IP Addresses assigned to the local station and stores the IP Addresses in a string array tag.

**Syntax**

GetIPAll( "tagArrayIP", optRefresh )

tagArrayIP

Name of the string array tag receiving the IP addresses found. This name must be enclosed in quotes, or the project will try to get the contents of the array tag.

optRefresh

*Optional* tag that triggers a refresh of this function, if you use it in a Text Data Link animation. Every time this tag changes value, the project will refresh the function.
**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid parameter type</td>
</tr>
<tr>
<td>( n )</td>
<td>Number of IP addresses found</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>GetIPAll( &quot;TagArrayIP&quot; )</code></td>
<td>( = 1 )</td>
</tr>
<tr>
<td>Tag</td>
<td><code>GetIPAll( &quot;TagArrayIP&quot;, Second )</code></td>
<td>( = 2 )</td>
</tr>
</tbody>
</table>

**GetLicenseGracePeriod**

This function returns the number of minutes remaining until the license grace period expires.

**Syntax**

```c
GetLicenseGracePeriod( )
```

This function has no parameters.

**Returned value**

If this function is executed successfully, then it returns the number of minutes remaining in the license grace period. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>The software is licensed.</td>
</tr>
<tr>
<td>0</td>
<td>The license grace period has expired. In most cases, this value will never</td>
</tr>
<tr>
<td></td>
<td>actually be returned because if the grace period has expired, then the</td>
</tr>
<tr>
<td></td>
<td>project has stopped.</td>
</tr>
</tbody>
</table>

**Notes**

The license grace period is relevant only if the software (i.e., the development application and/or the project runtime) is configured to get its license from a license server and it has somehow lost its connection to that server. For more information, see *About License Server* on page 58.

By default, the license grace period is 60 minutes. If you need to change the period, then please contact your InduSoft Web Studio software vendor.

Also, this function may be called by a Screen Script or Command animation on a specific thin client, but the returned value applies to the entire project runtime. Thin clients do not have their own licenses. Instead, the project runtime is licensed to support a number of thin clients, and the thin client on which the function is executed is included in that number.
Example
The following VBScript example shows how to check the grace period and then display an appropriate message to the user:

```vbnet
Dim remainingGracePeriod
remainingGracePeriod = $GetLicenseGracePeriod()
If (remainingGracePeriod = -1) Then
    MsgBox "License OK"
Else
    MsgBox "Running on Grace Period! [" + remainingGracePeriod + "] minute(s) remaining"
End If
```

**GetMemoryCE**

Returns the total memory available in a Windows Embedded device.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetMemoryCE</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Not supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetMemoryCE({ | optNumType |})

**optNumType**

A numeric flag that specifies which type of memory the project should retrieve from the device:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Total program memory</td>
</tr>
<tr>
<td>1</td>
<td>Total storage memory</td>
</tr>
<tr>
<td>2</td>
<td>Total memory</td>
</tr>
</tbody>
</table>

This is an optional parameter; if no value is specified, then the default is 0.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0</td>
<td>Size of memory in bytes.</td>
</tr>
<tr>
<td>-1</td>
<td>Coredll.dll file not found.</td>
</tr>
<tr>
<td>-2</td>
<td>GetMemoryCE function not found.</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid optional parameter.</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td></td>
<td>GetMemoryCE( 1 )</td>
</tr>
</tbody>
</table>

**GetNetMACID**

Gets the MAC ID unique code from the currently installed network adapter(s).
Syntax

\texttt{GetNetMACID( "optTagMACID", "optTagAdapterName" )}

\textbf{optStrMACID}
Name of a string tag, which receives the MAD ID of the network adapter. If there is more than one network adapter currently installed in the station, the user can configure a string array tag in this parameter, so each array position receives the MAC ID from one network adapter.

\textbf{optStrAdapterName}
Name of a string tag, which receives the name of the network adapter. If there is more than one network adapter currently installed in the station, the user can configure a string array tag in this parameter, so each array position receives the name from one network adapter. This parameter is optional.

\textbf{Returned value}

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0</td>
<td>Number of network adapters found.</td>
</tr>
<tr>
<td>0</td>
<td>No network adapters found.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>One of the parameters is not string type.</td>
</tr>
<tr>
<td>-3</td>
<td>Tag configured in optTagMACID does not exist.</td>
</tr>
<tr>
<td>-4</td>
<td>Tag configured in optTagAdapterName does not exist.</td>
</tr>
</tbody>
</table>

\textbf{Examples}

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumNIC</td>
<td>\texttt{GetNetMACID(&quot;MACIDTag&quot;)}</td>
</tr>
<tr>
<td>NumNIC</td>
<td>\texttt{GetNetMACID(&quot;MACIDTag&quot;, &quot;AdapterName&quot;)}</td>
</tr>
<tr>
<td>NumNIC</td>
<td>\texttt{GetNetMACID(&quot;MACIDTag[1]&quot;, &quot;AdapterName[1]&quot;)}</td>
</tr>
</tbody>
</table>

\textbf{GetOS}

The function \texttt{GetOS} gets the operating system of the computer that hosts the project runtime server.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetOS</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

\textbf{Syntax}

\texttt{GetOS()}

\texttt{GetOS()}
This function takes no parameters.

\textbf{Returned value}
This function returns one of the following possible values:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2     | Microsoft Windows or Windows Embedded Standard, which includes the following:  
  - Microsoft Windows XP Service Pack 3  
  - Microsoft Windows Vista Service Pack 2  
  - Microsoft Windows 7 Service Pack 1  
  - Microsoft Windows 8 or 8.1 (excluding Windows RT)  
  - Microsoft Windows Server 2003 Service Pack 2  
  - Microsoft Windows Server 2008 Service Pack 2  
  - Microsoft Windows Server 2008 R2 Service Pack 1  
  - Microsoft Windows Server 2012  
  - Microsoft Windows XP Embedded Service Pack 3  
  - Microsoft Windows Embedded Standard 7 |

| 3     | Windows Embedded Compact, which includes the following:  
  - Microsoft Windows Mobile or Windows CE 5.x/6.x  
  - Microsoft Windows Embedded Compact 7 |

**GetPerformanceMetric**

The `GetPerformanceMetric` function returns selected metrics for the graphics performance of your project runtime client.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPerformanceMetric</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
GetPerformanceMetric( numMetrictype, "optTagDescription"
)
```

GetPerformanceMetric({ numMetrictype | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 } | "optTagDescription" )

**numMetrictype**

The type of performance metric to get, as shown in the table below:

<table>
<thead>
<tr>
<th>Value of numMetrictype</th>
<th>Description, including the string that is written to optTagDescription</th>
</tr>
</thead>
</table>
| 0                      | Memory Allocation Enabled  
  How the virtual memory is currently allocated to shared objects: 0 = Disabled; 1 = Always; 2 = Dynamic; 3 = Critical.  
  **Disabled**  
  The Enable memory allocation option (Project Settings > Options > Performance Control) is not selected. Objects are loaded into memory only when they are actually used, and then they are immediately removed afterwards.  
  **Always**  
  Objects are always kept in memory, in order to increase run-time performance.  
  **Dynamic**  
  Objects are kept in memory until the memory is needed for new objects. If enough memory is freed, then the memory allocation state may return from Dynamic to Always. |
<table>
<thead>
<tr>
<th>Value of <code>numMetricType</code></th>
<th>Description, including the string that is written to <code>optTagDescription</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>All unused objects are cleared from memory and memory allocation is disabled until the project is restarted. This is effectively the same as Disabled (see above), except that the <strong>Enable memory allocation</strong> option is still selected.</td>
</tr>
<tr>
<td>1</td>
<td>Critical Free Memory Limit (KB) When the amount of free virtual memory (in KB) decreases to this limit, the memory allocation state is changed from Dynamic to Critical. This limit is set in <strong>Project Settings &gt; Options &gt; Performance Control</strong>.</td>
</tr>
<tr>
<td>2</td>
<td>Min Free Memory Limit (KB) When the amount of free virtual memory (in KB) decreases to this limit, the memory allocation state is changed from Always to Dynamic. This limit is set in <strong>Project Settings &gt; Options &gt; Performance Control</strong>.</td>
</tr>
<tr>
<td>3</td>
<td>Free Memory (KB) The amount of free virtual memory, in kilobytes.</td>
</tr>
<tr>
<td>4</td>
<td>Free Memory (%) The amount of free virtual memory, as a percentage of total virtual memory.</td>
</tr>
<tr>
<td>5</td>
<td>Total Allocation Allowed The maximum number of total shared objects that can be allocated.</td>
</tr>
<tr>
<td>6</td>
<td>Total Allocation Count The number of total shared objects that are currently allocated.</td>
</tr>
<tr>
<td>7</td>
<td>Total Allocation (%) The percentage ( <code>count / allowed</code> ) of total shared objects that are allocated.</td>
</tr>
<tr>
<td>8</td>
<td>Brush Allocation Allowed The maximum number of shared brush objects that can be allocated.</td>
</tr>
<tr>
<td>9</td>
<td>Brush Allocation Count The number of shared brush objects that are currently allocated.</td>
</tr>
<tr>
<td>10</td>
<td>Brush Allocation (%) The percentage ( <code>count / allowed</code> ) of shared brush objects that are allocated.</td>
</tr>
<tr>
<td>11</td>
<td>Brush Allocation State How the memory for shared brush objects is currently allocated: 0 = Disabled; 1 = Always; 2 = Dynamic; 3 = Critical.</td>
</tr>
<tr>
<td>12</td>
<td>Font Allocation Allowed The maximum number of shared font objects that can be allocated.</td>
</tr>
<tr>
<td>13</td>
<td>Font Allocation Count The number of shared font objects that are currently allocated.</td>
</tr>
<tr>
<td>14</td>
<td>Font Allocation (%) The percentage ( <code>count / allowed</code> ) of shared font objects that are allocated.</td>
</tr>
<tr>
<td>15</td>
<td>Font Allocation State How the memory for shared font objects is currently allocated: 0 = Disabled; 1 = Always; 2 = Dynamic; 3 = Critical.</td>
</tr>
<tr>
<td>16</td>
<td>Pen Allocation Allowed</td>
</tr>
<tr>
<td>Value of numMetrictype</td>
<td>Description, including the string that is written to optTagDescription</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>17 Pen Allocation Count</td>
<td>The maximum number of shared pen objects that can be allocated.</td>
</tr>
<tr>
<td>18 Pen Allocation (%)</td>
<td>The number of shared pen objects that are currently allocated.</td>
</tr>
<tr>
<td>19 Pen Allocation State</td>
<td>How the memory for shared pen objects is currently allocated: 0 = Disabled; 1 = Always; 2 = Dynamic; 3 = Critical.</td>
</tr>
<tr>
<td>20 Bitmap Allocation (%)</td>
<td>The percentage (count / allowed) of shared bitmap objects that are allocated.</td>
</tr>
<tr>
<td>21 Bitmap Allocation Count</td>
<td>The number of shared bitmap objects that are currently allocated.</td>
</tr>
<tr>
<td>22 Bitmap Allocation Allowed</td>
<td>The maximum number of shared bitmap objects that can be allocated.</td>
</tr>
<tr>
<td>23 Bitmap Allocation State</td>
<td>How the memory for shared bitmap objects is currently allocated: 0 = Disabled; 1 = Always; 2 = Dynamic; 3 = Critical.</td>
</tr>
<tr>
<td>24 Image Allocation Allowed</td>
<td>The maximum number of shared image objects that can be allocated.</td>
</tr>
<tr>
<td>25 Image Allocation Count</td>
<td>The number of shared image objects that are currently allocated.</td>
</tr>
<tr>
<td>26 Image Allocation (%)</td>
<td>The percentage (count / allowed) of shared image objects that are allocated.</td>
</tr>
<tr>
<td>27 Image Allocation State</td>
<td>How the memory for shared image objects is currently allocated: 0 = Disabled; 1 = Always; 2 = Dynamic; 3 = Critical.</td>
</tr>
<tr>
<td>28 Buffer Allocation Allowed</td>
<td>The maximum number of shared buffer objects that can be allocated.</td>
</tr>
<tr>
<td>29 Buffer Allocation Count</td>
<td>The number of shared buffer objects that are currently allocated.</td>
</tr>
<tr>
<td>30 Buffer Allocation (%)</td>
<td>The percentage (count / allowed) of shared buffer objects that are allocated.</td>
</tr>
<tr>
<td>31 Buffer Allocation State</td>
<td>How the memory for shared buffer objects is currently allocated: 0 = Disabled; 1 = Always; 2 = Dynamic; 3 = Critical.</td>
</tr>
</tbody>
</table>

**optTagDescription**

The name of a String tag that will receive a description of the selected metric. The description is the same as shown in the table for numMetrictype above.
Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This parameter is optional.

Returned value

If this function is successfully executed, then it returns a positive value according to the type of metric that was selected (see `numMetricType` above). Otherwise, if it was not successfully executed, then it returns a negative error value:

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>Wrong parameter type.</td>
</tr>
<tr>
<td>-3</td>
<td>Could not get the selected metric.</td>
</tr>
<tr>
<td>-4</td>
<td>Tag specified by <code>optTagDescription</code> does not exist.</td>
</tr>
<tr>
<td>-5</td>
<td>Could not write to tag specified by <code>optTagDescription</code>.</td>
</tr>
</tbody>
</table>

Notes

For more information about memory allocation, see [Configure the performance control settings](#) on page 101.

GetPrivateProfileString

Reads a specified parameter from an `.ini` file using the standard `.ini` format.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPrivateProfileString</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

GetPrivateProfileString( `strSection`, `strName`, `strDefault`, `strFileName` )

`strSection`

The section name to be read.

`strName`

The parameter name to be read.

`strDefault`

The default setting for this parameter. If the parameter is not found in the `.ini` file, the function will return this default setting.

`strFileName`

The path and name of the `.ini` file to be read.

Returned value

Returns the value of the specified parameter.

Examples

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetPrivateProfileString( &quot;boot loader&quot;, &quot;timeout&quot;, &quot;50&quot;, &quot;C:\boot.ini&quot; ) // Returned value = 30</td>
</tr>
</tbody>
</table>
**GetProductPath**

Gets the path to the program directory.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetProductPath</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetProductPath()

This function takes no parameters.

**Returned value**

Returns the path to the program directory as a string.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetProductPath() // Returned value = &quot;C:\Program Files\InduSoft Web Studio v7.1&quot;</td>
</tr>
</tbody>
</table>

**GetRegValue**

Gets the value of a variable in the Windows registry.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetRegValue</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetRegValue( numMainKey, strKey, strValueName )

**numMainKey**

Numeric tag with the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HKEY_LOCAL_MACHINE</td>
</tr>
<tr>
<td>1</td>
<td>HKEY_CLASSES_ROOT</td>
</tr>
<tr>
<td>2</td>
<td>HKEY_CURRENT_USER</td>
</tr>
<tr>
<td>3</td>
<td>HKEY_USERS</td>
</tr>
<tr>
<td>4</td>
<td>HKEY_CURRENT_CONFIG</td>
</tr>
<tr>
<td>5</td>
<td>HKEY_PERFORMANCE_DATA</td>
</tr>
</tbody>
</table>

**strKey**

Path where the value is located in the Main Key.

**strVariableName**

Name of the variable to get. The maximum length is 255 characters.

**Returned value**

If the function succeeds, then the function returns the variable value. Otherwise one of the following error codes will be returned:

<table>
<thead>
<tr>
<th>Code</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Invalid number of parameters or invalid Main Key.</td>
</tr>
</tbody>
</table>
−2 Variable type is not supported. You can only read DWord or String values from the registry.

−3 Failed to read the variable value; verify that you have the proper security rights.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetRegValue( 0, &quot;HARDWARE\DESCRIPTION\System&quot;, &quot;SystemBiosDate&quot; ) // Returned value = &quot;08/14/03&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>GetRegValue( 2, &quot;Control Panel\Current&quot;, &quot;Color Schemes&quot; ) // Returned value = &quot;Windows Standard&quot;</td>
</tr>
</tbody>
</table>

**GetRegValueType**

Gets the data type of the value of a variable in the Windows registry.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetRegValueType</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetRegValueType( numMainKey, strKey, strVariableName )

**numMainKey**

Numeric tag with the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Main Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HKEY_LOCAL_MACHINE</td>
</tr>
<tr>
<td>1</td>
<td>HKEY_CLASSES_ROOT</td>
</tr>
<tr>
<td>2</td>
<td>HKEY_CURRENT_USER</td>
</tr>
<tr>
<td>3</td>
<td>HKEY_USERS</td>
</tr>
<tr>
<td>4</td>
<td>HKEY_CURRENT_CONFIG</td>
</tr>
<tr>
<td>5</td>
<td>HKEY_PERFORMANCE_DATA</td>
</tr>
</tbody>
</table>

**strKey**

Path where the value is located in the Main Key.

**strVariableName**

Name of the variable to get. The maximum length is 255 characters.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Variable type is String.</td>
</tr>
<tr>
<td>0</td>
<td>Variable type is DWord.</td>
</tr>
<tr>
<td>−1</td>
<td>Invalid number of parameters or invalid Main Key.</td>
</tr>
<tr>
<td>−2</td>
<td>Variable type is not supported. You can only read DWord or String values from the registry.</td>
</tr>
<tr>
<td>−3</td>
<td>Failed to read the variable value; verify that you have the proper security rights.</td>
</tr>
</tbody>
</table>
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetRegValueType( 0, &quot;HARDWARE\DESCRIPTION\System&quot;, &quot;SystemBiosDate&quot; ) // Returned value = 1</td>
</tr>
<tr>
<td>Tag</td>
<td>GetRegValueType( 2, &quot;Control Panel\Desktop&quot;, &quot;Smooth Scroll&quot; ) // Returned value = 0</td>
</tr>
</tbody>
</table>

**GetServerHostName**

Gets the host name of the project’s Server station.

- **Syntax**
  
  GetServerHostName()

  This function takes no parameters.

- **Returned value**
  
  Server host name for ISSymbol and 127.0.0.1 for others.

**GetTickCount**

Gets the current value of the clock ticks counter.

- **Syntax**
  
  GetTickCount()

  This function takes no parameters.

- **Returned value**
  
  Returns an integer with the number of milliseconds counted by the clock for each initialization of the operational system.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetTickCount() // Returned value = 9400907</td>
</tr>
</tbody>
</table>

**InfoAppAlrDir**

Returns the file path of the project’s Alarm sub-folder.

- **Syntax**
  
  InfoAppAlrDir()

  This function takes no parameters.
Returned value

Returns the *Alarm* directory of the current project as a string.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>InfoAppAlrDir() // Returned value = &quot;C:\DemoApp\Alarm&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>InfoAppAlrDir() // Returned value = &quot;C:\Studio\Projects\App\Alarm&quot;</td>
</tr>
</tbody>
</table>

**InfoAppHstDir**

Returns the file path of the project's History sub-folder.

**Syntax**

InfoAppHstDir()

This function takes no parameters.

**Returned value**

Returns the *History* directory for the current project as a string.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>InfoAppHSTDir() // Returned value = &quot;C:\DemoApp\HST&quot;</td>
</tr>
<tr>
<td>Tag</td>
<td>InfoAppHSTDir() // Returned value = &quot;C:\Studio\Projects\App\HST&quot;</td>
</tr>
</tbody>
</table>

**InfoDiskFree**

Returns free disk space on the local computer.

**Syntax**

InfoDiskFree(strDisk)

*strDisk*  
The name of the disk volume to be checked.

**Returned value**

Returns disposable free space in the disk in bytes.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>InfoDiskFree( &quot;C&quot; ) // Returned value = 2803804605.000000</td>
</tr>
</tbody>
</table>
**InfoResources**

Returns the local computer's disposable resources.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>InfoResources</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
InfoResources(numSelect)
```

**numSelect**

A numeric flag that specifies which resource to examine:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>System functions (%)</td>
</tr>
<tr>
<td>1</td>
<td>GDI functions (%)</td>
</tr>
<tr>
<td>2</td>
<td>USER functions (%)</td>
</tr>
<tr>
<td>3</td>
<td>Memory (in bytes)</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>InfoResources(0)</td>
<td>76.000000</td>
</tr>
<tr>
<td>Tag</td>
<td>InfoResources(1)</td>
<td>76.000000</td>
</tr>
<tr>
<td>Tag</td>
<td>InfoResources(2)</td>
<td>80.000000</td>
</tr>
<tr>
<td>Tag</td>
<td>InfoResources(3)</td>
<td>16150528.000000</td>
</tr>
</tbody>
</table>

**Note:** The only valid selection on a Windows PC station is 3. Selecting 0, 1 or 2 returns 0.000000 only.

**IsActiveXReg**

Determines whether an ActiveX control is registered with the operating system.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsActiveXReg</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
IsActiveXReg(numType, strProgIDorFileName)
```

**numType**

A numeric flag that specifies a format for the strProgIDorFileName parameter:

<table>
<thead>
<tr>
<th>numType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Verify by Program ID</td>
</tr>
<tr>
<td>1</td>
<td>Verify by File Name</td>
</tr>
</tbody>
</table>

**strProgIDorFileName**

The program ID or file path of the ActiveX control.
Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ActiveX is not registered.</td>
</tr>
<tr>
<td>1</td>
<td>ActiveX is registered.</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>IsActiveXReg( 0, &quot;ISSYMBOL.ISSymbolCtrl.1&quot; ) // Returned value = 0</td>
</tr>
<tr>
<td>Tag</td>
<td>IsActiveXReg( 1, &quot;C:\WinNT\system32\MediaPlayer.ocx&quot; ) // Returned value = 1</td>
</tr>
</tbody>
</table>

IsAppChangedOnServer

The function IsAppChangedOnServer checks whether the project files available on the server are newer than the files currently on the client.

Syntax

IsAppChangedOnServer("optTagUpdateTrigger")

IsAppChangedOnServer({ "optTagUpdateTrigger" })

**optTagUpdateTrigger**

The name of a project tag that will server as a trigger. When the value of the specified tag changes, the function is executed. Unlike most other functions, once this function is called, it is kept in memory until the project is stopped.

To execute the function at regular intervals, use one of the system tags like Day or Month.

This parameter is optional; if no value is specified, the function is executed immediately and not kept in memory.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FALSE (i.e., the project files on the server have not changed)</td>
</tr>
<tr>
<td>1</td>
<td>TRUE (i.e., the project files on the server have changed)</td>
</tr>
</tbody>
</table>

If this function returns TRUE, you can use the function ReloadAppFromServer to update the client.

Notes

For this function, the server is the computer or device that hosts the downloadable project files (e.g., screens) for your thin clients. It might be different from the project runtime server (a.k.a. the data server), depending on how you deploy your project. For more information, see Configuring a Web Solution.

If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from
background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

**Examples**
Immediatly check whether the project files have changed:

```vb
IsAppChangedOnServer()
```

When the value of the tag `CheckVersion` changes, check whether the project files have changed:

```vb
IsAppChangedOnServer("CheckVersion")
```

**NoInputTime**
Returns the time elapsed since the last keyboard action.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoInputTime</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Keyboard input only</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```vb
NoInputTime( "optTagUpdateTrigger" )
```

```vb
NoInputTime( "optTagUpdateTrigger" )
```

**optTagUpdateTrigger**
The name of a project tag. When the value of the tag changes, it triggers an update when this function is used in a Text Data Link animation.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**Returned value**
Returns the time (in seconds) since the last keyboard action.

**Notes**
If VBScript debugging is enabled, then this function cannot be used; the normal execution cycle is suspended during debugging, so it is not possible to accurately measure the time elapsed without user input. For more information, see Debugging VBScript on page 983.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>NoInputTime()</td>
</tr>
</tbody>
</table>

**Note:** You cannot implement this function directly from a Text object.
**ProductVersion**

Returns the program version number.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductVersion</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

ProductVersion()

This function takes no parameters.

**Returned value**

Returns the program version number as a real number.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ProductVersion() // Returned value = 7.000000</td>
</tr>
</tbody>
</table>

**RegSaveCE**

Saves the Windows Embedded system registry. This function will only work if the save registry capability is enabled in the Windows Embedded device image.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegSaveCE</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

RegSaveCE()

This function takes no parameters.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success.</td>
</tr>
<tr>
<td>-1</td>
<td>Failed to save HKEY_CLASSES_ROOT.</td>
</tr>
<tr>
<td>-2</td>
<td>Failed to save HKEY_CURRENT_USER.</td>
</tr>
<tr>
<td>-3</td>
<td>Failed to save HKEY_LOCAL_MACHINE.</td>
</tr>
<tr>
<td>-4</td>
<td>Failed to save HKEY_USERS.</td>
</tr>
<tr>
<td>-5</td>
<td>Function executed in NT platform.</td>
</tr>
</tbody>
</table>

**Note:** This function calls the RegFlushKey function from the Windows CE API. The implementation of this function is OEM dependent therefore it is not guaranteed to work with all the Windows Embedded devices.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>RegSaveCE() // Returned value = 0 if successful</td>
</tr>
</tbody>
</table>
**ReloadAppFromServer**

The function `ReloadAppFromServer` reloads the necessary project files from the server while maintaining the current state of the project on the client.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReloadAppFromServer</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
ReloadAppFromServer()
```

This function takes no parameters.

**Returned value**

This function always returns 0.

**Notes**

For this function, the server is the computer or device that hosts the downloadable project files (e.g., screens) for your thin clients. It might be different from the project runtime server (a.k.a. the data server), depending on how you deploy your project. For more information, see Configuring a Web Solution.

If the option **Enforce Web functionality equivalence** is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

**Tip:** Before you call this function, you can use the function `IsAppChangedOnServer` to check the project files that are already on the client. If they match the files on the server, you might choose not to call this function.

**SaveAlarmFile**

Use this function to enable/disable the saving feature for alarm history and to set the path where the alarm history files must be handled.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaveAlarmFile</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
SaveAlarmFile( numType, optRemotePath )
```

**numType**

Tag containing the number and operation, as follows:

- 0  Disable save the alarm file to the local disk
- 1  Enable save the alarm file to local disk
- 2  Enable save the alarm file to local disk and to the remote path specified in the `optRemotePath` parameter

**optRemotePath**
Tag containing the name of the remote computer where the alarm file will be saved simultaneously to the local computer and to the remote path when `numType` equals 2.

### Returned value

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>2nd parameter is not a string</td>
</tr>
<tr>
<td>2</td>
<td>2nd parameter is missing</td>
</tr>
</tbody>
</table>

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>SaveAlarmFile( 0 )</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>SaveAlarmFile( 1 )</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>SaveAlarmFile( 2, &quot;Z:\Apps\AppDemo&quot; )</code></td>
</tr>
</tbody>
</table>

### SetAppAlarmPath

Sets the *Alarm* path for the current project.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetAppAlarmPath</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

#### Syntax

`SetAppAlarmPath(strPath)`

*strPath*  
The new *Alarm* path for the current project.

#### Returned value

This function does not return any value.

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>SetAppAlarmPath( &quot;C:\Studio\Alarm\&quot; )</code></td>
</tr>
</tbody>
</table>

### SetAppHstPath

Sets the file path (directory) where Trend history files will be saved, in the proprietary format (*.HST*).

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetAppHstPath</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

#### Syntax

`SetAppHstPath(strPath)`

*strPath*  
The file path (directory) where Trend history files will be saved.

#### Returned value

This function does not return any value.
Notes
This function is useful when you intend to change the file path during runtime. You can also set the file path to a network drive by mapping it on the local station, or by using the following syntax:

\Network Drive\File Path

Please note that this function does not copy existing history files from the default directory to a new one; it only sets the file path for new history files saved after the function is called.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SetAppHstPath(&quot;C:\Studio\History&quot;)</td>
</tr>
</tbody>
</table>

_setDateFormat_

Sets the format and separator for the date string.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetDateFormat</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

SetDateFormat( strSeparator, strFormat )

strSeparator
The separator character for the date string.

strFormat
String tag, which specifies the order of the Month (M), Day (D), and Year (Y) in the date string.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DMY</td>
<td>Day, Month, Year</td>
</tr>
<tr>
<td>MDY</td>
<td>Month, Day, Year</td>
</tr>
<tr>
<td>YMD</td>
<td>Year, Month, Day</td>
</tr>
</tbody>
</table>

Returned value

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>Invalid parameter</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SetDateFormat( &quot;/&quot;, &quot;MDY&quot; ) // Date = 04/18/2002</td>
</tr>
<tr>
<td></td>
<td>SetDateFormat( &quot;:&quot;, &quot;MYD&quot; ) // Date = 04:2002:18</td>
</tr>
</tbody>
</table>

_setKeyboardLanguage_

Sets the language of any Virtual Keyboards in the project.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetKeyboardLanguage</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

SetKeyboardLanguage( strLanguage )

SetKeyboardLanguage( strLanguage )

strLanguage

String tag with the language code used for the Virtual Keyboards. The currently available options include:

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>English (default)</td>
</tr>
<tr>
<td>GE</td>
<td>German</td>
</tr>
<tr>
<td>FR</td>
<td>French</td>
</tr>
<tr>
<td>CH</td>
<td>Chinese</td>
</tr>
<tr>
<td>JA</td>
<td>Japanese</td>
</tr>
</tbody>
</table>

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Error</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SetKeyboardLanguage( &quot;EN&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>SetKeyboardLanguage( tagLanguage )</td>
</tr>
</tbody>
</table>

SetRegValue

Sets the value of a variable in the Windows registry.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetRegValue</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

SetRegValue( numMainKey, strKey, strVariableName, numType, strOrNumValue )

numMainKey

Numeric tag with the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Value Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HKEY_LOCAL_MACHINE</td>
</tr>
<tr>
<td>1</td>
<td>HKEY_CLASSES_ROOT</td>
</tr>
<tr>
<td>2</td>
<td>HKEY_CURRENT_USER</td>
</tr>
<tr>
<td>3</td>
<td>HKEY_USERS</td>
</tr>
<tr>
<td>4</td>
<td>HKEY_CURRENT_CONFIG</td>
</tr>
<tr>
<td>5</td>
<td>HKEY_PERFORMANCE_DATA</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

**strKey**
Path where the value is located in the Main Key.

**strVariableName**
Name of the variable to be set. The maximum length is 255 characters.

**numType**
Two types are currently supported:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DWord</td>
</tr>
<tr>
<td>1</td>
<td>String</td>
</tr>
</tbody>
</table>

**strOrNumValue**
Variable value.

**Returned value**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters or invalid Main Key.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid type.</td>
</tr>
<tr>
<td>-3</td>
<td>Failed to read the variable value; verify that you have the proper security rights.</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SetRegValue( 0, &quot;HARDWARE\DEVICEMAP\SERIALCOMM&quot;, &quot;\Device\Serial1&quot;, 1, &quot;COM3&quot; ) // Returned value = 0 if successful</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>SetRegValue( 2, &quot;Control Panel\Desktop&quot;, &quot;Smooth Scroll&quot;, 0, 1 ) // Returned value = 0 if successful</td>
<td></td>
</tr>
</tbody>
</table>

⚠️ **Caution:** This register can affect the Windows system configuration. You should be extremely careful and edit the registry only when you are certain about the configuration.

**SetWebConfig**
Configures the Web settings for the current project. The settings configured in the function are updated on the HTML files of the project.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetWebConfig</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Syntax**

```
SetWebConfig( strServerIP, optStrBackupURL, optStrPathFile, optNumHostPort, optStrSecondaryServerIP, optStrISSymbolURL )
```

**strServerIP**
Data Server IP Address. IP address (or hostname) of the computer where the TCP Server module of IWS is running.

**optStrBackupURL**
Optional alternative URL for the project's Web pages. The Thin Client will look for the Web pages in this URL if it does not find them in the same URL written in the Address field of the Web browser.

**optStrPathFile**

Optional file path and name of the HTML file to be updated. If you specify only the file path without a file name, then all of the HTML files in the specified file path will be updated.

**Note:** You can specify an individual file (e.g., you only want to update one file). This is especially useful for projects running on Windows Embedded target systems.

**optNumHostPort**

Optional TCP Port number that the Thin Client must use to exchange data with the TCP Server module of IWS.

**optStrSecondaryServerIP**

Optional alternative data server IP address. The Thin Client will attempt to connect to the TCP Server module of IWS in this IP Address if it is not able to connect to the TCP Server module running in the IP Address specified in the strServerIP parameter.

**optNumProtocolFlag**

Optional When you use the Web Tunneling Gateway option, this parameter specifies whether the Thin Client will use either HTTP to exchange data with the Web Server or HTTPS (SSL – Secure Socket Layer). If this flag has the value 0, the Thin Client will use HTTP. If this flag has the value 1, the Thin Client will use HTTPS (SSL).

**optNumGtwPort**

Optional TCP Port number that the Thin Client must use to exchange data with the Web Server when using the Web Tunneling Gateway.

**optStrGtwIP**

Optional IP Address (or hostname) of the computer where the Web Tunneling Gateway is running.

**optStrSecondaryGtwIP**

Optional Alternative IP Address (or hostname) of the computer where the Web Tunneling Gateway is running. The Thin Client will attempt to connect to the Web Tunneling Gateway in this IP Address if it is not able to connect to the Web Tunneling Gateway running in the IP Address specified in the optStrGtwIP parameter.

**optStrISSymbolURL**

Optional URL from where the updated version of ISSymbol (ActiveX control) must be downloaded if it is not properly registered in the Thin Client station.

**Note:**

- You can use tags or expressions as arguments of this function. Therefore, you can use this function to configure the web settings automatically during runtime, according to the network settings of each project (IP address, Web Server URL, and so forth).
- Only the first parameter of this function (strServerIP) is mandatory. All other parameters are optional. The parameters that are not configured in the function assume the default value configured in the Web tab of Project Settings.
- The following parameters should be omitted unless you intend to use the Web Tunneling Gateway: optNumProtocolFlag, optNumGtwPort, optStrGtwIP, optStrSecondaryGtwIP, and optStrISSymbolURL.
Returned value

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>2</td>
<td>Invalid Server IP address</td>
</tr>
<tr>
<td>3</td>
<td>Invalid URL</td>
</tr>
<tr>
<td>4</td>
<td>Invalid optional path</td>
</tr>
<tr>
<td>5</td>
<td>No Web pages found</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SetWebConfig( &quot;192.168.1.28&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>SetWebConfig( GetComputerIP(), &quot;http://&quot; + GetComputerIP() + &quot;/&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>SetWebConfig( &quot;192.168.1.28&quot;, &quot;<a href="http://200.0.0.10/">http://200.0.0.10/</a>&quot;, &quot;C:\MyWebPages &quot;, 1234, &quot;192.168.1.29&quot;, 0, 80, &quot;200.0.0.1&quot;, &quot;200.0.0.10&quot;, &quot;<a href="http://200.0.0.10/MyISSymbol/">http://200.0.0.10/MyISSymbol/</a>&quot; )</td>
</tr>
</tbody>
</table>

SNMPGet

Gets information from computers or network devices through the SNMP protocol.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPGet</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

`SNMPGet( strAddress, strCommunity, strOID, "strTagName" )`

**strAddress**

The address of the machine/computer (e.g., "127.0.0.1" or "localhost").

**strCommunity**

SNMP community name when communicating with the computer (e.g., "public").

**strOID**

OID to be consulted (e.g., ".1.3.6.1.2.1.1.1.0").

**strTagName**

Name of the tag that will receive the requested value.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.
## Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>-5</td>
<td>GET operation failed</td>
</tr>
<tr>
<td>-7</td>
<td>Invalid tag name</td>
</tr>
<tr>
<td>-8</td>
<td>Invalid tag type</td>
</tr>
<tr>
<td>-9</td>
<td>This function is not supported on the current operating system</td>
</tr>
</tbody>
</table>

If you receive any other values, please contact technical support.

## Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorTag</td>
<td>SNMPGet( &quot;127.0.0.1&quot;, &quot;public&quot;, &quot;.1.3.6.1.2.1.1.1.0&quot;, &quot;SysDescrTag&quot; )</td>
</tr>
</tbody>
</table>

//ErrorTag will receive the error code. If the function succeeds, the value in the OID ".1.3.6.1.2.1.1.1.0" will be saved in the tag SysDescrTag.

## SNMPSet

Uses the Simple Network Management Protocol (SNMP) to set a value on a target computer of network device.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMPSet</td>
<td>System Info</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

### Syntax

```plaintext
SNMPSet( strAddress, strCommunity, strOID, Value, optNumType )
```

**strAddress**

The address of the target computer or device (e.g., "127.0.0.1" or "localhost").

**strCommunity**

The SNMP community name (e.g., "public") when communicating with the target computer or device.

**strOID**

The Object ID (OID) to be set.

**Value**

The value to be set to the specified OID.

**optNumType**

A numeric value, or a tag of Integer type, specifying the data type of Value. This is an optional parameter, but if it is included, then it must have one of following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OCTETSTRING</td>
<td>An octet string variable</td>
</tr>
<tr>
<td>1</td>
<td>INTEGER32</td>
<td>A 32-bit signed integer variable</td>
</tr>
<tr>
<td>2</td>
<td>TIMETICKS</td>
<td>A timeticks variable</td>
</tr>
</tbody>
</table>
### Value Table

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>GAUGE32</td>
<td>A gauge variable</td>
</tr>
<tr>
<td>4</td>
<td>COUNTER32</td>
<td>A counter variable</td>
</tr>
<tr>
<td>5</td>
<td>IPADDRESS</td>
<td>An IP address variable</td>
</tr>
<tr>
<td>6</td>
<td>OBJECTIDENTIFIER</td>
<td>An object identifier variable</td>
</tr>
<tr>
<td>7</td>
<td>SEQUENCE</td>
<td>An ASN sequence variable</td>
</tr>
<tr>
<td>8</td>
<td>OPAQUE</td>
<td>An opaque variable</td>
</tr>
</tbody>
</table>

### Returned value Table

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>−1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>−2</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>−5</td>
<td>SET operation failed</td>
</tr>
<tr>
<td>−7</td>
<td>Invalid tag name</td>
</tr>
<tr>
<td>−8</td>
<td>Invalid tag type</td>
</tr>
<tr>
<td>−9</td>
<td>This function is not supported on the current operating system</td>
</tr>
</tbody>
</table>

If you receive any other values, please contact technical support.

### Examples

#### WritePrivateProfileString

Writes a specified setting to the project viewer initialization file, using the standard `.ini` format.

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>SNMPSet( &quot;127.0.0.1&quot;, &quot;public&quot;, &quot;.1.3.6.1.2.1.1.1.0&quot;, 123, 1 )</code> //Sets an integer value of 123 to the specified OID on the localhost (127.0.0.1).</td>
</tr>
</tbody>
</table>

#### Syntax

```
WritePrivateProfileString ( strSection, strName, strValue, strFileName )
```

**strSection**

The section name to be written.

**strName**

The parameter name to be written.

**strValue**

The value to be written.

**strFileName**

The path and name of the `.ini` file to be written.
**Returned value**

The function returns 1 if the file was updated successfully.

**Notes**

When running on Windows Embedded this function will rewrite the entire file, therefore its use is not recommended for lengthy files on Windows Embedded devices. The function will also add the following lines at the end of the file when on a Windows Embedded device:

```
[FileBackUpControl]
Valid=1
```

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>WritePrivateProfileString( Section, Name, Value, FileName )</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>WritePrivateProfileString( &quot;Options&quot;, &quot;ds1&quot;, &quot;Value&quot;, &quot;C:\Viewer.ini&quot; )</code></td>
</tr>
</tbody>
</table>
Tags Database functions

These functions are used to directly change the values of project tags.

**ExecuteAlarmAck**

This function acknowledges an active alarm on the specified tag. The advantage of using this function is that if used from the Thin Client, the Alarm task will store the user name and station from which the alarm was acknowledged.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecuteAlarmAck</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

```c
ExecuteAlarmAck( "strTagName", optStrComment, optStrAlarmType )
```

**strTagName**

Name of the tag on which the alarm will be acknowledged.

- **Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**optStrComment**

An optional comment to send to the Alarm task, along with the user name and station.

**optStrAlarmType**

If more than one alarm is active on the specified tag, you can specify which alarm (e.g., Hi, Lo, HiHi, LoLo) to acknowledge. Otherwise, the function acknowledges the most recently activated alarm.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successfully executed.</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid tag name.</td>
</tr>
<tr>
<td>-3</td>
<td>Executed, but did not wait for confirmation from Alarms task. See note.</td>
</tr>
</tbody>
</table>

**Notes**

When this function is used to acknowledge an alarm, it typically waits for confirmation from the Alarms task before returning a value of 0 to indicate successful execution. In some cases, however, waiting for confirmation might cause the project runtime to hang. When that happens, if the function is properly formed with valid parameters, then it will execute as intended but it will not wait for confirmation.

**Examples**

Acknowledge the active Hi alarm on tag A, with the comment Hi alarm on tag A:

```c
ExecuteAlarmAck( "A", "Hi alarm on tag A", "Hi" )
```
**ForceTagChange**

Forces the database to write a value to a tag and trigger events as if the tag changed, even if the new value is equal to the old value.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceTagChange</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
ForceTagChange( "strTagName", numValue )
```

**strTagName**

The name of the tag being forced to accept the new value.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**numValue**

The new value to be written to the specified tag.

**Returned value**

This function does not return any value.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>ForceTagChange( &quot;TagA&quot;, 5 )</td>
</tr>
</tbody>
</table>

**GetAlarmCount**

This function gets the number of active alarms.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAlarmCount</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
GetAlarmCount()
```

**GetAlarmCount()**

This function has no parameters.

**Returned value**

This function returns the number of active alarms.

Please note that the function will return a valid value only when it is executed on the project server. If it is called by a Graphics Script, Screen Script, or Command animation on a project client, it will return -1.
**GetAlarmInfo**

This function gets information about an active alarm.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAlarmInfo</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported (see &quot;Notes&quot; below)</td>
<td>Not supported (see &quot;Notes&quot; below)</td>
</tr>
</tbody>
</table>

**Syntax**

GetAlarmInfo(*numIndex*, *numInfo*)

GetAlarmInfo(*numIndex*, { *numInfo* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 })

**numIndex**

The index number of the alarm about which you want to get information.

The index number must be between 0 and the total number of alarms minus 1. To get the total number of alarms, use the function GetAlarmCount. For example, if GetAlarmCount returns 6, then numIndex must be between 0 and 5.

The list of alarms is sorted by the times when the alarms first became active, so that the oldest alarm is index 0. When an alarm is both normalized and acknowledged, it is removed from the list and the remaining alarms are shifted accordingly. For example, when the alarm at index 3 is removed, the alarm at index 4 is shifted to index 3, the alarm at index 5 is shifted to index 4, and so on.

**Note:** It is possible for an alarm to be normalized (i.e., become inactive) and then become active again without being acknowledged. That alarm will remain on the list of alarms throughout, and its Activation Time — that is, time when the alarm first became active — will not be updated.

**numInfo**

The information or alarm property that you want to get:

<table>
<thead>
<tr>
<th>Value</th>
<th>Alarm Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm Group</td>
</tr>
<tr>
<td>1</td>
<td>Tag Name</td>
</tr>
<tr>
<td>2</td>
<td>Alarm Message</td>
</tr>
<tr>
<td>3</td>
<td>Alarm Type</td>
</tr>
<tr>
<td>4</td>
<td>Tag value when the alarm became active</td>
</tr>
<tr>
<td>5</td>
<td>Activation Time</td>
</tr>
<tr>
<td>6</td>
<td>Norm Time</td>
</tr>
<tr>
<td>7</td>
<td>Ack Time</td>
</tr>
<tr>
<td>8</td>
<td>Alarm Priority</td>
</tr>
<tr>
<td>9</td>
<td>Alarm Selection</td>
</tr>
<tr>
<td>10 to 19</td>
<td>Custom Field 1 to Custom Field 10</td>
</tr>
</tbody>
</table>

**Tip:** You can create custom fields in the Alarms worksheet.
Returned value

This function returns the current value of the specified property (numInfo) of the specified alarm (numIndex). For the property Alarm Type (i.e., numInfo is 3), this function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Alarm Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HiHi</td>
</tr>
<tr>
<td>2</td>
<td>Hi</td>
</tr>
<tr>
<td>4</td>
<td>Lo</td>
</tr>
<tr>
<td>8</td>
<td>LoLo</td>
</tr>
<tr>
<td>16</td>
<td>Rate of change</td>
</tr>
<tr>
<td>32</td>
<td>Deviation+</td>
</tr>
<tr>
<td>64</td>
<td>Deviation-</td>
</tr>
</tbody>
</table>

For all other properties, this function returns the actual value of the property. As such, the project tag or VBScript variable that you configure to receive the returned value should be of the appropriate data type. For example, you may configure an Integer tag to receive the value of Alarm Group or Alarm Type, but you should configure a String tag to receive the value of Tag Name or Ack Time. It is safest to always configure a String tag to receive the value, because a String tag can hold any value as a string, but that might make it more difficult to process the value after it is received. Please keep this in mind as you design and develop your project.

If the specified alarm has not yet been normalized or acknowledged, this function returns no values at all for the properties Norm Time and Ack Time, respectively.

If the function is not executed successfully, it returns one of the following error codes:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Function was called by a Graphics Script, Screen Script, or Command animation on a project client. (See &quot;Notes&quot; below.)</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid alarm index (numIndex).</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid alarm property (numInfo).</td>
</tr>
</tbody>
</table>

Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a thin client. For more information, see Edit the tags database during run time on page 171.

Also, this function is similar to the function TagsDBGetAlarm, except that it gets information about a specified alarm in the list of active alarms rather than about an alarm on a specified tag.

Examples

Get the type (e.g., 2, meaning Hi) of the first alarm:

GetAlarmInfo(0,3)

Get the name of the project tag (e.g., MyTag) that the second alarm is on:

GetAlarmInfo(1,1)

Get the date and time (e.g., 12/05/2013 14:11:29) when the newest alarm became active, and then use the function DateTime2UTC to convert that from the current time zone to Coordinated Universal Time (UTC):

DateTime2UTC(GetAlarmInfo(GetAlarmCount()-1,5))
**GetTagValue**

Gets the value of the specified tag from the project tags database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetTagValue</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

GetTagValue( "strTagName", optNumRefresh )

**strTagName**

The name of the tag of which you want to get the value.

- **Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**optNumRefresh**

Tag that you want to use as a trigger to refresh the function. When the value of the specified Tag changes, the function is executed again. (Normally, a function executes only when the object on which it is configured changes in some way, such as when a Pushbutton object is clicked.) To execute the function at a regular interval, you can use one of project’s system tags such as Second, Minute or Hour.

**Returned value**

This function only returns the value of the Tag specified by strTagName. If the specified Tag does not exist, then the function returns null.

- **Note:** The value of the Tag specified by optNumRefresh does not affect the function’s returned value in any way.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>requiredTag = 15</td>
<td>GetTagName( &quot;requiredTag&quot;, Second ) // Return = 15</td>
</tr>
<tr>
<td>pointerTag = &quot;Required Tag&quot; requiredTag = 15 optNum = Second</td>
<td>GetTagName( pointerTag, optNum ) // Return = 15</td>
</tr>
</tbody>
</table>

**SetTagValue**

Sets the value of the specified tag in the project tags database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetTagValue</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Syntax**

SetTagValue( "strTagName", TagValue )

**strTagName**
The name of the tag that you want to set.

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

TagValue

The new value to be set to the specified tag.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Invalid tag name</td>
</tr>
<tr>
<td>0</td>
<td>No error</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagA</td>
<td>SetTagValue( &quot;TagA&quot;, &quot;Hello&quot; )</td>
<td>&quot;Hello&quot;</td>
</tr>
<tr>
<td>TagA</td>
<td>SetTagValue( &quot;TagA&quot;, 123 )</td>
<td>123</td>
</tr>
<tr>
<td>TagA TagB = 15</td>
<td>SetTagValue( &quot;TagA&quot;, TagB )</td>
<td>15</td>
</tr>
</tbody>
</table>

TagsDBAddClass

This function adds a new class to the tags database during run time.

Syntax

TagsDBAddClass( strClassName )

strClassName

The name of the class to be added.

Returned value

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>A class with the specified name already exists in the tags database.</td>
</tr>
<tr>
<td>-4</td>
<td>Maximum number of classes reached.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function TagsDBBeginEdit was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>
Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function **TagsDBBeginEdit** must have been executed previously, in order to lock the tags database for editing; and
- The function **TagsDBEndEdit** must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see **Edit the tags database during run time** on page 171.

**TagsDBAddClassMember**

This function adds a new class member to an existing class during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBAddClassMember</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
TagsDBAddClassMember( strClassName, strMemberName, strMemberType, strDescription )
```

```
TagsDBAddClassMember( strClassName, strMemberName, { strMemberType | "Boolean" | "Integer" | "Real" | "String" }, strDescription )
```

**strClassName**

The name of the class to which the member will be added.

**strMemberName**

The name of the class member to be added.

**strMemberType**

The data type of the class member to be added. This parameter accepts only the following values: "Boolean", "Integer", "Real", or "String".

**strDescription**

A description of the class member to be added.

**Returned value**

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>The specified class does not exist in the tags database.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function <strong>TagsDBBeginEdit</strong> was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

**Notes**

The following restrictions apply to the execution of this function:
• The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
• The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
• The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

**TagsDBAddTag**

This function adds a new project tag to the tags database during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBAddTag</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
TagsDBAddTag(strTagName, optStrTagType, optNumArraySize, optStrDescription, optNumScope)
```

**strTagName**
The name of the project tag to be added.

**optStrTagType**
The data type of the project tag to be added. This parameter accepts only the following values: "Boolean", "Integer", "Real", "String", or "classname".

This parameter is optional; if you do not specify a tag or value, then the default value is "Integer".

**optNumArraySize**
The array size of the project tag to be added.

This parameter is optional; if you do not specify a tag or value, then the default value is 0.

**optStrDescription**
A description of the project tag to be added.

This parameter is optional; if you do not specify a tag or value, then the default value is "".

**optNumScope**
The scope of the project tag to be added. This parameter accepts only the following values: 0 (Local) or 1 (Server).

This parameter is optional; if you do not specify a tag or value, then the default value is 1.

**Returned value**

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>The maximum number of tags (as determined by the project's target system) has been reached.</td>
</tr>
<tr>
<td>-4</td>
<td>A project tag with the specified name already exists in the tags database.</td>
</tr>
</tbody>
</table>
### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function <code>TagsDBBeginEdit</code> was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function `TagsDBBeginEdit` must have been executed previously, in order to lock the tags database for editing; and
- The function `TagsDBEndEdit` must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

### TagsDBBeginEdit

This function locks the tags database for editing during run time. You must call this function before you call any other Tags Database functions to add or remove project tags or configure tag properties.

#### Syntax

```
TagsDBBeginEdit()
```

This function has no parameters.

#### Returned value

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
<tr>
<td>-1</td>
<td>Project runtime is busy; another function is editing the tags database. Try again later.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-3</td>
<td>Project runtime is busy; an Alarm or Trend task is using the tags database. Try again later.</td>
</tr>
<tr>
<td>-4</td>
<td>The project has Collaboration enabled. To edit the tags database, you must check the tags database files out of source control. For more information, see About collaboration and source control on page 452.</td>
</tr>
</tbody>
</table>

#### Notes

The following restrictions apply to the execution of this function:
• The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client; and

• The function `TagsDBEndEdit` must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

Please note this function has a persistent effect, which means that if you call the function to lock the tags database during run time and then stop the project, then the database will remain locked and you will not be able to manually edit it.

Restarting the project may or may not unlock the database, depending on how you designed your project and which function call locked the database in the first place. As such, you should use the Database Spy to manually call the function `TagsDBEndEdit`. When that is executed successfully, you can safely restart your project.

**TagsDBEndEdit**

This function finishes changes made by other Tags Database functions and releases the database so that normal run-time execution may resume.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBEndEdit</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`TagsDBEndEdit()`

This function has no parameters.

**Returned value**

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function <code>TagsDBBeginEdit</code> was not executed successfully before this function was called. No changes were made.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

**Notes**

The following restrictions apply to the execution of this function:

• The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client; and

• The function `TagsDBBeginEdit` must have been executed previously, in order to lock the tags database for editing.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.
**TagsDBGetAlarm**

This function gets the properties of an alarm condition during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetAlarm</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
TagsDBGetAlarm( strTagName, numAlarmType, numProperty )
```

TagsDBGetAlarm( strTagName, { numAlarmType | 1 | 2 | 4 | 8 | 16 | 32 | 64 }, { numProperty | 0 | 1 | 2 | 3 | 4 | 5 })

**strTagName**

The name of the project tag on which the alarm condition is set.

**numAlarmType**

The alarm condition, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Alarm Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HiHi</td>
</tr>
<tr>
<td>2</td>
<td>Hi</td>
</tr>
<tr>
<td>4</td>
<td>Lo</td>
</tr>
<tr>
<td>8</td>
<td>LoLo</td>
</tr>
<tr>
<td>16</td>
<td>Rate</td>
</tr>
<tr>
<td>32</td>
<td>DeviationP</td>
</tr>
<tr>
<td>64</td>
<td>DeviationM</td>
</tr>
</tbody>
</table>

**Tip:** To get tag properties on project tags, use the function `TagsDBGetTagProperty`.

**numProperty**

The specific property of the alarm condition, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Property</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Limit value</td>
<td>Real</td>
</tr>
<tr>
<td>1</td>
<td>Message</td>
<td>String</td>
</tr>
<tr>
<td>2</td>
<td>Alarm group (or worksheet) number</td>
<td>Integer</td>
</tr>
<tr>
<td>3</td>
<td>Priority</td>
<td>Integer</td>
</tr>
<tr>
<td>4</td>
<td>Selection</td>
<td>String</td>
</tr>
<tr>
<td>5 to 14</td>
<td>Custom field</td>
<td>String</td>
</tr>
</tbody>
</table>

**Returned value**

If this function is executed successfully, then it returns the value of the specified property. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Project tag or alarm type does not exist.</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

### TagsDBGetClassMember

This function gets the data type or description of a specified class member during run time.

#### Syntax

```plaintext
TagsDBGetClassMember( strClassName, strMemberName, numPropertyID )
```

```plaintext
TagsDBGetClassMember( strClassName, strMemberName, { numPropertyID | 0 | 1 } )
```

- **strClassName**
  The name of the class that contains the member.

- **strMemberName**
  The name of the class member.

- **numPropertyID**
  The property to be gotten, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Property</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Data type</td>
<td>String (&quot;Boolean&quot;, &quot;Integer&quot;, &quot;Real&quot;, &quot;String&quot;)</td>
</tr>
<tr>
<td>1</td>
<td>Description</td>
<td>String</td>
</tr>
</tbody>
</table>

#### Returned value

If this function is executed successfully, then it returns the value of the specified property. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>The specified member does not exist in the specified class.</td>
</tr>
<tr>
<td>-4</td>
<td>The specified class does not exist in the tags database.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

### Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.
**TagsDBGetClassMemberCount**

This function gets a count of the members in a specified class.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetClassMemberCount</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`TagsDBGetClassMemberCount( strClassName )`

**TagsDBGetClassMemberCount (strClassName)**  
**strClassName**                   
The name of the class in which to count members.

**Returned value**

If this function is executed successfully, then it returns the requested count of class members. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>The specified class does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

**Notes**

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

**TagsDBGetFirstClass**

This function returns the name of the first class in the tags database. It acts like an array pointer, with the array being the tags database, and it may be used in coordination with the function TagsDBGetNextClass either to generate a list of classes in the tags database or to process the classes one at a time, depending on how you write your script.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetFirstClass</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`TagsDBGetFirstClass()`

**TagsDBGetFirstClass()**

This function has no parameters.
Returned value

If this function is executed successfully, then it returns the name of the first class. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>No classes found in the tags database.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function TagsDBBeginEdit does not need to have been executed previously. If that is the case, however, then TagsDBGetFirstClass and TagsDBGetNextClass can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

TagsDBGetFirstClassMember

This function returns the name of the first member in a specified class. It acts like an array pointer, with the array being the class members, and it may be used in coordination with the function TagsDBGetNextClassMember either to generate a list of members in the class or to process the class members one at a time, depending on how you write your script.

Function Group Execution Windows Embedded Thin Client Mobile Access
TagsDBGetFirstClassMember Tags Database Synchronous Supported (see "Notes" below) Not supported Not supported Not supported

Syntax

TagsDBGetFirstClassMember( strClassName )

strClassName

The name of the class in which to get the first member.

Returned value

If this function is executed successfully, then it returns the name of the first class member. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>No members found in the specified class.</td>
</tr>
<tr>
<td>-4</td>
<td>The specified class does not exist in the tags database.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function TagsDBBeginEdit does not need to have been executed previously. If that is the case, however, then TagsDBGetFirstClassMember and
TagsDBGetNextClassMember can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

**TagsDBGetFirstTag**

This function returns the name of the first project tag in the tags database. It acts like an array pointer, with the array being the tags database, and it may be used in coordination with the function TagsDBGetNextTag either to generate a list of tags in the tags database or to process the tags one at a time, depending on how you write your script.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetFirstTag</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`TagsDBGetFirstTag( optNumProjectOrSystem )`

`TagsDBGetFirstTag({ optNumProjectOrSystem | 0 | 1 })`

**optNumProjectOrSystem**

Determines whether to get the first tag of the user-defined Project Tags or the pre-defined System Tags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Project Tags</td>
</tr>
<tr>
<td>1</td>
<td>System Tags</td>
</tr>
</tbody>
</table>

This parameter is optional; if you do not specify a tag or value, then the default value is 1.

**Returned value**

If this function is executed successfully, then it returns the name of the first project tag. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>No project tags found in the tags database.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

**Notes**

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function TagsDBBeginEdit does not need to have been executed previously. If that is the case, however, then TagsDBGetFirstTag and TagsDBGetNextTag can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

**TagsDBGetNextClass**

This function returns the name of the next class in the tags database, after the function TagsDBGetFirstClass has been executed to get the first class. It acts like an array pointer, with the array
being the tags database, and it may be used to generate a list of classes in the tags database or to process the classes one at a time, depending on how you write your script.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetNextClass</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`TagsDBGetNextClass()`

This function has no parameters.

**Returned value**

If this function is executed successfully, then it returns the name of the next class. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>No class found; at end of database.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

**Notes**

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function `TagsDBBeginEdit` does not need to have been executed previously. If that is the case, however, then `TagsDBGetFirstClass` and `TagsDBGetNextClass` can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

**TagsDBGetNextClassMember**

This function returns the name of the next member in the specified class, after the function `TagsDBGetFirstClassMember` has been executed to get the first member. It acts like an array pointer, with the array being the members of the class, and it may be used to generate a list of classes members or to process the class members one at a time, depending on how you write your script.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetNextClassMember</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`TagsDBGetNextClassMember( strClassName )`

`TagsDBGetNextClassMember(strClassName)`

`strClassName`

The name of the class in which to get the next member.
Returned value

If this function is executed successfully, then it returns the name of the next class member. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>No members found; at end of class.</td>
</tr>
<tr>
<td>-4</td>
<td>The specified class does not exist in the tags database.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function TagsDBBeginEdit does not need to have been executed previously. If that is the case, however, then TagsDBGetFirstClassMember and TagsDBGetNextClassMember can only generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

TagsDBGetNextTag

This function returns the name of the next tag in the tags database, after the function TagsDBGetFirstTag has been executed to get the first tag. It acts like an array pointer, with the array being the tags database, and it may be used to generate a list of tags in the tags database or to process the tags one at a time, depending on how you write your script.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetNextTag()</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see “Notes” below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

TagsDBGetNextTag()

This function has no parameters.

Returned value

If this function is executed successfully, then it returns the name of the next tag. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>No tag found; at end of database.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

Notes

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

Also, this function may be called at any time; the function TagsDBBeginEdit does not need to have been executed previously. If that is the case, however, then TagsDBGetFirstTag and TagsDBGetNextTag can only
generate a list of classes. They cannot be used, along with with the other Tags Database functions, to edit the classes.

For more information about the Tags Database functions and examples of how to use them, see *Edit the tags database during run time* on page 171.

**TagsDBGetTagCount**

This function gets a count of the project tags in the tags database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetTagCount</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
TagsDBGetTagCount( numCountType )
```

TagsDBGetTagCount( { numCountType | 0 | 1 | 2 | 3 } )

**numCountType**

The type of count to be performed, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The total number of project tags in the tags database.</td>
</tr>
<tr>
<td>1</td>
<td>The number of project tags supported by the target system.</td>
</tr>
<tr>
<td>2</td>
<td>The number of project tags that may still be created before exceeding the number supported by the target system.</td>
</tr>
<tr>
<td>3</td>
<td>The total number of pre-defined system tags in the tags database.</td>
</tr>
</tbody>
</table>

**Returned value**

If this function is executed successfully, then it returns the specified tag count. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

**Notes**

This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see *Edit the tags database during run time* on page 171.

**TagsDBGetTagProperty**

This function gets the value of a specified tag property during run time. Tag properties are the metadata on project tags, such as tag name, array size, data type, description, scope, and so on. In some cases, calling this function is the same as using tag fields (e.g., `tagname->property`).

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetTagProperty</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

`TagsDBGetTagProperty( strTagName, numPropertyID )`

`TagsDBGetTagProperty(strTagName, numPropertyID | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 )`

**strTagName**
The name of the project tag from which the tag property will be gotten.

**numPropertyID**
The tag property to be gotten, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Property</th>
<th>Data Type</th>
<th>Same as...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tag name</td>
<td>String</td>
<td><code>tagname-&gt;Name</code></td>
</tr>
<tr>
<td>1</td>
<td>Array size</td>
<td>Integer</td>
<td><code>tagname-&gt;Size</code></td>
</tr>
<tr>
<td>2</td>
<td>Data type</td>
<td>String (&quot;Boolean&quot;, &quot;Integer&quot;, &quot;Real&quot;, &quot;String&quot;, &quot;classname&quot;)</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Description</td>
<td>String</td>
<td><code>tagname-&gt;Description</code></td>
</tr>
<tr>
<td>4</td>
<td>Scope</td>
<td>Integer (0 for Local, 1 for Server)</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Startup value</td>
<td>String</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Minimum value</td>
<td>Real</td>
<td><code>tagname-&gt;Min</code></td>
</tr>
<tr>
<td>7</td>
<td>Maximum value</td>
<td>Real</td>
<td><code>tagname-&gt;Max</code></td>
</tr>
<tr>
<td>8</td>
<td>Engineering unit</td>
<td>String</td>
<td><code>tagname-&gt;Unit</code></td>
</tr>
<tr>
<td>9</td>
<td>Retentive value</td>
<td>Integer (0 for disabled, 1 for enabled)</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>Retentive properties</td>
<td>Integer (0 for disabled, 1 for enabled)</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Dead band</td>
<td>Real</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>Smoothing</td>
<td>Integer (0 for disabled, 1 for enabled)</td>
<td>-</td>
</tr>
</tbody>
</table>

**Tip:** To get alarm conditions on project tags, use the function `TagsDBGetAlarm`. Other tag properties not listed here cannot be gotten during run time.

**numOrStrPropertyValue**
The value to set to the tag property. The value must be of the appropriate data type.

**Returned value**
If this function is executed successfully, then it returns the value of the specified tag property. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Project tag (as specified by strTagName) does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>
Notes
This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

TagsDBGetTrend
This function determines the trend group to which a project tag is assigned.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBGetTrend</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

TagsDBGetTrend( strTagName, numProperty )

TagsDBGetTrend(strTagName, {numProperty | 0 | 1})

strTagName
The name of the project tag that is assigned to a trend group.

numProperty
The specific property to be gotten, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Property</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Trend group (or worksheet) number</td>
<td>Integer</td>
</tr>
<tr>
<td>1</td>
<td>Log dead band</td>
<td>Real</td>
</tr>
</tbody>
</table>

Returned value
If this function is executed successfully, then it returns the value of the specified property. Otherwise, it returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Project tag (as specified by strTagName) does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
</tbody>
</table>

Notes
This function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

TagsDBRemoveAlarm
This function removes an alarm condition from a project tag during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBRemoveAlarm</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

TagsDBRemoveAlarm( strTagName, numAlarmType )

TagsDBRemoveAlarm(strTagName, { numAlarmType | 1 | 2 | 4 | 8 | 16 | 32 | 64 })

strTagName

The name of the project tag from which the alarm condition will be removed.

numAlarmType

The alarm condition to be removed, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Alarm Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HiHi</td>
</tr>
<tr>
<td>2</td>
<td>Hi</td>
</tr>
<tr>
<td>4</td>
<td>Lo</td>
</tr>
<tr>
<td>8</td>
<td>LoLo</td>
</tr>
<tr>
<td>16</td>
<td>Rate</td>
</tr>
<tr>
<td>32</td>
<td>DeviationP</td>
</tr>
<tr>
<td>64</td>
<td>DeviationM</td>
</tr>
</tbody>
</table>

Returned value

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Project tag (as specified by strTagName) does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function TagsDBBeginEdit was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

TagsDBRemoveClass

This function removes an existing class from the tags database during run time.
Appendix: Built-in Scripting Language

Syntax

```plaintext
TagsDBRemoveClass( strClassName )
```

TagsDBRemoveClass(strClassName)

strClassName

The name of the class to be removed.

Returned value

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>The specified class does not exist in the tags database.</td>
</tr>
<tr>
<td>-4</td>
<td>The specified class is being used (i.e., it has one or more tags associated with it).</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function TagsDBBeginEdit was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

**TagsDBRemoveClassMember**

This function removes an existing class member from a specified class during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBRemoveClassMember</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

```plaintext
TagsDBRemoveClassMember( strClassName, strMemberName )
```

TagsDBRemoveClassMember(strClassName, strMemberName)

strClassName

The name of the class that contains the member to be removed.

strMemberName

The name of the class member to be removed.
Returned value
This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>The specified member does not exist in the specified class.</td>
</tr>
<tr>
<td>-4</td>
<td>The specified class does not exist in the tags database.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function TagsDBBeginEdit was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

Notes
The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

TagsDBRemoveTag
This function removes an existing project tag from the tags database during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBRemoveTag</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see “Notes” below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

TagsDBRemoveTag( strTagName )

strTagName
The name of the project tag to be removed.

Returned value
This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>The specified tag does not exist in the tags database.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function TagsDBBeginEdit was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>
Notes
The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
- The function TagsDBEndEdit must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

Also, please note that executing this function does not completely remove the specified project tag from the tags database. That is not possible, due to how the database is maintained during run time. Instead, executing this function removes all uses of the specified tag in the project, so that they do not count against the tag limit on your runtime license, and then it reduces the tag to a single blank line in the database. Therefore, any time you execute this function to remove tags, you should stop the project as soon as it is practical to do so and then verify the project to remove the blank lines. For more information, see Verify the project.

TagsDBRemoveTrend
This function removes a project tag from its trend group during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBRemoveTrend</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

TagsDBRemoveTrend( strTagName )

TagsDBRemoveTrend(strTagName)
strTagName
  The name of the project tag to be removed.

Returned value
This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Project tag (as specified by strTagName) does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function TagsDBBeginEdit was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

Notes
The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function TagsDBBeginEdit must have been executed previously, in order to lock the tags database for editing; and
• The function `TagsDBEndEdit` must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

**TagsDBSetAlarm**

This function sets the properties of an alarm condition during run time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBSetAlarm</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
TagsDBSetAlarm( strTagName, numAlarmType, numProperty, strOrNumVal )
```

**strTagName**

The name of the project tag on which the alarm condition will be set.

**numAlarmType**

The type of alarm condition to be set, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Alarm Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HiHi</td>
</tr>
<tr>
<td>2</td>
<td>Hi</td>
</tr>
<tr>
<td>4</td>
<td>Lo</td>
</tr>
<tr>
<td>8</td>
<td>LoLo</td>
</tr>
<tr>
<td>16</td>
<td>Rate</td>
</tr>
<tr>
<td>32</td>
<td>DeviationP</td>
</tr>
<tr>
<td>64</td>
<td>DeviationM</td>
</tr>
</tbody>
</table>

**Tip:** To set tag properties on project tags, use the function `TagsDBSetTagProperty`.

**numProperty**

The specific property of the alarm condition to be set, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Property</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Limit value</td>
<td>Real</td>
</tr>
<tr>
<td>1</td>
<td>Message</td>
<td>String</td>
</tr>
<tr>
<td>2</td>
<td>Alarm group (or worksheet) number</td>
<td>Integer</td>
</tr>
<tr>
<td>3</td>
<td>Priority</td>
<td>Integer</td>
</tr>
<tr>
<td>4</td>
<td>Selection</td>
<td>String</td>
</tr>
<tr>
<td>5 to 14</td>
<td>Custom field</td>
<td>String</td>
</tr>
</tbody>
</table>

**strOrNumVal**
The value to set to the specified property. The value must be of the appropriate data type.

**Returned value**

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Project tag (as specified by strTagName) does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function <code>TagsDBBeginEdit</code> was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

**Notes**

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function `TagsDBBeginEdit` must have been executed previously, in order to lock the tags database for editing; and
- The function `TagsDBEndEdit` must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see [Edit the tags database during run time](#) on page 171.

**Note:** A specific alarm condition, as defined by the tag name and the alarm type (e.g., the HiHi alarm on *MyTag1*), can be in only one alarm group at a time. Therefore, when you assign an alarm condition to an alarm group, it will be removed automatically from its previous group, if any.

Also, make sure that an alarm group with the correct group/worksheet number actually exists before you try to assign an alarm condition to it.

### TagsDBSetTagProperty

This function sets the value of a specified tag property during run time. Tag properties are the metadata on project tags, such as tag name, array size, data type, description, scope, and so on.

**Syntax**

```
TagsDBSetTagProperty( strTagName, numPropertyID, numOrStrPropertyValue )

TagsDBSetTagProperty( strTagName, { numPropertyID | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 }, numOrStrPropertyValue )
```

- **strTagName**
  - The name of the project tag on which the tag property will be set.
- **numPropertyID**
  - The tag property to be set, identified by one of the following values:
### Value | Property | Data Type
---|---|---
0 | Tag name | String
1 | Array size | Integer
2 | Data type | String ("Boolean", "Integer", "Real", "String", "classname")
3 | Description | String
4 | Scope | Integer (0 for Local, 1 for Server)
5 | Startup value | String
6 | Minimum value | Real
7 | Maximum value | Real
8 | Engineering unit | String
9 | Retentive value | Integer (0 for disabled, 1 for enabled)
10 | Retentive properties | Integer (0 for disabled, 1 for enabled)
11 | Dead band | Real
12 | Smoothing | Integer (0 for disabled, 1 for enabled)

**Tip:** To set alarm conditions on project tags, use the function `TagsDBSetAlarm`. Other tag properties not listed here cannot be set during run time.

#### numOrStrPropertyValue

The value to set to the tag property. The value must be of the appropriate data type.

#### Returned value

This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Project tag (as specified by strTagName) does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function <code>TagsDBBeginEdit</code> was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

#### Notes

The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function `TagsDBBeginEdit` must have been executed previously, in order to lock the tags database for editing; and
- The function `TagsDBEndEdit` must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see *Edit the tags database during run time* on page 171.
**TagsDBSetTrend**

This function assigns a project tag to a trend group.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>TagsDBSetTrend</td>
<td>Tags Database</td>
<td>Synchronous</td>
<td>Supported (see &quot;Notes&quot; below)</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`TagsDBSetTrend( strTagName, numProperty, strOrNumVal )`

`TagsDBSetTrend(strTagName, { numProperty | 0 | 1 }, strOrNumVal)`

**strTagName**
The name of the project tag to assign to the trend group.

**numProperty**
The specific property to be set, identified by one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Property</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Trend group (or worksheet) number</td>
<td>Integer</td>
</tr>
<tr>
<td>1</td>
<td>Log dead band</td>
<td>Real</td>
</tr>
</tbody>
</table>

**strOrNumVal**
The value to set to the specified property. The value must be of the appropriate data type.

**Returned value**
This function returns one of the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Project tag (as specified by strTagName) does not exist.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong parameter type or inconsistent value.</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid number of parameters.</td>
</tr>
<tr>
<td>-1</td>
<td>The function <code>TagsDBBeginEdit</code> was not executed successfully before this function was called.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
</tbody>
</table>

**Notes**
The following restrictions apply to the execution of this function:

- The function can only be executed on the project server — it cannot be called by a Graphics Script, Screen Script, or Command animation running on a project client;
- The function `TagsDBBeginEdit` must have been executed previously, in order to lock the tags database for editing; and
- The function `TagsDBEndEdit` must be executed when the editing is finished, in order to release the tags database and resume normal run-time operations.

For more information about the Tags Database functions and examples of how to use them, see Edit the tags database during run time on page 171.

**Note:** A specific project tag can be in only one trend group at a time. Therefore, when you assign a project tag to a trend group, it will be removed automatically from its previous group, if any.
Also, make sure that a trend group with the correct group/worksheet number actually exists before you try to assign a project tag to it.

Finally, new trends will not start recording until either the system clock changes to a new day or you restart the project.
Loop functions

These functions are used to implement repeating, incrementing loop within a script.

**For...Next**

Implements a `For...Next` loop in a Math worksheet. The section of script included in the loop begins with the `For()` call and ends with the `Next` notation. The `Next` notation directs back to the beginning of the loop.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>For</td>
<td>Loop</td>
<td>N/A</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
For(numInitialValue, numFinalValue, numStep)
```

**numInitialValue**

The initial or starting value of the loop.

**numFinalValue**

The final or ending value of the loop.

**numStep**

The step or increment value.

**Returned value**

This function returns the current value of the loop.

**Notes**

You must partner every `For()` call with a `Next` notation, although you may have any number of worksheet rows between them. And as shown in the example below, you must place the `Next` notation in the Tag Name column of the worksheet.

The loop ends when its current value equals the value of numFinalValue. Specifically, each time the execution of the worksheet encounters the `Next` notation, the following happens:

1. The value of numStep is added to the current value of the loop;
2. The execution returns to the `For()` call at the beginning of the loop; and
3. The current value of the loop is compared to the value of numFinalValue. If the current value is less than numFinalValue, then the loop is executed again. If the current value is greater than or equal to numFinalValue, then the loop is skipped and execution resumes with the first row after the `Next` notation.

Please note that the `For...Next` loop may only be used in Math worksheets. It may not be used in either Command animations or any VBScript interface. (VBScript has its own `For...Next` statement.)

**Examples**

**Example of For...Next loop in a Math worksheet**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyTag</td>
<td>For(1,5,1)</td>
</tr>
<tr>
<td>tagname</td>
<td>function</td>
</tr>
<tr>
<td>tagname</td>
<td>function</td>
</tr>
<tr>
<td>tagname</td>
<td>function</td>
</tr>
<tr>
<td>Tag Name</td>
<td>Expression</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>Next</td>
<td></td>
</tr>
</tbody>
</table>
ODBC functions

These functions are used to interact with an external database via Open Database Connectivity (ODBC).

**Note:** These functions are provided to support legacy database interfaces. In most cases, we recommend that you use the newer Database/ERP connections manager and functions.

**ODBCBeginTrans**

Begins a transaction with the connected data source.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCBeginTrans</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
ODBCBeginTrans(numHandler)
```

**numHandler**
The handler returned by the **ODBCOpen** function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Error beginning transaction</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCBeginTrans( 5 )</td>
</tr>
</tbody>
</table>

**ODBCBindCol**

Binds a column to a tag.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCBindCol</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
ODBCBindCol( numHandler, strColName, strColType, strTagName )
```

**numHandler**
The handler returned by the **ODBCOpen** function.

**strColName**
The Database column name.

**strColType**
The SQL data type (one of the following):
- SQL_BIT
Appendix: Built-in Scripting Language

- SQL_TINYINT
- SQL_LONGVARCHAR
- SQL_CHAR
- SQL_VARCHAR
- SQL_DECIMAL
- SQL_NUMERIC
- SQL_DATE
- SQL_TIME
- SQL_TIMESTAMP
- SQL_DOUBLE
- SQL_REAL
- SQL_SMALLINT
- SQL_INTEGER

**strTagName**

The name of the tag to bind to the column.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid Handler</td>
</tr>
<tr>
<td>2</td>
<td>Invalid parameter type</td>
</tr>
<tr>
<td>3</td>
<td>One of the parameters has an empty string</td>
</tr>
<tr>
<td>4</td>
<td>ColType contains an invalid type</td>
</tr>
</tbody>
</table>

**Notes**

Every time you finish binding columns, you must call the **ODBCQuery** function.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCBindCol( 5, &quot;OrderDate&quot;, &quot;SQL_DATE&quot;, &quot;Order_Date&quot; )</td>
</tr>
</tbody>
</table>

See also: **ODBCUnbindCol()**

**ODBCCanAppend**

Returns whether the database will allow you to add new records.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCCanAppend</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

**ODBCCanAppend**(numHandler)

**numHandler**

The handler returned by the **ODBCOpen** function.
Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Database does not allow appending new records</td>
</tr>
<tr>
<td>Non-Zero</td>
<td>Database does allow appending new records</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCCanAppend( 5 )</td>
</tr>
</tbody>
</table>

**ODBCCanTransact**

Returns whether the database allows transactions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCCanTransact</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

```
ODBCCanTransact( numHandler )
```

**numHandler**

The handler returned by the `ODBCOpen` function.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Database does not allow transactions.</td>
</tr>
<tr>
<td>Non-Zero</td>
<td>Database does allow transactions.</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCCanTransact( 2 )</td>
</tr>
</tbody>
</table>

**ODBCCanUpdate**

Returns whether the database can be updated.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCCanUpdate</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

```
ODBCCanUpdate( numHandler )
```

**numHandler**

The handler returned by the `ODBCOpen` function.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Database does not allow updates.</td>
</tr>
<tr>
<td>Non-Zero</td>
<td>Database does allow updates.</td>
</tr>
</tbody>
</table>
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCCanUpdate( 6 )</td>
</tr>
</tbody>
</table>

**ODBCClose**

Closes a connection to the database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCClose</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ODBCClose(numHandler)`

`numHandler`  
The handler returned by the `ODBCOpen` function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid Handler</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCClose( 5 )</td>
</tr>
</tbody>
</table>

**ODBCCommitTrans**

Commits a transaction. Call this function upon completing transactions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCCommitTrans</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ODBCCommitTrans(numHandler)`

`numHandler`  
The handler returned by the `ODBCOpen` function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Error committing transaction</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCCommitTrans( 1 )</td>
</tr>
</tbody>
</table>
**ODBCDelete**

Deletes the current record.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCDelete</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

ODBCDelete(*numHandler*)

*numHandler*

The handler returned by the ODBCOpen function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Delete error</td>
</tr>
</tbody>
</table>

**Notes**

After a successful deletion, you must explicitly call one of the "move" functions (i.e., ODBCMove, ODBCMoveFirst, ODBCMoveLast, ODBCMoveNext, ODBCMovePrev) to move off the deleted record.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCDelete( 5 )</td>
</tr>
</tbody>
</table>

**ODBCExecuteSQL**

Directly executes an SQL statement.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCExecuteSQL</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

ODBCExecuteSQL(*numHandler, strSqlCommand*)

*numHandler*

The handler returned by the ODBCOpen function.

*strSqlCommand*

A valid SQL statement.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Invalid parameter</td>
</tr>
</tbody>
</table>
4 Error executing SQL command

**Note:** This function does not return any records, regardless of the statement that is executed.

## Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODCCtr</td>
<td>ODBCExecuteSQL( 3, &quot; )</td>
</tr>
<tr>
<td>ODCCst</td>
<td>ODBCExecuteSQL( 4, &quot; )</td>
</tr>
</tbody>
</table>

### ODCCInsert

Inserts a new record in the database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODCCInsert</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
ODCCInsert(numHandler)
```

*numHandler*

The handler returned by the ODBCOpen function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Insert error</td>
</tr>
</tbody>
</table>

**Notes**

This function uses the values of the tags bound by the ODBCBindCol function to create the new record.

## Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODCCtr</td>
<td>ODBCInsert( 7 )</td>
</tr>
</tbody>
</table>

### ODCCIsBOF

Returns whether you have gone above the first record of the record set. (Call this function before scrolling from record to record.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODCCIsBOF</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
ODCCIsBOF(numHandler)
```

*numHandler*

The handler returned by the ODBCOpen function.
## Returned value

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Record found</td>
</tr>
<tr>
<td>non-zero</td>
<td>Record set contains no records or you move backward, above the first record</td>
</tr>
</tbody>
</table>

You also can use this function along with the `ODBCIsEOF` function to determine whether the record set contains any records or is empty. Immediately after calling the `ODBCQuery` function, and if the record set contains no records, `ODBCIsBOF` returns non-zero. When you open a record set with at least one record, the first record is the current record and `ODBCIsBOF` returns a zero (0). If the first record is the current record, and you call `ODBCMovePrev`, `ODBCIsBOF` will subsequently return a non-zero.

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ODBCIsBOF(1)</code></td>
</tr>
</tbody>
</table>

### `ODBCIsDeleted`

Reports whether the current record was deleted.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCIsDeleted</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

### Syntax

`ODBCIsDeleted(numHandler)`

**numHandler**

The handler returned by the `ODBCOpen` function.

### Returned value

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Record set is not positioned on a deleted record.</td>
</tr>
<tr>
<td>non-zero</td>
<td>Record set is positioned on a deleted record.</td>
</tr>
</tbody>
</table>

If you move to a record and this function returns a non-zero, then you must move to another record before you can perform any other operations.

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ODBCIsDeleted(8)</code></td>
</tr>
</tbody>
</table>

### `ODBCIsEOF`

Reports whether you have gone beyond the last record of the record set. (Call this function as you scroll from record to record.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCIsEOF</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

### Syntax

`ODBCIsEOF(numHandler)`

**numHandler**

The handler returned by the `ODBCOpen` function.
Returned value

<table>
<thead>
<tr>
<th>Returned Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Record found.</td>
</tr>
<tr>
<td>non-zero</td>
<td>Record set contains no records or you moved forward, after the last record.</td>
</tr>
</tbody>
</table>

You can use this function along with the `ODBCIsBOF` function to determine whether the record set contains any records or is empty. Immediately after calling the `ODBCQuery` function, and if the record set contains no records, `ODBCIsBOF` returns non-zero. When you open a record set with at least one record, the first record is the current record and `ODBCIsEOF` returns a zero (0). If the last record is the current record, and you call `ODBCMoveNext`, `ODBCIsEOF` will subsequently return a non-zero.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ODBCIsEOF( 5 )</code></td>
</tr>
</tbody>
</table>

**ODBCIsFieldNULL**

Reports whether a specified field in a record set was flagged as `NULL`.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCIsFieldNULL</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ODBCIsFieldNULL(numHandler, strColName)`

- **numHandler**
  - The handler returned by the `ODBCOpen` function.
- **strColName**
  - The column name.

**Returned value**

<table>
<thead>
<tr>
<th>Returned Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The specified field is not flagged as Null.</td>
</tr>
<tr>
<td>Non-Zero</td>
<td>The specified field is flagged as Null.</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ODBCIsFieldNULL( 7, &quot;CustomerName&quot; )</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>ODBCIsFieldNULL( 3, &quot;CompanyName&quot; )</code></td>
</tr>
</tbody>
</table>

**ODBCIsFieldNullable**

Reports whether a specified field is nullable (i.e., can be set to a `NULL` value).

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCIsFieldNullable</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ODBCIsFieldNullable(numHandler, strColName)`

- **numHandler**
  - The handler returned by the `ODBCOpen` function.
**strColName**

The column name.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The specified field is not flagged as Nullable.</td>
</tr>
<tr>
<td>Non-Zero</td>
<td>The specified field is flagged as Nullable.</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCIsFieldNullable( 1, &quot;Price&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>ODBCIsFieldNullable( 1, &quot;Model&quot; )</td>
</tr>
</tbody>
</table>

**ODBCMove**

Moves the current record pointer within a record set, either forward or backward.

**Function**  | **Group**  | **Execution** | **Windows** | **Embedded** | **Thin Client** | **Mobile Access** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCMove</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Description**

Moves the current record pointer within a record set, either forward or backward.

**Syntax**

`ODBCMove( numHandler, numOffset )`

numHandler

The handler returned by the `ODBCOpen` function.

numOffset

The number of rows to move forward or backward:
- Positive values move forward, toward the end of the record set.
- Negative values move backward, toward the beginning of the record set.
- A value of 0 refreshes the current record.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Move error</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCMove( 2, 3 )</td>
</tr>
<tr>
<td>Tag</td>
<td>ODBCMove( 8, 2 )</td>
</tr>
</tbody>
</table>
**ODBCMoveFirst**

Moves to the first record within the record set.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCMoveFirst</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ODBCMoveFirst(numHandler)`

`numHandler` is the handler returned by the `ODBCOpen` function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Move error</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCMoveFirst(4)</td>
</tr>
</tbody>
</table>

**ODBCMoveLast**

Moves to the last record within the record set.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCMoveLast</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ODBCMoveLast(numHandler)`

`numHandler` is the handler returned by the `ODBCOpen` function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Move error</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCMoveLast(7)</td>
</tr>
</tbody>
</table>
**ODBCMoveNext**

Moves to the next record within the record set.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCMoveNext</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
ODBCMoveNext(numHandler)
```

`numHandler`  
The handler returned by the `ODBCOpen` function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>End of record set reached</td>
</tr>
<tr>
<td>4</td>
<td>Move error</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ODBCMoveNext(9)</code></td>
</tr>
</tbody>
</table>

**ODBCMovePrev**

Moves to the next record within the record set.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCMovePrev</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
ODBCMovePrev(numHandler)
```

`numHandler`  
The handler returned by the `ODBCOpen` function.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Beginning of record set reached</td>
</tr>
<tr>
<td>4</td>
<td>Move error</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ODBCMovePrev(2)</code></td>
</tr>
</tbody>
</table>
**ODBCOpen**

Opens a connection to the database and returns a numeric handler to be used by other ODBC functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCOpen</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
ODBCOpen( strDsn, strUser, strPassw, strTable, strFilter, strSort )
```

**strDsn**

The name of the data source.

**strUser**

The user name.

**strPassw**

The password.

**strTable**

The name of the database table

**strFilter**

The SQL *WHERE* clause.

**strSort**

The SQL *ORDER BY* clause.

**Returned value**

<table>
<thead>
<tr>
<th>Return</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>On success, returns the handler to identify the database</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>-2</td>
<td>DSN or TableName contain an empty string</td>
</tr>
</tbody>
</table>

**Notes**

This function does not read or write any data; it simply creates a handle to manipulate the database. You must bind the columns using the ODBCBindCol function, and then you must call the ODBCQuery function to retrieve the first record.

**Examples**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCOpen( &quot;MyDSNFile&quot;, &quot;Alex&quot;, &quot;,&quot;, &quot;Table1&quot;, &quot;Name='Mayer'&quot;, &quot;Name ASC&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>ODBCOpen( &quot;DSNFileName&quot;, &quot;Robert&quot;, &quot;Robot&quot;, &quot;Table1&quot;, &quot;,&quot;, &quot;,&quot; )</td>
</tr>
</tbody>
</table>

**ODBCQuery**

Retrieves the currently selected record from a database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCQuery</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Syntax

ODBCQuery(numHandler)

numHandler

The handler returned by the ODBCOpen function.

Returned value

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>No columns bound</td>
</tr>
<tr>
<td>3</td>
<td>Cannot open database</td>
</tr>
<tr>
<td>4</td>
<td>Cannot restart database</td>
</tr>
<tr>
<td>5</td>
<td>Query error</td>
</tr>
</tbody>
</table>

Notes
If you modify the column binding, or if you modify the filter and sort, then you must call this function again.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCQuery( 6 )</td>
</tr>
</tbody>
</table>

ODBCRollBack

Reverses the changes made during a transaction.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCRollBack</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

ODBCRollBack(numHandler)

numHandler

The handler returned by the ODBCOpen function.

Returned value

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database no open</td>
</tr>
<tr>
<td>3</td>
<td>Error rolling back transaciton</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCRollback( 4 )</td>
</tr>
</tbody>
</table>
**ODBCSetFieldNULL**

Flags a field data member in the record set as **NULL** (specifically having no value) or as **non-NULL**.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCSetFieldNULL</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```c
ODBCSetFieldNULL( numHandler, strColName, numValue )
```

**numHandler**
- The handler returned by the `ODBCOpen` function.

**strColName**
- The column name.

**numValue**
- A numeric tag that specifies the field data as NULL if 0 and non-NULL if non-zero.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>4</td>
<td>Invalid column name</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCSetFieldNULL( 2, &quot;Price&quot;, 1 )</td>
</tr>
<tr>
<td>Tag</td>
<td>ODBCSetFieldNULL( 4, &quot;CompanyName&quot;, 0 )</td>
</tr>
</tbody>
</table>

**ODBCSetFilter**

Constrains the records selected in a database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCSetFilter</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```c
ODBCSetFilter( numHandler, strFilter )
```

**numHandler**
- The handler returned by the `ODBCOpen` function.

**strFilter**
- The a SQL **WHERE** clause.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
</tbody>
</table>
You may find this function useful for selecting a subset of records, such as “all salespersons based in California” ("state = 'CA'"). Remember to call ODBCQuery after calling this function.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCSetFilter( 3, &quot;Name='Morgan'&quot; )</td>
</tr>
</tbody>
</table>

**ODBCSetSort**

Sorts the records selected in a database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCSetSort</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

ODBCSetSort(numHandler, strSort)

numHandler

The handler returned by the ODBCOpen function.

strSort

The SQL ORDER BY clause.

**Returned value**

<table>
<thead>
<tr>
<th>0</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Invalid parameter type</td>
</tr>
</tbody>
</table>

**Notes**

You can use this feature to sort the records in one or more columns. Remember to call ODBCQuery after calling this function.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>ODBCSetSort( 5, &quot;Name DESC&quot; )</td>
</tr>
</tbody>
</table>

**ODBCUnbindCol**

Unbinds a column that was previously bound using the ODBCBindCol function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCUnbindCol</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

ODBCUnbindCol(numHandler, strColName)

numHandler
The handler returned by the `ODBCOpen` function.

**strColName**

The column name.

**Returned value**

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Invalid parameter type</td>
</tr>
<tr>
<td>3</td>
<td>Column not bound</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ODBCUnbindCol( 7, &quot;Name&quot; )</code></td>
</tr>
</tbody>
</table>

**ODBCUpdate**

Updates the current record.

**Function**

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODBCUpdate</td>
<td>ODBC</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Not supported</td>
<td>Executed on Server</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`ODBCUpdate( numHandler)`

**numHandler**

The handler returned by the `ODBCOpen` function.

**Returned value**

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid handler</td>
</tr>
<tr>
<td>2</td>
<td>Database not open</td>
</tr>
<tr>
<td>3</td>
<td>Update error</td>
</tr>
</tbody>
</table>

**Notes**

This function uses the values of the tags bound by the `ODBCBindCol` function to update the current record.

**Examples**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>ODBCUpdate( 1 )</code></td>
</tr>
</tbody>
</table>
Email functions

These functions are used to configure and send email from within a project.

**CnfEmail**

This function configures the email settings used by other features in the project that can send email, such as Alarm worksheets and the SendEmail and SendEmailExt functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CnfEmail</td>
<td>Email</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported, except for encryption</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Syntax**

```plaintext
CnfEmail( strSMTP, strFrom, strPOP3, strUser, strPassword, optNumTimeout, optNumAuthType, optStrSMTPUser, optStrSMTPPassword )
```

strSMTP

The hostname or IP address of the outgoing email server, which is also known as the SMTP server. You can include a port number if the server does not use one of the standard SMTP ports.

Note: For projects that will run on Windows Embedded devices, you must specify an IP address.

strFrom

The email address from which emails will be sent and at which emails may be received. This should be a valid address on the POP3 server (see strPOP3 below).

strPOP3

The hostname or IP address of the incoming email server, which is also known as the POP3 server. You can include a port number if the server does not use one of the standard POP3 ports.

Note: For projects that will run on Windows Embedded devices, you must specify an IP address.

strUser

The username to be used to log onto the POP3 server.

strPassword

The password to be used to log onto the POP3 server.

optNumTimeout

The timeout limit (in seconds) to be used when sending email. If no response is received from the SMTP server within this period of time, then the operation is aborted.

This is an optional parameter; if no timeout is specified, then the project will keep trying forever until it receives a response. You should specify some timeout, however, to make sure that your project won’t freeze.

optNumAuthType

optStrSMTPUser
**optStrSMTPPassword**

By default, SMTP servers do not require authentication for outgoing email. If your server does require authentication, set optNumAuthType to 1 (unencrypted) or 2 (encrypted via TLS/SSL), and then specify the username and password. (If your SMTP username and password are the same as your POP3 username and password, then you can skip optStrSMTPUser and optStrSMTPPassword. The project will automatically use the values from strUser and strPassword."

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid format for strSMTP</td>
</tr>
<tr>
<td>2</td>
<td>Invalid format for strFrom</td>
</tr>
<tr>
<td>3</td>
<td>Invalid format for strPOP3</td>
</tr>
<tr>
<td>4</td>
<td>Invalid format for strUser</td>
</tr>
<tr>
<td>5</td>
<td>Invalid format for strPassword</td>
</tr>
<tr>
<td>6</td>
<td>Invalid format for optNumTimeout</td>
</tr>
<tr>
<td>7</td>
<td>Wrong number of parameters</td>
</tr>
<tr>
<td>8</td>
<td>Error getting host IP address (invalid POP3 server)</td>
</tr>
<tr>
<td>9</td>
<td>Error connecting to POP3 server</td>
</tr>
<tr>
<td>10</td>
<td>Error sending username</td>
</tr>
<tr>
<td>11</td>
<td>Error sending password</td>
</tr>
<tr>
<td>12</td>
<td>SMTP server does not support selected authentication mode</td>
</tr>
<tr>
<td>13</td>
<td>Invalid SMTP username</td>
</tr>
<tr>
<td>14</td>
<td>Authentication failed</td>
</tr>
</tbody>
</table>

**Notes**

The email configuration created by this function works only within the Windows process where the function was called.

For example, if you place a Button object in a screen and then set the object to call this function when it is pressed, the resulting email configuration will work only on the Client station where the screen is displayed and the button is pressed. It will not work on any other Client stations nor on the Server station, because the project viewer running on the Client station only exchanges data (i.e., changes in tag values) with the data server running on the Server station. One cannot directly call functions on the other; it can only use triggers to force the other to call functions. Please note that is true even when the Client station and the Server station are the same physical device, because the project viewer and the data server are two separate processes in Windows.

If you want an email configuration to apply to your project’s background tasks — for example, to be able to send emails when alarms become active — then you must either use the E-mail Settings dialog to configure default settings for the entire project OR call this function in some place like the project’s Startup Script, a Script Group, or a Math worksheet.
Examples

CnfEmail( "smtp.company.com", "Robert@company.com", "pop.company.com", "RobertH", "Shades556", 100 )

CnfEmail( "smtp.company.com:4455", "Robert@company.com", "pop.company.com:9900", "RobertH", "Shades556", 5, 1 )

CnfEmail( "195.11.22.33:4455", "Robert@company.com", "195.66.77.88:9900", "RobertH", "Shades556", 5, 2, "JohnS", "abcd1234" )

GetStatusSendEmailExt

Returns status of the last email sent using the SendEmailExt function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetStatusSendEmailExt</td>
<td></td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

GetStatusSendEmailExt({ | optTagName })

optTagName

Optional tag that causes the function to update its return value. This parameter is optional but you must use it when configuring this function for an object animation (e.g., Text Data Link, Position).

Returned value

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Incorrect version of the INDMail.DLL library.</td>
</tr>
<tr>
<td>-1</td>
<td>The INDMail.DLL library is corrupted.</td>
</tr>
<tr>
<td>0</td>
<td>SendEmailExt function is not being executed.</td>
</tr>
<tr>
<td>1</td>
<td>Still sending last email. Cannot execute the SendEmailExt function.</td>
</tr>
<tr>
<td>2</td>
<td>Last email was sent successfully. You can execute the SendEmailExt function again.</td>
</tr>
<tr>
<td>3</td>
<td>There was an error sending the last email. Execute the SendEmailExt function again.</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>GetStatusSendEmailExt( Second )</td>
</tr>
<tr>
<td>Tag</td>
<td>GetStatusSendEmailExt()</td>
</tr>
</tbody>
</table>

SendEmail

This function sends an email message.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendEmail</td>
<td>Email</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

Syntax

SendEmail( strSubject, strMessage, strTo )
SendEmail(\textit{strSubject}, \textit{strMessage}, \textit{strTo})

\textbf{strSubject} \\
The subject of the email.

\textbf{strMessage} \\
The message body of the email, up to 255 characters long.

\textbf{strTo} \\
The email address of the intended recipient.

\textbf{Returned value}

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Invalid format for \textit{strSubject}</td>
</tr>
<tr>
<td>2</td>
<td>Invalid format for \textit{strMessage}</td>
</tr>
<tr>
<td>3</td>
<td>Invalid format for \textit{strTo}</td>
</tr>
<tr>
<td>4</td>
<td>Wrong number of parameters</td>
</tr>
<tr>
<td>5</td>
<td>Start socket error</td>
</tr>
<tr>
<td>6</td>
<td>Error getting host IP Address (i.e., invalid SMTP server)</td>
</tr>
<tr>
<td>7</td>
<td>Error connecting to SMTP server</td>
</tr>
<tr>
<td>8</td>
<td>Error sending HELO command (i.e., initialization)</td>
</tr>
<tr>
<td>9</td>
<td>Error sending MAIL command (i.e., the &quot;From&quot; address)</td>
</tr>
<tr>
<td>10</td>
<td>Error sending RCPT command (i.e., the &quot;To&quot; address)</td>
</tr>
<tr>
<td>11</td>
<td>Error sending DATA (i.e., the message body)</td>
</tr>
<tr>
<td>12</td>
<td>Error sending SMTP authentication command</td>
</tr>
<tr>
<td>13</td>
<td>Invalid username</td>
</tr>
<tr>
<td>14</td>
<td>Invalid password</td>
</tr>
</tbody>
</table>

\textbf{Notes}

Before you can send any email, you must first configure your project's email settings either by using the \textit{E-mail Settings} dialog or by calling the \textit{CnfEmail} function. Incorrect settings can result in several different error codes (see "Returned value" above).

Also, \textit{SendEmail} cannot be used when encryption via TLS/SSL is enabled or to send an email that contains Unicode characters. Use the \textit{SendEmailExt} function instead.

\textbf{Examples}

\begin{verbatim}
SendEmail( "Hi!", "How are you?", "rogers@pnd.net" )

SendEmail( statusSummary, statusDetail, adminAddress )
\end{verbatim}

\textbf{SendEmailExt}

This function sends email messages with attached files.
Syntax

SendEmailExt(strSubject, strMessage, strTo|, optStrCc|, optStrBcc|, optStrFile1,..., optStrFileN))

strSubject
The email subject (up to 255 characters).

strMessage
The email message (up to 255 characters).

strTo
The recipient’s address. You can specify more than one recipient, using a semicolon (;) to separate the addresses.

optStrCc
The recipients’ addresses to be Cc’ed. You can specify more than one recipient, using a semicolon (;) to separate the addresses.

This is an optional parameter, but if you need to use subsequent parameters, then you can specify a null string (“”) here.

optStrBcc
The recipients’ addresses to be Bcc’ed. You can specify more than one recipient, using a semicolon (;) to separate the addresses.

This is an optional parameter, but if you need to use subsequent parameters, then you can specify a null string (“”) here.

optStrFile1...optStrFileN
Complete file paths and names of file attachments.

Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>Some or all of the specified file attachments were not found.</td>
</tr>
<tr>
<td>-3</td>
<td>Wrong number of parameters (at least three parameters are required).</td>
</tr>
<tr>
<td>-2</td>
<td>The library INDMail.DLL is the wrong version.</td>
</tr>
<tr>
<td>-1</td>
<td>The library INDMail.DLL is corrupted.</td>
</tr>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Cannot send email because another email is still pending.</td>
</tr>
<tr>
<td>2</td>
<td>Cannot send email because a new thread cannot be created.</td>
</tr>
</tbody>
</table>

Notes

Before you can send any email, you must first use the function CnfEmail to configure the email settings. Incorrect settings can result in several different error codes (see “Returned value” above).

Examples

<table>
<thead>
<tr>
<th>Tag</th>
<th>Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td></td>
<td>SendEmailExt(&quot;Subject&quot;, &quot;Message&quot;, &quot;<a href="mailto:Sam@universe.com">Sam@universe.com</a>&quot;, &quot;,&quot;, &quot;,&quot;, &quot;C:\Projects eport.txt&quot;)</td>
</tr>
<tr>
<td>Tag Name</td>
<td>Expression</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>SendEmailExt( &quot;Subject&quot;, &quot;Message&quot;, &quot;<a href="mailto:David@Ohio.net">David@Ohio.net</a>&quot;, &quot;<a href="mailto:Ted@Austin.com">Ted@Austin.com</a>&quot;, &quot;<a href="mailto:Bart@Springfield.gov">Bart@Springfield.gov</a>&quot;, &quot;C:\TechRef51.doc&quot; )</td>
<td></td>
</tr>
</tbody>
</table>
Dial-up functions

These functions are used to configure the computer’s modem (if any) and establish dial-up connections to other computers.

Note: These functions are not supported on Windows 7.

DialError

Returns the error codes regarding each connection.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialError</td>
<td>Dial-up</td>
<td>Synchronous</td>
<td>Supported (see note)</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

DialError( numType, strPhonebookEntryOrModem, "optError", optRefresh )

numType

A numeric flag that specifies the content of the strPhonebookEntryOrModem parameter.

- 0: Phonebook Name
- 1: Modem Name
- 2: Direct Connection Name

strPhonebookEntryOrModem

The Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType parameter specifies which of these methods is used.

optError

Optional The name of the string tag receiving the Error Message.

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

optRefresh

Optional tag, which causes the function to update its return value. This parameter is optional but you must use it when configuring this function for an object animation (e.g., Text Data Link, Position).

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>-1</td>
<td>Error: INDRas.DLL not found.</td>
</tr>
<tr>
<td>-2</td>
<td>Error: INDRas.DLL damaged.</td>
</tr>
<tr>
<td>-3</td>
<td>Error: invalid number of parameters (minimum=2).</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid value for the numType parameter (0 or 1).</td>
</tr>
<tr>
<td>-5</td>
<td>PhoneBook or Modem does not exist.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>600</td>
<td>An operation is pending.</td>
</tr>
<tr>
<td>601</td>
<td>The port handle is invalid.</td>
</tr>
<tr>
<td>602</td>
<td>The port is already open.</td>
</tr>
<tr>
<td>603</td>
<td>Caller's buffer is too small.</td>
</tr>
<tr>
<td>604</td>
<td>Wrong information specified.</td>
</tr>
<tr>
<td>605</td>
<td>Cannot set port information.</td>
</tr>
<tr>
<td>606</td>
<td>The port is not connected.</td>
</tr>
<tr>
<td>607</td>
<td>The event is invalid.</td>
</tr>
<tr>
<td>608</td>
<td>The device does not exist.</td>
</tr>
<tr>
<td>609</td>
<td>The device type does not exist.</td>
</tr>
<tr>
<td>610</td>
<td>The buffer is invalid.</td>
</tr>
<tr>
<td>611</td>
<td>The route is not available.</td>
</tr>
<tr>
<td>612</td>
<td>The route is not allocated.</td>
</tr>
<tr>
<td>613</td>
<td>Invalid compression specified.</td>
</tr>
<tr>
<td>614</td>
<td>Out of buffers.</td>
</tr>
<tr>
<td>615</td>
<td>The port was not found.</td>
</tr>
<tr>
<td>616</td>
<td>An asynchronous request is pending.</td>
</tr>
<tr>
<td>617</td>
<td>The port or device is already disconnecting.</td>
</tr>
<tr>
<td>618</td>
<td>The port is not open.</td>
</tr>
<tr>
<td>619</td>
<td>The port is disconnected.</td>
</tr>
<tr>
<td>620</td>
<td>There are no endpoints.</td>
</tr>
<tr>
<td>621</td>
<td>Cannot open the phone book file.</td>
</tr>
<tr>
<td>622</td>
<td>Cannot load the phone book file.</td>
</tr>
<tr>
<td>623</td>
<td>Cannot find the phone book entry.</td>
</tr>
<tr>
<td>624</td>
<td>Cannot write the phone book file.</td>
</tr>
<tr>
<td>625</td>
<td>Invalid information found in the phone book file.</td>
</tr>
<tr>
<td>626</td>
<td>Cannot load a string.</td>
</tr>
<tr>
<td>627</td>
<td>Cannot find key.</td>
</tr>
<tr>
<td>628</td>
<td>The port was disconnected.</td>
</tr>
<tr>
<td>629</td>
<td>The data link was terminated by the remote machine.</td>
</tr>
<tr>
<td>630</td>
<td>The port was disconnected due to hardware failure.</td>
</tr>
<tr>
<td>631</td>
<td>The port was disconnected by the user.</td>
</tr>
<tr>
<td>632</td>
<td>The structure size is incorrect.</td>
</tr>
<tr>
<td>633</td>
<td>The port is already in use or is not configured for Remote Access dial out.</td>
</tr>
<tr>
<td>634</td>
<td>Cannot register your computer on on the remote network.</td>
</tr>
<tr>
<td>635</td>
<td>Unknown error.</td>
</tr>
<tr>
<td>636</td>
<td>The wrong device is attached to the port.</td>
</tr>
<tr>
<td>637</td>
<td>The string could not be converted.</td>
</tr>
<tr>
<td>638</td>
<td>The request has timed out.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>639</td>
<td>No asynchronous net available.</td>
</tr>
<tr>
<td>640</td>
<td>A NetBIOS error occurred.</td>
</tr>
<tr>
<td>641</td>
<td>The server cannot allocate NetBIOS resources needed to support the client.</td>
</tr>
<tr>
<td>642</td>
<td>One of your NetBIOS names is already registered on the remote network.</td>
</tr>
<tr>
<td>643</td>
<td>A network adapter at the server failed.</td>
</tr>
<tr>
<td>644</td>
<td>You will not receive network message pop-ups.</td>
</tr>
<tr>
<td>645</td>
<td>Internal authentication error.</td>
</tr>
<tr>
<td>646</td>
<td>The account is not permitted to log on at this time of day.</td>
</tr>
<tr>
<td>647</td>
<td>The account is disabled.</td>
</tr>
<tr>
<td>648</td>
<td>The password has expired.</td>
</tr>
<tr>
<td>649</td>
<td>The account does not have Remote Access permission.</td>
</tr>
<tr>
<td>650</td>
<td>The Remote Access server is not responding.</td>
</tr>
<tr>
<td>651</td>
<td>Your modem (or other connecting device) has reported an error.</td>
</tr>
<tr>
<td>652</td>
<td>Unrecognized response from the device.</td>
</tr>
<tr>
<td>653</td>
<td>A macro required by the device was not found in the device .INF file section.</td>
</tr>
<tr>
<td>654</td>
<td>A command or response in the device .INF file section refers to an undefined macro.</td>
</tr>
<tr>
<td>655</td>
<td>The &lt;message macro was not found in the device .INF file section.</td>
</tr>
<tr>
<td>656</td>
<td>The &lt;defaultoff macro in the device .INF file section contains an undefined macro.</td>
</tr>
<tr>
<td>657</td>
<td>The device .INF file could not be opened.</td>
</tr>
<tr>
<td>658</td>
<td>The device name in the device .INF or media .INI file is too long.</td>
</tr>
<tr>
<td>659</td>
<td>The media .INI file refers to an unknown device name.</td>
</tr>
<tr>
<td>660</td>
<td>The device .INF file contains no responses for the command.</td>
</tr>
<tr>
<td>661</td>
<td>The device .INF file is missing a command.</td>
</tr>
<tr>
<td>662</td>
<td>Attempted to set a macro not listed in device .INF file section.</td>
</tr>
<tr>
<td>663</td>
<td>The media .INI file refers to an unknown device type.</td>
</tr>
<tr>
<td>664</td>
<td>Cannot allocate memory.</td>
</tr>
<tr>
<td>665</td>
<td>The port is not configured for Remote Access.</td>
</tr>
<tr>
<td>666</td>
<td>Your modem (or other connecting device) is not functioning.</td>
</tr>
<tr>
<td>667</td>
<td>Cannot read the media .INI file.</td>
</tr>
<tr>
<td>668</td>
<td>The connection dropped.</td>
</tr>
<tr>
<td>669</td>
<td>The usage parameter in the media .INI file is invalid.</td>
</tr>
<tr>
<td>670</td>
<td>Cannot read the section name from the media .INI file.</td>
</tr>
<tr>
<td>671</td>
<td>Cannot read the device type from the media .INI file.</td>
</tr>
<tr>
<td>672</td>
<td>Cannot read the device name from the media .INI file.</td>
</tr>
<tr>
<td>673</td>
<td>Cannot read the usage from the media .INI file.</td>
</tr>
<tr>
<td>674</td>
<td>Cannot read the maximum connection BPS rate from the media .INI file.</td>
</tr>
<tr>
<td>675</td>
<td>Cannot read the maximum carrier BPS rate from the media .INI file.</td>
</tr>
<tr>
<td>676</td>
<td>The line is busy.</td>
</tr>
<tr>
<td>677</td>
<td>A person answered instead of a modem.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>678</td>
<td>There is no answer.</td>
</tr>
<tr>
<td>679</td>
<td>Cannot detect carrier.</td>
</tr>
<tr>
<td>680</td>
<td>There is no dial tone.</td>
</tr>
<tr>
<td>681</td>
<td>General error reported by device.</td>
</tr>
<tr>
<td>682</td>
<td>ERROR_WRITING_SECTIONNAME</td>
</tr>
<tr>
<td>683</td>
<td>ERROR_WRITING_DEVICETYPE</td>
</tr>
<tr>
<td>684</td>
<td>ERROR_WRITING_DEVICENAME</td>
</tr>
<tr>
<td>685</td>
<td>ERROR_WRITING_MAXCONNECTBPS</td>
</tr>
<tr>
<td>686</td>
<td>ERROR_WRITING_MAXCARRIERBPS</td>
</tr>
<tr>
<td>687</td>
<td>ERROR_WRITING_USAGE</td>
</tr>
<tr>
<td>688</td>
<td>ERROR_WRITING_DEFAULTOFF</td>
</tr>
<tr>
<td>689</td>
<td>ERROR_READING_DEFAULTOFF</td>
</tr>
<tr>
<td>690</td>
<td>ERROR_EMPTY_INI_FILE</td>
</tr>
<tr>
<td>691</td>
<td>Access denied because username and/or password is invalid on the domain.</td>
</tr>
<tr>
<td>692</td>
<td>Hardware failure in port or attached device.</td>
</tr>
<tr>
<td>693</td>
<td>ERROR_NOT_BINARY_MACRO</td>
</tr>
<tr>
<td>694</td>
<td>ERROR_DCB_NOT_FOUND</td>
</tr>
<tr>
<td>695</td>
<td>ERROR_STATE_MACHINES_NOT_STARTED</td>
</tr>
<tr>
<td>696</td>
<td>ERROR_STATE_MACHINES_ALREADY_STARTED</td>
</tr>
<tr>
<td>697</td>
<td>ERROR_PARTIAL_RESPONSE_LOOPING</td>
</tr>
<tr>
<td>698</td>
<td>A response keyname in the device .INF file is not in the expected format.</td>
</tr>
<tr>
<td>699</td>
<td>The device response caused buffer overflow.</td>
</tr>
<tr>
<td>700</td>
<td>The expanded command in the device .INF file is too long.</td>
</tr>
<tr>
<td>701</td>
<td>The device moved to a BPS rate not supported by the COM driver.</td>
</tr>
<tr>
<td>702</td>
<td>Device response received when none expected.</td>
</tr>
<tr>
<td>703</td>
<td>The Application does not allow user interaction the connection requires interaction with the user to complete successfully</td>
</tr>
<tr>
<td>704</td>
<td>ERROR_BAD_CALLBACK_NUMBER</td>
</tr>
<tr>
<td>705</td>
<td>ERROR_INVALID_AUTH_STATE</td>
</tr>
<tr>
<td>706</td>
<td>ERROR_WRITING_INITBPS</td>
</tr>
<tr>
<td>707</td>
<td>X.25 diagnostic indication.</td>
</tr>
<tr>
<td>708</td>
<td>The account has expired.</td>
</tr>
<tr>
<td>709</td>
<td>Error changing password on domain The password may be too short or may match a previously used password.</td>
</tr>
<tr>
<td>710</td>
<td>Serial overrun errors were detected while communicating with your modem.</td>
</tr>
<tr>
<td>711</td>
<td>RasMan initialization failure Check the event log.</td>
</tr>
<tr>
<td>712</td>
<td>Biplex port initializing Wait a few seconds and redial.</td>
</tr>
<tr>
<td>713</td>
<td>No active ISDN lines are available.</td>
</tr>
<tr>
<td>714</td>
<td>No ISDN channels are available to make the call.</td>
</tr>
<tr>
<td>715</td>
<td>Too many errors occurred because of poor phone line quality.</td>
</tr>
<tr>
<td>716</td>
<td>The Remote Access IP configuration is unusable.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>717</td>
<td>No IP addresses are available in the static pool of Remote Access IP addresses.</td>
</tr>
<tr>
<td>718</td>
<td>Timed out waiting for a valid response from the remote PPP peer.</td>
</tr>
<tr>
<td>719</td>
<td>PPP terminated by remote machine.</td>
</tr>
<tr>
<td>720</td>
<td>No PPP control protocols configured.</td>
</tr>
<tr>
<td>721</td>
<td>Remote PPP peer is not responding.</td>
</tr>
<tr>
<td>722</td>
<td>The PPP packet is invalid.</td>
</tr>
<tr>
<td>723</td>
<td>The phone number including prefix and suffix is too long.</td>
</tr>
<tr>
<td>724</td>
<td>The IPX protocol cannot dial-out on the port because the machine is an IPX router.</td>
</tr>
<tr>
<td>725</td>
<td>The IPX protocol cannot dial-in on the port because the IPX router is not installed.</td>
</tr>
<tr>
<td>726</td>
<td>The IPX protocol cannot be used for dial-out on more than one port at a time.</td>
</tr>
<tr>
<td>727</td>
<td>Cannot access TCPPCFG.DLL.</td>
</tr>
<tr>
<td>728</td>
<td>Cannot find an IP adapter bound to Remote Access.</td>
</tr>
<tr>
<td>729</td>
<td>SLIP cannot be used unless the IP protocol is installed.</td>
</tr>
<tr>
<td>730</td>
<td>Computer registration is not complete.</td>
</tr>
<tr>
<td>731</td>
<td>The protocol is not configured.</td>
</tr>
<tr>
<td>732</td>
<td>The PPP negotiation is not converging.</td>
</tr>
<tr>
<td>733</td>
<td>The PPP control protocol for this network protocol is not available on the server.</td>
</tr>
<tr>
<td>734</td>
<td>The PPP link control protocol terminated.</td>
</tr>
<tr>
<td>735</td>
<td>The requested address was rejected by the server.</td>
</tr>
<tr>
<td>736</td>
<td>The remote computer terminated the control protocol.</td>
</tr>
<tr>
<td>737</td>
<td>Loopback detected.</td>
</tr>
<tr>
<td>738</td>
<td>The server did not assign an address.</td>
</tr>
<tr>
<td>739</td>
<td>The authentication protocol required by the remote server cannot use the Windows NT encrypted password Redial, entering the password explicitly.</td>
</tr>
<tr>
<td>740</td>
<td>Invalid TAPI configuration.</td>
</tr>
<tr>
<td>741</td>
<td>The local computer does not support the required encryption type.</td>
</tr>
<tr>
<td>742</td>
<td>The remote computer does not support the required encryption type.</td>
</tr>
<tr>
<td>743</td>
<td>The remote computer requires encryption.</td>
</tr>
<tr>
<td>744</td>
<td>Cannot use the IPX network number assigned by remote server Check the event log.</td>
</tr>
<tr>
<td>745</td>
<td>ERROR_INVALID_SMM</td>
</tr>
<tr>
<td>746</td>
<td>ERROR_SMM_UNINITIALIZED</td>
</tr>
<tr>
<td>747</td>
<td>ERROR_NO_MAC_FOR_PORT</td>
</tr>
<tr>
<td>748</td>
<td>ERROR_SMM_TIMEOUT</td>
</tr>
<tr>
<td>749</td>
<td>ERROR_BAD_PHONE_NUMBER</td>
</tr>
<tr>
<td>750</td>
<td>ERROR_WRONG_MODULE</td>
</tr>
<tr>
<td>751</td>
<td>Invalid callback number Only the characters 0 to 9, T, P, W, ( ), - , @, and space are allowed in the number.</td>
</tr>
<tr>
<td>752</td>
<td>A syntax error was encountered while processing a script.</td>
</tr>
<tr>
<td>753</td>
<td>The connection could not be disconnected because it was created by the Multi-Protocol Router.</td>
</tr>
</tbody>
</table>
Notes
This function is not supported on Windows 7.

Examples

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>DialError( 0, &quot;Office DialUp&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>DialError( 1, &quot;USRobotics_SportsterFaxModem&quot;, &quot;StatusMessage&quot;, second )</td>
</tr>
<tr>
<td>Tag</td>
<td>DialError( 2, &quot;DirectDial&quot;, &quot;DialupError&quot; )</td>
</tr>
</tbody>
</table>

See also: FindModem()

DialGetClientIP
This function gets the IP address of a Remote Access Service (RAS) client station.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialGetClientIP</td>
<td>Dial-Up</td>
<td>Synchronous</td>
<td>Supported (see note)</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

DialGetClientIP({ numType | 0 | 1 | 2 }, strPhoneBookOrModem, "tagClientIP", | optTagRefresh })

numType
The type of information specified by strPhoneBookOrModem: 0 is a Phonebook Name, 1 is a Modem Name, and 2 is a Direct Connection Name.

strPhoneBookOrModem
The Phonebook Name, Modem Name, or Direct Connection Name used to make the connection.

tagClientIP
The name of a String tag that will receive the IP address.

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

optTagRefresh
The name of a tag that, whenever the value of the tag changes, will trigger the function to refresh its returned value.

Also, this parameter is optional but you must include it when you configure the function on an object animation such as Text Data Link or Position.

Returned value
This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>DialGetClientIP was not found in IndRAS.dll.</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid type; check numType.</td>
</tr>
<tr>
<td>-3</td>
<td>Incorrect number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>DialStatus was not found in IndRAS.dll.</td>
</tr>
<tr>
<td>-1</td>
<td>IndRAS.dll was not loaded.</td>
</tr>
</tbody>
</table>
This is the value returned by the function itself. The IP address is written to the tag specified by tagClientIP.

**Notes**
This function must be executed on the RAS client station, not on the server station. (Essentially, the station gets its own IP address.)

Also, this function is not supported on Windows 7.

**Examples**
Use the Phonebook Name "Office DialUp” and write the resulting IP address to ClientIPTag:

```
DialGetClientIP( 0, "Office DialUp", "ClientIPTag" )
```

Use the Modem Name "USRobotics_SportsterFaxModem," write the resulting IP address to ClientIPAddress, and refresh every second (that is, every time the system tag Second changes):

```
DialGetClientIP( 1, "USRobotics_SportsterFaxModem", "ClientIPAddress", Second )
```

Use the Direct Connection Name "DirectDial" and write the resulting IP address to IPAdd:

```
DialGetClientIP( 2, "DirectDial", "IPAdd" )
```

**DialGetServerIP**
DialGetServerIP is a built-in scripting function that gets the IP address of a Remote Access Service (RAS) server station.

<table>
<thead>
<tr>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dial-up</td>
<td>Synchronous</td>
<td>Supported (see Notes)</td>
<td>Not Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**
```
DialGetServerIP({ numType | 0 | 1 | 2 }, strPhoneBookOrModem, "tagServerIP"{ | , optTagRefresh })
```

**numType**
The type of information specified by strPhoneBookOrModem: 0 is a Phonebook Name, 1 is a Modem Name, and 2 is a Direct Connection Name.

**strPhoneBookOrModem**
The Phonebook Name, Modem Name, or Direct Connection Name used to make the connection.

**tagServerIP**
The name of a String tag that will receive the IP address.

\[\textbf{Note:}\] The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

**optTagRefresh**
The name of a tag that, whenever the value of the tag changes, will trigger the function to refresh its returned value.

Also, this parameter is optional but you must include it when you configure the function on an object animation such as Text Data Link or Position.
Returned value

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>DialGetServerIP was not found in IndRAS.dll.</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid type; check numType.</td>
</tr>
<tr>
<td>-3</td>
<td>Incorrect number of parameters.</td>
</tr>
<tr>
<td>-2</td>
<td>DialStatus was not found in IndRAS.dll.</td>
</tr>
<tr>
<td>-1</td>
<td>IndRAS.dll was not loaded.</td>
</tr>
<tr>
<td>n</td>
<td>Status code returned by DialStatus.</td>
</tr>
</tbody>
</table>

This is the value returned by the function itself. The IP address is written to the tag specified by tagServerIP.

Notes

This function must be executed on the RAS client station, not on the server station. (Essentially, the station gets the IP address of the server to which it is connected.)

Also, this function is not supported on Windows 7.

Examples

Use the Phonebook Name "Office DialUp" and write the resulting IP address to ServerIPTag:

```
DialGetServerIP( 0, "Office DialUp", "ServerIPTag" )
```

Use the Modem Name "USRRobotics_SportsterFaxModem," write the resulting IP address to ServerIPAddress, and refresh every second (that is, every time the system tag Second changes):

```
DialGetServerIP( 1, "USRRobotics_SportsterFaxModem", "ServerIPAddress", Second )
```

Use the Direct Connection Name "DirectDial" and write the resulting IP address to IPAdd:

```
DialGetServerIP( 2, "DirectDial", "IPAdd" )
```

DialStatus

Returns the status of the dial-up connection.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialStatus</td>
<td>Dial-up</td>
<td>Synchronous</td>
<td>Supported (see note)</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Note:** This function is not supported on Windows 7.

Syntax

```
DialStatus( numType, strPhonebookEntryOrModem, "optStatus", optRefresh )
```

**numType**

A numeric flag that specifies the content of the strPhonebookEntryOrModem parameter.

- 0: Phonebook Name
• 1: Modem Name
• 2: Direct Connection Name

strPhonebookEntryOrModem
The Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType parameter specifies which of these methods is used.

optStatus
Optional The name of the string tag receiving the status message.

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

optRefresh
Optional tag that causes the function to update its return value. This parameter is optional, but you must use it when configuring this function for an object animation (e.g., Text Data Link, Position).

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>PhoneBook or modem does not exist</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid value for the numType parameter (0 or 1)</td>
</tr>
<tr>
<td>-3</td>
<td>Error: invalid number of parameters (minimum = 2)</td>
</tr>
<tr>
<td>-2</td>
<td>Error: INDRAS.DLL damaged</td>
</tr>
<tr>
<td>-1</td>
<td>Error: INDRAS.DLL not found</td>
</tr>
<tr>
<td>0</td>
<td>Opening the port…</td>
</tr>
<tr>
<td>1</td>
<td>Port was opened successfully.</td>
</tr>
<tr>
<td>2</td>
<td>Connecting to the device…</td>
</tr>
<tr>
<td>3</td>
<td>The device has connected successfully.</td>
</tr>
<tr>
<td>4</td>
<td>All devices in the device chain have successfully connected.</td>
</tr>
<tr>
<td>5</td>
<td>Verifying the user name and password…</td>
</tr>
<tr>
<td>6</td>
<td>An authentication event has occurred.</td>
</tr>
<tr>
<td>7</td>
<td>Requested another validation attempt with a new user.</td>
</tr>
<tr>
<td>8</td>
<td>Server has requested a callback number.</td>
</tr>
<tr>
<td>9</td>
<td>The client has requested to change the password</td>
</tr>
<tr>
<td>10</td>
<td>Registering your computer on the network…</td>
</tr>
<tr>
<td>11</td>
<td>The link-speed calculation phase is starting…</td>
</tr>
<tr>
<td>12</td>
<td>An authentication request is being acknowledged.</td>
</tr>
<tr>
<td>13</td>
<td>Reauthentication (after callback) is starting.</td>
</tr>
<tr>
<td>14</td>
<td>The client has successfully completed authentication.</td>
</tr>
<tr>
<td>15</td>
<td>The line is about to disconnect for callback.</td>
</tr>
<tr>
<td>16</td>
<td>Delaying to give the modem time to reset for callback.</td>
</tr>
<tr>
<td>17</td>
<td>Waiting for an incoming call from server.</td>
</tr>
<tr>
<td>18</td>
<td>Projection result information is available.</td>
</tr>
<tr>
<td>19</td>
<td>User authentication is being initiated or retried.</td>
</tr>
</tbody>
</table>
### Value | Description
--- | ---
20 | Client has been called back and is about to resume authentication.
21 | Logging on to the network…
22 | Subentry has been connected.
23 | Subentry has been disconnected.
24 | Terminal state supported by `RASPHONE.EXE`.
25 | Retry authentication state supported by `RASPHONE.EXE`.
26 | Callback state supported by `RASPHONE.EXE`.
27 | Change password state supported by `RASPHONE.EXE`.
8192 | Connected to remote server successfully!
8193 | Disconnected.

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>DialStatus( 0, &quot;Office DialUp&quot; )</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>DialStatus( 1, &quot;USRobotics_SportsterFaxModem&quot;, &quot;StatusMessage&quot;, second )</code></td>
</tr>
<tr>
<td>Tag</td>
<td><code>DialStatus( 2, &quot;DirectDial&quot;, &quot;DialupError&quot;)</code></td>
</tr>
</tbody>
</table>

See also: `FindModem()`

### `DialUp`

Establishes a dial-up connection.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialUp</td>
<td>Dial-up</td>
<td>Asynchronous</td>
<td>Supported (see note)</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Note:** This function is not supported on Windows 7.

### Syntax

```plaintext
DialUp( numType, strPhonebookEntryOrModem, strUserName, strPassword, optStrDomain, strPhoneNumber )
```

#### `numType`

A numeric flag that specifies the content of the `strPhonebookEntryOrModem` parameter.
- 0: Phonebook Name
- 1: Modem Name
- 2: Direct Connection Name

#### `strPhonebookEntryOrModem`

The Phonebook Name, Modem Name, or Direct Connection Name used to make the connection.
- The `numType` parameter specifies which of these methods is used.

#### `strUserName`

```plaintext
```
The Username to use for logging on.

**strPassword**

The Password to use for logging on.

**optStrDomain**

*Optional* The domain name to specify when logging on.

**strPhoneNumber**

The phone number to dial (used only when the parameter numType=1).

### Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK: dialing started</td>
</tr>
<tr>
<td>-1</td>
<td>Error: INDRA.S.DLL not found</td>
</tr>
<tr>
<td>-2</td>
<td>Error: INDRA.S.DLL damaged</td>
</tr>
<tr>
<td>-3</td>
<td>Error: invalid number of parameters (minimum=5)</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid value for the numType parameter (0 or 1)</td>
</tr>
<tr>
<td>-5</td>
<td>Invalid value for the strPhonebookEntryOrModem parameter (string)</td>
</tr>
<tr>
<td>-6</td>
<td>PhoneBook or Modem does not exist</td>
</tr>
<tr>
<td>-7</td>
<td>PhoneBook or Modem is in use</td>
</tr>
</tbody>
</table>
| -8    | Depends on the numType parameter:  
  • If numType = 0: Could not read properties from PhoneBook.  
  • If numType = 1: More than 1000 connections are enabled at same time. |
| -9    | Unable to create a temporary PhoneBook. |

### Notes

The operating system’s RAS Server executes the dial-in for Windows PC stations automatically.

### Examples

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>DialUp( 0, &quot;OfficeDialup&quot;, &quot;Guest&quot;, &quot;Password&quot; )</code></td>
</tr>
</tbody>
</table>
| Tag | `DialUp( 1, "USRobotics_SportsterFaxModem", "HR12378", "HRPass",  
"15125554321" )` |
| Tag | `DialUp( 2, "DirectDial", "Rberton", "MyPassword", "156.48.25.0")` |

See also: [FindModem()](#)

### DialUpToCE

Runs the DialUpToCE program, which sends the information necessary to CERAsSvr.exe calls back to the Server.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DialUpToCE</td>
<td>Dial-up</td>
<td>Asynchronous</td>
<td>Supported (see note)</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

⚠️ **Note:** This function is not supported on Windows 7.
Appendix: Built-in Scripting Language

Syntax

DialUpToCE( numModem, strDialPhone, strMyNumber, strUser, strPassword, optStrDomain, optAutoDial )

numModem
The modem used to dial to the Windows CE remote station.

strDialPhone
The telephone number of the Windows CE remote station.

strMyNumber
The telephone number sent to the Windows CE remote station. CERasSvr.exe will call back to this phone number.

strUser
The user name to be sent to Windows CE remote station. CERasSvr.exe will use this name to connect to the Windows XP/Vista/7 computer after calling back to it.

strPassword
The password to be sent to Windows CE remote station. CERasSvr.exe will use this password to connect to the Windows XP/Vista/7 computer after calling back to it.

optStrDomain
Optional The domain name to specify when logging on.

optAutoDial
Optional tag, which can be set to one of the following:
• 1: Triggers the DialupToCE connection automatically when the function is executed
• 0: Requests confirmation before triggering the DialupToCE connection automatically when the function is executed

optAutoClose
Optional tag, which can be set to one of the following:
• 1: Closes the DialUpToCE dialog automatically after dialing the Windows CE remote station
• 0: Leaves the DialUpToCE dialog open

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Fail, unable to call DialUpToCE.</td>
</tr>
<tr>
<td>1</td>
<td>Success, DialUpToCE executed.</td>
</tr>
</tbody>
</table>

Notes
The DialUpToCE program was developed to dial a remote Windows CE station. Because Windows CE v3.00 does not provide a RAS Server, you must be running the CERasSvr.exe program on the Windows Embedded device to answer a call, and call back to a Windows XP/Vista/7 computer using parameters sent by the DialUpToCE function. You must configure the RAS Server service on the Windows XP/Vista/7 computer to answer the call back from the Windows Embedded device and set the TCP/IP connection.

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>DialUpToCE( 0, &quot;12344321&quot;, &quot;98765432&quot;, &quot;Administrator&quot;, &quot;MyPass&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>DialUpToCE( 0, &quot;12344321&quot;, &quot;98765432&quot;, &quot;Administrator&quot;, &quot;MyPass&quot;, &quot;,&quot;, 1, 1 )</td>
</tr>
</tbody>
</table>
**FindAllDevices**

Returns the list of all the available modems and direct connection interfaces (COM ports) in the local station.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindAllDevices</td>
<td>Dial-up</td>
<td>Synchronous</td>
<td>Supported (see note)</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Note:** This function is not supported on Windows 7.

**Syntax**

```
FindAllDevices( "tagArray" )
```

**tagArray**

Name of a string array tag receiving the list of available modems and direct connection interfaces.

**Returned value**

Returns the number of modems and/or interfaces found.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>FindAllDevices( &quot;SerialConnections[1]&quot; )</td>
</tr>
</tbody>
</table>

**FindModem**

Returns the list of all available modems in the local station.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindModem</td>
<td>Dial-up</td>
<td>Synchronous</td>
<td>Supported (see note)</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Note:** This function is not supported on Windows 7.

**Syntax**

```
FindModem( "tagArray" )
```

**tagArray**

Name of a string array tag receiving the list of available modems.

**Returned value**

Returns the number of modems found.

**Notes**

You can use this function to get the serial interface name for a dial-up connection via modem, and then use this information to fill the strPhonebookEntryOrModem parameter for the DialError, DialStatus, DialUp, and HangUp functions.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>FindModem( &quot;Modems[1]&quot; )</td>
</tr>
</tbody>
</table>

**HangUp**

Hangs-up a dial-up connection.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>HangUp</td>
<td>Dial-up</td>
<td>Synchronous</td>
<td>Supported (see note)</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Note:** This function is not supported on Windows 7.

**Syntax**

```plaintext
HangUp( numType, strPhonebookEntryOrModem )
```

**numType**

A numeric flag that specifies the content of the strPhonebookEntryOrModem parameter.
- 0: Phonebook Name
- 1: Modem Name
- 2: Direct Connection Name

**strPhonebookEntryOrModem**

The Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType parameter specifies which of these methods is used.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>-1</td>
<td>Error: INDRAS.DLL not found</td>
</tr>
<tr>
<td>-2</td>
<td>Error: INDRAS.DLL damaged</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid value for the numType parameter (0 or 1)</td>
</tr>
<tr>
<td>-4</td>
<td>PhoneBook or modem does not exist</td>
</tr>
<tr>
<td>-5</td>
<td>No configured modems exist</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>HangUp( 0, &quot;OfficeDialup&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>HangUp( 1, &quot;USRobotics_SportsterFaxModem&quot; )</td>
</tr>
<tr>
<td>Tag</td>
<td>HangUp( 2, &quot;DirectDial&quot;, &quot;Rberton&quot;, &quot;MyPassword&quot; )</td>
</tr>
</tbody>
</table>

See also: FindModem()
**PhoneDialUp**

Dials to a phone number using Telephony Application Program Interface (TAPI).

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhoneDialUp</td>
<td>Dial-up</td>
<td>Asynchronous</td>
<td>Supported (see note)</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Note:** This function is not supported on Windows 7.

**Syntax**

```plaintext
PhoneDialUp( strPhoneNumber, optStrModemName )
```

**strPhoneNumber**

Telephone number the function will call.

**optStrModemName**

Name of the modem used to dial. If you do not specify a modem, the function will use the first modem found on the operating system.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK (dial triggered)</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>-3</td>
<td>INDTAPI.DLL library not found</td>
</tr>
<tr>
<td>-4</td>
<td>PhoneDialUp () function not supported by the current INDTAPI.DLL library</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PhoneDialUp( &quot;512-123-4567&quot; )</td>
</tr>
<tr>
<td></td>
<td>PhoneDialUp( StringPhoneNumberTag )</td>
</tr>
<tr>
<td></td>
<td>PhoneDialUp( StringPhoneNumberTag, StringModemNameTag )</td>
</tr>
</tbody>
</table>

**PhoneDisableListen**

Stops listening to the modem for incoming calls.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhoneDisableListen</td>
<td>Dial-up</td>
<td>Synchronous</td>
<td>Supported (see note)</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Note:** This function is not supported on Windows 7.

**Syntax**

```plaintext
PhoneDisableListen( optStrModemName )
```

**optStrModemName**
Name of the modem used to dial. If you do not specify a modem, the function will use the first modem found on the operating system.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OK (stop listening for incoming calls)</td>
</tr>
<tr>
<td>-1</td>
<td>INDTAPI.DLL library not found</td>
</tr>
<tr>
<td>-2</td>
<td>PhoneDisableListen () function not supported by the current INDTAPI.DLL library</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PhoneDisableListen()</td>
</tr>
<tr>
<td></td>
<td>PhoneDisableListen( &quot;Hayes Compatible Modem on COM1&quot; )</td>
</tr>
<tr>
<td></td>
<td>PhoneDisableListen( StringModemNameTag )</td>
</tr>
</tbody>
</table>

See also: **PhoneEnableListen()**

**PhoneEnableListen**

Resumes listening to the modem for incoming calls.

Syntax

```
PhoneEnableListen( optStrModemName )
```

**optStrModemName**

Name of the modem used to dial. If you do not specify a modem, the function will use the first modem found on the operating system.

Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OK (listening for incoming calls)</td>
</tr>
<tr>
<td>0</td>
<td>Error executing the PhoneEnableListen() function</td>
</tr>
<tr>
<td>-1</td>
<td>INDTAPI.DLL library not found</td>
</tr>
<tr>
<td>-2</td>
<td>PhoneEnableListen () function not supported by the current INDTAPI.DLL library</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PhoneEnableListen()</td>
</tr>
<tr>
<td></td>
<td>PhoneEnableListen( &quot;Hayes Compatible Modem on COM1&quot; )</td>
</tr>
<tr>
<td></td>
<td>PhoneEnableListen( StringModemNameTag )</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

See also: PhoneDisableListen()

**PhoneHangUp**

Hangs up a dial-up connection previously established with the PhoneDialUp function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhoneHangUp</td>
<td>Dial-up</td>
<td>Asynchronous</td>
<td>Supported (see note)</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

⚠️ **Note:** This function is not supported on Windows 7.

**Syntax**

PhoneHangUp( optStrModemName )

*optStrModemName*

Name of the modem used to dial. If you do not specify a modem, the function will use the first modem found on the operating system.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OK (dial connection was dropped)</td>
</tr>
<tr>
<td>-1</td>
<td>INDTAPI.DLL library not found</td>
</tr>
<tr>
<td>-2</td>
<td>PhoneHangUp () function not supported by the current INDTAPI.DLL library</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PhoneHangUp()</td>
</tr>
<tr>
<td></td>
<td>PhoneHangUp( &quot;Hayes Compatible Modem on COM1&quot; )</td>
</tr>
<tr>
<td></td>
<td>PhoneHangUp( StringModemNameTag )</td>
</tr>
</tbody>
</table>

**PhoneStatus**

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhoneStatus</td>
<td>Dial-up</td>
<td>Synchronous</td>
<td>Supported (see note)</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

⚠️ **Note:** This function is not supported on Windows 7.

**Description**

Checks the status of the current connections.

**Syntax**

PhoneStatus ( "strStatus", optStrModemName )

*strStatus*
Name of the tag that will receive the status description text

**optStrModemName**
Name of the modem used to dial. If you do not specify a modem, the function will use the first modem found on the operating system

**Returned value**

<table>
<thead>
<tr>
<th>Returned value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;Ready to make a call&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;Call was shut down&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;Line Ringing&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;Dial Tone&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Dialing Call&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;Call is Proceeding&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;Ring Back&quot;</td>
</tr>
<tr>
<td>7</td>
<td>&quot;Line is Busy&quot;</td>
</tr>
<tr>
<td>8</td>
<td>&quot;Line is Idle&quot;</td>
</tr>
<tr>
<td>9</td>
<td>&quot;Disconnected&quot;</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag Status Code</td>
<td>PhoneStatus(&quot;String Tag Status&quot;)</td>
</tr>
<tr>
<td>Tag Status Code</td>
<td>PhoneStatus(&quot;String Tag Status&quot;, &quot;Hayes Compatible Modem on COM1&quot;)</td>
</tr>
<tr>
<td>Tag Status Code</td>
<td>PhoneStatus(&quot;String Tag Status&quot;, &quot;StringModemNameTag&quot;)</td>
</tr>
</tbody>
</table>
ActiveX and .NET Control functions

These functions are used to directly run ActiveX and .NET Control objects in the project, as well as to get and set property values on those objects.

**XGet**

The function `XGet` gets the current value of a Property on an ActiveX Control or .NET Control object.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>XGet</td>
<td>ActiveX and .NET Control</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>ActiveX Controls only (see &quot;Notes&quot; below)</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
XGet(strName, strProperties)
```

**strName**
The unique name of the ActiveX Control or .NET Control object, as specified in the **Name** box in the **Object Properties** dialog box.

**strProperties**
The Property for which you want to get the value. Available Properties are listed in the **Configuration** (for an ActiveX Control) or **Members** (for a .NET Control) dialog box.

**Returned value**
This function returns the value of the specified Property.

**Notes**
If the option **Enforce Web functionality equivalence** is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see **Preferences tab** on page 108.

Also, this function is not supported for .NET Control objects running on a Windows Embedded device.

**Examples**
Get the current value of the Color property on the ActiveX Control object named "ActXRec":

```
XGet("ActXRec","Color")
```

**XRun**

The function `XRun` runs a Method on an ActiveX Control or .NET Control object.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>XRun</td>
<td>ActiveX and .NET Control</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>ActiveX Controls only (see &quot;Notes&quot; below)</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

Syntax

XRun(strName, strMethod, Parameter1,…, ParameterN)

XRun (strName, strMethod, Parameter1, ParameterN)

strName
The unique name of the ActiveX Control or .NET Control object, as specified in the Name box in the Object Properties dialog box.

strMethod
The Method that you want to run. Available Methods are listed in the Configuration (for an ActiveX Control) or Members (for a .NET Control) dialog box.

Parameter(1…N)
Data of various types that are required by the Method to run. The number of parameters can range from 0 to 255 and depends on the specified Method. The data types (e.g., Boolean, Integer, Real or String) of referring tags must match the parameters on the Method.

Returned value
This function returns the Method result as reported by the ActiveX Control or .NET Control object. Not all Methods return results.

Notes
If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

Also, this function is not supported for .NET Control objects running on a Windows Embedded device.

Examples
Run the XPos method on the ActiveX Control named "ActXCir," with four original values passed to the method:

XRun("ActXCir","XPos",FALSE,12,4.6,"This is my text.")

Run the XPos method on the ActiveX Control named "ActXCir," with four referring tags passed to the method:

XRun("ActXCir","XPos",TagA,TagB,TagC,TagD)

XSet
The function XSet sets the value of a Property on an ActiveX Control or .NET Control object.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSet</td>
<td>ActiveX and .NET Control</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>ActiveX Controls only (see &quot;Notes&quot; below)</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

Syntax

XSet(strName, strProperties, Value)

XSet (strName, strProperties, Value)
**strName**

The unique name of the ActiveX Control or .NET Control object, as specified in the Name box in the Object Properties dialog box.

**strProperties**

The Property that you want to set the value of. Available Properties are listed in the Configuration (for an ActiveX Control) or Members (for a .NET Control) dialog box.

**Value**

A tag, expression, or data value of any type; the value to which you want to set the Property.

**Returned value**

This function does not return any value.

**Notes**

If the option Enforce Web functionality equivalence is selected in the project settings, this function cannot be called from Global Procedures or Script worksheets. This is because the function is meaningful only when it is executed on stations that display project screens; it might cause unexpected behavior if it is called from background tasks that are executed on the project runtime server. For more information, see Preferences tab on page 108.

Also, this function is not supported for .NET Control objects running on a Windows Embedded device.

**Examples**

Set the value of the Display property on the ActiveX Control named "ActXDisplay" to "Status Normal":

```
XSet("ActXDisplay","Display","Status Normal")
```
Event Logger functions

These functions are used to send events and comments to the Event Logger.

**SendEvent**

Sends an event to the Event Logger.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendEvent</td>
<td>Event Logger</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

**Description**

Use to send an event to the Event Log file. When the Comment option is enabled, the user is prompted to enter a comment after executing the SendEvent() function. This comment will be saved in the Event Log file.

**Syntax**

SendEvent( strEvent, optBoolFlag, optStrComment, optCustom1, ..., optCustom10 )

- **strEvent**
  The text to be saved in the Event Logger.

- **optBoolFlag**
  If omitted or 0 (zero), the event does not have a comment. Otherwise, there is a comment associated to the event.

- **optStrComment**
  The text of the comment for the event saved in the database. If omitted, a standard dialog prompts the user to type a comment.

- **optCustom1 ... optCustom10**
  Custom fields that will be saved in Event History. Any type of data may be specified.
  The number of custom fields can be set in Project Settings: Options.

**Returned value**

This function returns the following possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Function executed successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Event Logger is disabled in the Event Settings dialog.</td>
</tr>
<tr>
<td>2</td>
<td>Event Logger is enabled, but Custom Messages are disabled in the Event Settings dialog.</td>
</tr>
</tbody>
</table>

**Notes**

This function is synchronous. Therefore, the execution of the function finishes only after the event data (including the comment, if any) is saved in the database file. It is recommended that you do not configure this function in background tasks (e.g., Math and Scheduler), unless you do not plan to use the comment or configure it directly (type from the dialog) in the function.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SendEvent( &quot;Valve Open&quot; )</td>
<td>// Saves the event message.</td>
</tr>
<tr>
<td>Tag Name</td>
<td>Expression</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td><code>SendEvent( &quot;Valve Open Oven No. &quot; + OvenID )</code> // Saves the event message concatenated with the value of the OvenID tag.</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td><code>SendEvent( &quot;Valve Open&quot; , 1 )</code> // Displays the dialog where the operator can type comments.</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td><code>SendEvent( &quot;Valve Open&quot; , 1 , TagComment)</code> // Saves the event message with the comment configured in the TagComment tag.</td>
<td></td>
</tr>
</tbody>
</table>
FTP functions

These functions are used to configure the FTP settings for the project, as well as to get files from and put files on a remote server.

**CnfFTP**

This function configures the FTP settings used by other features in the project that can transfer files, such as the FTPGet and FTPPut functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CnfFTP</td>
<td>FTP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

CnfFTP( strServer, optStrUser, optStrPwn, optNumPassiveMode, optNumPort )

**strServer**

The address of the FTP server.

**optStrUser**

The username for the FTP account.

This parameter is optional; if no value is given, then the username "anonymous" is used by default.

**optStrPwd**

The password for the FTP account.

This parameter is optional; if no value is given, then the password is left blank.

**optNumPassiveMode**

A numeric flag that specifies whether Passive FTP mode is enabled. (Passive FTP can be used to bypass some firewall configurations.)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Passive FTP mode is disabled (default).</td>
</tr>
<tr>
<td>1</td>
<td>Passive FTP mode is enabled.</td>
</tr>
</tbody>
</table>

This parameter is optional; if no value is given, then default value is 0.

**optNumPort**

The TCP/IP port number.

This parameter is optional; if no value is given, then port 21 is used by default.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid server name</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid user name</td>
</tr>
</tbody>
</table>
### Notes

This function does not actually connect to the server, so these error codes do not show the quality of the connection. They only show whether the FTP settings have been successfully configured.

You must call this function at least once to configure these settings before you can use the `FTPGet` and `FTPPut` functions to transfer files.

#### Example

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>CnfFTP( &quot;ftp.mycompany.com&quot;, &quot;admin&quot;, &quot;12345&quot;, 1 )</code> // Configures the FTP server using passive mode.</td>
</tr>
</tbody>
</table>

### FTPGet

Gets a file from a remote server and saves it on the local machine.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTPGet</td>
<td>FTP</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

#### Syntax

```
FTPGet( strRemoteFile, strLocalFile, numTransferType, numOverWrite )
```

**strRemoteFile**

The full path and name of the desired file on the remote server, using the syntax "/file path/file name.extension". Some FTP servers are case sensitive, so you should always use correct capitalization.

**strLocalFile**

The full path and name where you want to save the file on the local machine, using the syntax "C:\file name.extension".

**numTransferType**

A numeric flag that specifies the file transfer type. This parameter is optional; if no value is given, then the transfer type is unknown (0) by default.

<table>
<thead>
<tr>
<th>Value</th>
<th>Transfer Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unknown</td>
</tr>
<tr>
<td>1</td>
<td>ASCII</td>
</tr>
<tr>
<td>2</td>
<td>Binary</td>
</tr>
<tr>
<td>10</td>
<td>Unknown, without caching</td>
</tr>
<tr>
<td>11</td>
<td>ASCII, without caching</td>
</tr>
<tr>
<td>12</td>
<td>Binary, without caching</td>
</tr>
</tbody>
</table>

**Note**: In most cases, you should use option 10. This automatically detects the format (ASCII or Binary) of the remote file and sets the transfer type accordingly, and it also forces the project to download the file from the actual FTP server rather than from an intervening proxy or cache server.

**numOverWrite**
A numeric value that specifies whether the local file (specified by `strLocalFile`) may be overwritten if it already exists. This parameter is optional; the default value is 0.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not overwrite — return an error if the file already exists.</td>
</tr>
<tr>
<td>1</td>
<td>Overwrite.</td>
</tr>
</tbody>
</table>

**Returned value**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Failed to open FTP connection</td>
</tr>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>-2</td>
<td>Unknown system error</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid remote file</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid local file</td>
</tr>
<tr>
<td>-5</td>
<td>Invalid transfer type</td>
</tr>
</tbody>
</table>

**Notes**

Before you can call this function, you must configure the FTP settings (i.e., server address and login) using the `CnfFTP` function.

Also, this function is executed asynchronously, so you must call the `FTPStatus` function to see if the transfer has been completed successfully.

**Example**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>FTPGet( &quot;\Reports\040303.txt&quot;, &quot;C:\Report.txt&quot; ) // Retrieves the file 040303.txt in the Reports directory, from the FTP server that was previously specified by CnfFTP(), It then saves the file locally at C:\Report.txt.</td>
</tr>
</tbody>
</table>

**FTPPut**

Puts a file on a remote server.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTPPut</td>
<td>FTP</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

`FTPPut( strLocalFile, strRemoteFile, numTransferType )`

**strLocalFile**

Full qualified name of the local file (e.g., "C:\file.extension")

**strRemoteFile**

Full qualified name of the remote file (e.g., "/Folder/File.extension"). Note that some FTP servers are case sensitive, so you have to enter the name of the file with the correct capitalization.

**numTransferType**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unknown</td>
</tr>
<tr>
<td>1</td>
<td>ASCII</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

| 2 | Binary |

Default is 0.

### Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Failed to create FTP thread</td>
</tr>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid number of parameters</td>
</tr>
<tr>
<td>-2</td>
<td>Unknown system error</td>
</tr>
<tr>
<td>-3</td>
<td>Invalid remote file</td>
</tr>
<tr>
<td>-4</td>
<td>Invalid local file</td>
</tr>
<tr>
<td>-5</td>
<td>Invalid transfer type</td>
</tr>
</tbody>
</table>

### Notes

Before you can call this function, you must configure the FTP settings (i.e., server address and login) using the `CnfFTP` function.

Also, this function is executed asynchronously, so you must call the `FTPStatus` function to see if the transfer has been completed successfully.

### Example

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>FTPPut(&quot;C:\Report.txt&quot;, &quot;\Reports\040303.txt&quot;) // Retrieves the file C:\Report.txt with the name 040303.txt in the folder Reports in the FTP Server.</code></td>
</tr>
</tbody>
</table>

### FTPStatus

This function returns the current status of a file transfer started with `FTPGet` or `FTPPut`.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTPStatus</td>
<td>FTP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

### Syntax

`FTPStatus("strStatusTag")`

`strStatusTag`  
Name of the string tag that will receive the current status description when the function returns.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

### Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Transaction executed successfully</td>
</tr>
<tr>
<td>2</td>
<td>Resolving name</td>
</tr>
<tr>
<td>3</td>
<td>Name resolved</td>
</tr>
<tr>
<td>4</td>
<td>Connecting to server</td>
</tr>
<tr>
<td>5</td>
<td>Connected to server</td>
</tr>
<tr>
<td>6</td>
<td>Closing connection</td>
</tr>
<tr>
<td>Tag</td>
<td>Connection closed</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
</tr>
<tr>
<td>8</td>
<td>Sending request</td>
</tr>
<tr>
<td>9</td>
<td>Request sent</td>
</tr>
<tr>
<td>10</td>
<td>Receiving response</td>
</tr>
<tr>
<td>11</td>
<td>Intermediate response received</td>
</tr>
<tr>
<td>12</td>
<td>Response received</td>
</tr>
<tr>
<td>13</td>
<td>Request completed</td>
</tr>
<tr>
<td>0</td>
<td>No transaction is being executed</td>
</tr>
<tr>
<td>-2</td>
<td>Invalid opttagErrorDescription</td>
</tr>
<tr>
<td>-6</td>
<td>Error opening connection (see status string for details)</td>
</tr>
<tr>
<td>-7</td>
<td>Error establishing connection (see status string for details)</td>
</tr>
<tr>
<td>-8</td>
<td>Error receiving the file (see status string for details)</td>
</tr>
<tr>
<td>-9</td>
<td>Transfer pending</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>FTPStatus( &quot;StatusDescription&quot; ) // Retrieves the status of a current transfer. The return code is stored in the StatusCode tag and the description in the StatusDescription tag.</td>
</tr>
</tbody>
</table>
Database/ERP functions

These functions are used interact with external databases and ERP systems using SQL-like commands.

**DBCursorClose**

Closes the cursor and releases the result set.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCursorClose</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```
DBCursorClose( numCur, "optStrErrorTag" )
```

**numCur**

The cursor handle of the result set. The cursor handle is returned by DBCursorOpen or DBCursorOpenSQL.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

In case of error, returns a negative number. Otherwise, returns 0.

**Notes**

When the cursor is closed, it is destroyed and cannot be used again. You must create a new cursor using DBCursorOpen or DBCursorOpenSQL.

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nErrorCode</td>
<td>DBCursorClose( nCursor )</td>
</tr>
</tbody>
</table>

**DBCursorColumnCount**

Gets the total number of columns in a SQL result set.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCursorColumn</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>
Appendix: Built-in Scripting Language

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```
DBCursorColumnCount( numCur, "optStrErrorTag" )
```

- **numCur**
  The cursor handle of the result set. The cursor handle is returned by `DBCursorOpen` or `DBCursorOpenSQL`.

- **optStrErrorTag**
  The name of a String tag that will receive detailed error messages, if errors occur during runtime.

  **Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

Returns the number of columns. In case of error, returns a negative number.

**Notes**

See also `DBCursorRowCount`.

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see [Database Interface](#) on page 605.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nErrorCode</td>
<td><code>DBCursorColumnCount( nCursor )</code></td>
</tr>
</tbody>
</table>

**DBCursorColumnInfo**

Gets information about a column in a SQL result set. The column is specified by number rather than by name, so this function can be used to retrieve unknown column names.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCursorColumnInfo</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```
DBCursorColumnInfo( numCur, numColumn, numTypeInfo, "optStrErrorTag" )
```

- **numCur**
The cursor handle of the result set. The cursor handle is returned by `DBCursorOpen` or `DBCursorOpenSQL`.

**numColumn**

The number of the column about which you want to get information. Remember that a result set may include only some of the columns in the original database table.

**numTypeInfo**

The type of information you want to get about the column:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Column name</td>
</tr>
<tr>
<td>1</td>
<td>Column data type</td>
</tr>
</tbody>
</table>

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

In case of error, returns a negative number. Otherwise, returns 0.

**Notes**

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nErrorCode</td>
<td><code>DBCursorColumnInfo( nCursor, 2, 0 )</code></td>
</tr>
</tbody>
</table>

// Gets the column name of the second column in the result set.

**DBCursorCurrentRow**

Returns the row number of the current row (i.e., the cursor position) in a SQL result set.

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```
DBCursorCurrentRow( numCur, "optStrErrorTag" )

numCur
```
The cursor handle of the result set. The cursor handle is returned by `DBCursorOpen` or `DBCursorOpenSQL`.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

> **Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

Returns the number of the current row. In case of error, returns a negative number.

**Notes**

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRow</td>
<td><code>DBCursorCurrentRow( nCursor )</code></td>
</tr>
</tbody>
</table>

**DBCursorGetValue**

Gets the value of the specified column of the current row (i.e., the cursor position) in a SQL result set.

**Syntax**

```
DBCursorGetValue( numCur, strColumn, "optStrErrorTag" )
```

**numCur**

The cursor handle for the result set. The cursor handle is returned by `DBCursorOpen` or `DBCursorOpenSQL`.

**strColumn**

The name of the column.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

> **Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.
Returned value

Returns the value of the specified column. If the value is NULL or the cursor is invalid, then it returns an empty string with quality BAD.

Notes

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

Examples

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>DBCursorGetValue( nCursor, &quot;Column1&quot; )</td>
</tr>
</tbody>
</table>

**DBCursorMoveTo**

Moves the cursor to the specified row in a SQL result set and copies that row’s values to the mapped tags. If the specified row doesn’t exist — that is, if it’s outside the range of the result set — then the function returns an error code and doesn’t change the mapped tags.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCursorMoveTo</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

Syntax

`DBCursorMoveTo( numCur, numRow, "optStrErrorTag" )`

**numCur**

The cursor handle of the result set. The cursor handle is returned by DBCursorOpen or DBCursorOpenSQL.

**numRow**

The row of the result set to which the cursor will be moved.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

Returned value

In case of error, returns a negative number. Otherwise, returns 0.

Notes

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.
Examples

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nErrorCode</td>
<td>DBCursorMoveTo( nCursor, 4 ) // Moves the cursor to the fourth row of the result set and copies those values.</td>
</tr>
</tbody>
</table>

**DBCursorNext**

Moves the cursor to the next row in a SQL result set and copies that row's values to the mapped tags. If there is no next row — that is, if the current row is the last — then the function returns an error code and doesn't change the mapped tags.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCursorNext</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```plaintext
DBCursorNext( numCur, "optStrErrorTag" )
```

**numCur**

The cursor handle of the result set. The cursor handle is returned by DBCursorOpen or DBCursorOpenSQL.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

In case of error, returns a negative number. Otherwise, returns 0.

**Notes**

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nErrorCode</td>
<td>DBCursorNext( nCursor )</td>
</tr>
</tbody>
</table>
**DBCursorOpen**

Selects a set of rows and columns in a database table, initializes the cursor at the first row of the result set, copies that row’s values to mapped tags, and then returns a cursor handle that can be referenced by other DB/ERP functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCursorOpen</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```
DBCursorOpen( strDBConn, strTable, optStrCondition, optStrColumns, optStrTags, optStrOrder, "optStrErrorTag" )
```

- **strDBConn**
  The name of the database connection. Connections are configured in the Database/ERP folder.

- **strTable**
  The name of the table in the database.

- **optStrCondition**
  A string specifying which rows of the table to select. This is equivalent to the SQL WHERE clause, and the string should follow the same syntax.
  
  This is an optional parameter. If no rows are specified, then all rows of the table will be selected.

- **optStrColumns**
  A string specifying which columns of the table to select. This list of column names should be comma-delimited.
  
  This is an optional parameter. If no columns are specified, then all columns of the table will be selected.

- **optStrTags**
  A string specifying the project tags to which the columns will be mapped. This list of tag names should be comma-delimited and in the same order as the columns specified by `optStrColumns`. As the cursor is moved through the result set, the values in the current row are copied to these tags.
  
  This is an optional parameter. If no tags are specified, then no values will be copied.

- **optStrOrder**
  The order in which the rows will be sorted. This is equivalent to the SQL ORDER BY clause, and the string should follow the same syntax.
  
  This is an optional parameter. If no order is specified, then the rows will be left in the default order of the table.

- **optStrErrorTag**
  The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.
This is an optional parameter.

**Returned value**

Returns a numeric value that represents the cursor handle. In case of error, returns a negative number.

**Notes**

This function is equivalent to a SQL SELECT statement, except that it breaks the clauses of the statement into separate function parameters. If you know SQL and want to construct your own SELECT statement from scratch, you may use `DBCursorOpenSQL` instead.

See also `DBCursorClose`.

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCursor</td>
<td><code>DBCursorOpen(&quot;DB1&quot;, &quot;Table1&quot;, &quot;Column1 &gt; 3&quot;, &quot;Column1, Column2&quot;, &quot;Tag1, Tag2&quot;, &quot;Column1, Column2 DESC&quot;, &quot;TagError&quot;)</code> // Opens Table1 of DB1 and selects all rows where Column1 has a value greater than 3. Column1 is mapped to Tag1, and Column2 is mapped to Tag2. Rows are ordered first by Column1, then by Column2, in descending order. Error messages are written to TagError.</td>
</tr>
</tbody>
</table>

**DBCursorOpenSQL**

Selects a set of rows and columns in a database table, initializes the cursor at the first row of the result set, copies that row's values to mapped tags, and then returns a cursor handle that can be referenced by other DB/ERP functions. (This function is equivalent to a SQL SELECT statement.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DBCursorOpenSQL</code></td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

`DBCursorOpenSQL( strDBConn, strSQL, optStrTags, "optStrErrorTag" )`

**strDBConn**

The name of the database connection. Connections are configured in the Database/ERP folder.

**strSQL**

A string that gives a complete, syntactically correct SQL SELECT statement.

**Note:** In SQL, curly brackets ({}) are typically used to enclose an expression that must be evaluated before the entire SQL statement is executed. For example:

```sql
SELECT * INTO inmates FROM OPENROWSET
    ('MSDASQL', 'Driver={Microsoft Text Driver (*.txt; *.csv)};DEFAULTDIR=C:\;Extensions=CSV;', 'SELECT * FROM flat.csv')
```

In IWS, however, curly brackets are used to reference tags in text fields that are not normally evaluated (e.g., in the caption of a Button object). If you pass a
SQL statement that includes such an expression to DBCursorOpenSQL, then the project will try to evaluate the expression as a tag reference and the function will fail.

To pass the SQL statement so that the project can correctly evaluate the expression, create a new String tag that contains the text of the expression and then reference the tag in the SQL statement. For example:

```plaintext
$AuxTag = "{Microsoft Text Driver (*.txt; *.csv)}"

$DBCursorOpenSQL( "inmates", "SELECT * INTO inmates FROM OPENROWSET ('MSDASQL', 'Driver=' + AuxTag + ';DEFAULTDIR=C:\;Extensions=CSV;', 'SELECT * FROM flat.csv')")
```

**optStrTags**

A string that lists the project tags to which the columns will be mapped. This list of tag names should be comma-delimited and in the same order as the columns specified by the WHERE clause of strSQL. As the cursor is moved through the result set, the values in the current row are copied to these tags.

This is an optional parameter. If no tags are specified, then no values will be copied.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

Returns a numeric value that represents the cursor handle. In case of error, returns a negative number.

**Notes**

See also DBCursorClose.

You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nCursor</td>
<td>DBCursorOpenSQL( &quot;DB1&quot;, &quot;SELECT Column1, Column2 FROM Table1 WHERE Column1 &gt; 3 ORDER BY Column1, Column2 DESC&quot;, &quot;Tag1, Tag2&quot;)</td>
</tr>
</tbody>
</table>

**DBCursorPrevious**

Moves the cursor to the previous row of the result set and copies that row's values to the mapped tags. If there is no previous row — that is, if the current row is the first — then the function returns an error code and doesn't change the mapped tags.
**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```
DBCursorPrevious( numCur, "optStrErrorTag" )
```

**numCur**
The cursor handle of the result set. The cursor handle is returned by `DBCursorOpen` or `DBCursorOpenSQL`.

**optStrErrorTag**
The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**
In case of error, returns a negative number. Otherwise, returns 0.

**Notes**
You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

**Examples**
As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nErrorCode</td>
<td>DBCursorPrevious( nCursor )</td>
</tr>
</tbody>
</table>

**DBCursorRowCount**
Gets the total number of rows in a SQL result set.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCursorRowCount</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Description**
Gets the total number of rows in a SQL result set.

See also `DBCursorColumnCount()`.

**Syntax**

```
DBCursorRowCount( numCur, "optStrErrorTag" )
```
numCur
The cursor handle of the result set. The cursor handle is returned by DBCursorOpen or DBCursorOpenSQL.

optStrErrorTag
The name of a String tag that will receive detailed error messages, if errors occur during runtime.

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

Returned value
Returns the number of rows. In case of error, returns a negative number.

Notes
See also DBCursorColumnCount.
You can use the Studio Database Gateway to directly monitor database connections for leaks and errors. For more information, see Database Interface on page 605.

Examples
As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRowCount</td>
<td>DBCursorRowCount( nCursor )</td>
</tr>
</tbody>
</table>

DBDelete
Deletes selected rows from a database table. (This function is equivalent to a SQL DELETE statement.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBDelete</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

Note: This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

Syntax

DBDelete( strDBConn, strTable, strCondition, "optStrErrorTag" )

strDBConn
The name of the database connection. Connections are configured in the Database/ERP folder.

strTable
The name of the table in the database.

strCondition
A string that specifies which rows of the table to select. This is equivalent to the SQL WHERE clause, and the string should follow the same syntax.
**Tip:** To delete all rows in the table, make the condition statement a single space (" ").

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

Returns the number of rows deleted from the table. In case of error, returns a negative number.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRowsDeleted</td>
<td>DBDelete( &quot;DB1&quot;, &quot;Table1&quot;, &quot;Column1 &gt; 1000&quot;, &quot;TagError&quot; ) // Deletes all rows in Table1 where the value of Column1 is greater than 1000. The returned value (i.e., the number of rows deleted) is written to TagError.</td>
</tr>
<tr>
<td>Tag</td>
<td>DBDelete( &quot;DB1&quot;, &quot;Table1&quot;, &quot; &quot; ) // Deletes all rows of Table1.</td>
</tr>
</tbody>
</table>

**DBExecute**

DBExecute is a built-in scripting function that executes a custom SQL statement on an external database. If the statement is a query (e.g., SELECT), then the database values are copied to specified array tags.

**Syntax**

DBExecute({ strDBConn, strSQL | optStrTags, optNumMaxRows | optStrErrorTag })

**strDBConn**

The name of the database connection. Connections are configured in the Database/ERP folder in the Project Explorer.

**strSQL**

A complete, syntactically correct SQL statement.

**Note:** In SQL, curly brackets ({}), are typically used to enclose an expression that must be evaluated before the entire SQL statement is executed. For example:

```
SELECT * INTO inmates FROM OPENROWSET ('MSDASQL','Driver={Microsoft Text Driver (*.txt; *.csv)};DEFAULTDIR=C:\;Extensions=CSV;','SELECT * FROM flat.csv')
```

In IWS, however, curly brackets are used to reference project tags in text fields that are not normally evaluated (e.g., in the caption of a Button object). If you pass a SQL statement that includes such an expression to DBExecute, then the project will try to evaluate the expression as a tag reference and the function will fail.
To pass the SQL statement so that the project can correctly evaluate the expression, create a new string tag that contains the text of the expression and then reference that tag in the SQL statement. For example:

```plaintext
AuxTag = "{(Microsoft Text Driver (*.txt; *.csv))}
```

```plaintext
DBExecute( "inmates", "SELECT * INTO inmates FROM OPENROWSET ('MSDASQL','Driver={AuxTag};DEFAULTDIR=C:\;Extensions=CSV;','SELECT * FROM flat.csv')"
```

**optStrTags**

A comma-delimited list of the names of array tags in your project, to which the columns of a SQL SELECT result set will be mapped. The database values will be copied to these array tags, with the first row of the result set being copied to array index 0. Make sure the arrays are large enough to receive all of the rows in the result set.

This parameter is required only when strSQL contains a SQL SELECT statement. For all other types of statements, this parameter is ignored and can be omitted. However, if you need to maintain the syntax of the function in order to continue through to optStrErrorTag, then give this parameter an empty string (""").

**optNumMaxRows**

The maximum number of rows to be copied from a SQL SELECT result set. In most cases, to copy all of the rows, specify a number greater than the expected number of rows in the result set.

This parameter is required only when strSQL contains a SQL SELECT statement. For all other types of statements, this parameter is ignored and can be omitted. However, if you need to maintain the syntax of the function in order to continue through to optStrErrorTag, then give this parameter a value of 0.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

> **Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

This function returns the total number of rows affected by the SQL statement. If an error occurs, then it returns a negative number.

Please note this is the value returned by the function itself; in the case of a SQL SELECT statement, the database values are copied to the array tags specified by optStrTags.

**Notes**

This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL statements are formed and executed before you use this feature.

**Examples**

```plaintext
DBExecute( "DB1", "INSERT INTO Table1(Column1,Column2) values(1,1)"
```

```plaintext
DBExecute( "DB1", "SELECT max(Column1),max(Column2) FROM Table1", "Tag1,Tag2", 1, "TagError"
```


**DBInsert**

Inserts one new row into a database table. (This function is equivalent to a SQL INSERT statement.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBInsert</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```plaintext
DBInsert( strDBConn, strTable, strValues, optStrColumns, "optStrErrorTag" )
```

**strDBConn**

The name of the database connection. Connections are configured in the **Database/ERP** folder in the **Project Explorer**.

**strTable**

The name of the table in the database.

**strValues**

A string that lists the values to be written in the new row. This list of values should be comma-delimited, and string values must be enclosed in single quotes.

**optStrColumns**

A string that lists the columns into which the values will be written. This list of column names should be comma-delimited and in the same order as the values specified by strValues.

This is an optional parameter. If no columns are specified, then the values will be written in the default column order of the database table.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

In case of error, returns a negative number. Otherwise, returns 1.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>nErrorCode</td>
<td>DBInsert( &quot;DB1&quot;, &quot;Table1&quot;, &quot;1, 'one'&quot;, &quot;Column1, Column2&quot; )</td>
</tr>
</tbody>
</table>
**DBSelect**

DBSelect is a built-in scripting function that selects a result set from an external database (equivalent to a SQL SELECT statement), maps the columns to array tags in your project, and copies the values from the result set to the array tags.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSelect</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Syntax**

```
DBSelect(strDBConn, strTable, strTags, strColumns, strCondition, strOrder| optNumMaxRows| optStrErrorTag })
```

**strDBConn**

The name of the database connection. Connections are configured in the Database/ERP folder in the Project Explorer.

**strTable**

The name of the database table from which you want to select.

**strTags**

A comma-delimited list of the names of array tags in your project, to which the columns of the database table will be mapped. The database values will be copied to these array tags, with the first row of the result set being copied to array index 0. Make sure the arrays are large enough to receive all of the rows in the result set.

**strColumns**

A comma-delimited list of which columns in the database table to select. The list order should correspond to the list in strTags.

To select all of the columns in the table, in their original order, give this parameter an empty string ('"').

**strCondition**

A statement specifying which rows in the database table to select. This is equivalent to the SQL WHERE clause and must follow the same syntax.

To select all of the rows in the table, give this parameter an empty string ('"').

**strOrder**

A statement specifying the order in which the rows should be sorted. This is equivalent to the SQL ORDER BY clause and must follow the same syntax.

To leave the rows in their original order, give this parameter an empty string ('"').

**optNumMaxRows**

The maximum number of rows to be copied. In most cases, to copy all of the rows, specify a number greater than the expected number of rows in the result set.

This is an optional parameter; if no value is specified, then only the first row of the result set will be copied.

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

Note: The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.
Returned value
This function returns the total number of rows in the result set. If an error occurs, then it returns a negative number.

Please note this is the value returned by the function itself; the database values are copied to the array tags specified by strTags.

Notes
This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL statements are formed and executed before you use this feature.

Examples

```
DBSelect( "DB1", "Table1", "Array1,Array2", "Column1,Column2", ",", "," )
```

```
DBSelect( "DB1", "Table1", "Array1,Array2", "Column1,Column2", "Column2 < Column1", "Column1", 4, "TagError" )
```

**DBUpdate**

Selects a result set and then writes the same value to all rows of a specified column. (This function is equivalent to a SQL UPDATE statement.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBUpdate</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Note:** This feature emulates SQL (Structured Query Language) database operations. You should be familiar with how SQL commands are formed and executed before you use this feature.

**Syntax**

```
DBUpdate( strDBConn, strTable, strValues, strColumns, optStrCondition, "optStrErrorTag" )
```

**strDBConn**

The name of the database connection. Connections are configured in the **Database/ERP** folder in the **Project Explorer**.

**strTable**

The name of the table in the database.

**strValues**

A string that lists the values to be written to the columns. This list of values should be comma-delimited, and string values must be enclosed in single quotes.

**strColumns**

A string that lists the columns into which the values will be written. This list of column names should be comma-delimited and in the same order as the values specified by **strValues**.

**optStrCondition**

A string that specifies which rows of the table to select. This is equivalent to the SQL WHERE clause, and the string should follow the same syntax.

This is an optional parameter. If no rows are specified, then all rows of the table will be selected.

**optStrErrorTag**
The name of a String tag that will receive detailed error messages, if errors occur during runtime.

**Note:** The tag name should be enclosed in quotes, as shown in the syntax diagram, or else the function will try to use the value of the tag.

This is an optional parameter.

**Returned value**

Returns the number of rows updated. In case of error, returns a negative number.

**Examples**

As used in a Math worksheet:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>DBUpdate( &quot;DB1&quot;, &quot;Table1&quot;, &quot;'X'&quot;, &quot;Column2&quot;, &quot;Column1 = 1&quot;, &quot;TagError&quot; ) // In Table1 of DB1, for all rows where Column1 equals 1, writes &quot;X&quot; to Column2.</td>
</tr>
</tbody>
</table>

**SyncAlarm**

Synchronizes the alarm database.

**Syntax**

```plaintext
SyncAlarm( optStrStartDate, optStrEndDate )
```

**optStrStartDate**

String with the start date. If this parameter is not specified, then the current date is used.

**optStrEndDate**

String with the end date. If this parameter is not specified, then the function uses the same as the start date.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fail to start synchronization; the database is probably being synchronized.</td>
</tr>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid group number</td>
</tr>
<tr>
<td>-2</td>
<td>The format is not set to &quot;Database&quot;.</td>
</tr>
<tr>
<td>-4</td>
<td>Start date specified is invalid.</td>
</tr>
<tr>
<td>-5</td>
<td>End date specified is invalid.</td>
</tr>
<tr>
<td>-6</td>
<td>Start date is greater than the end date.</td>
</tr>
</tbody>
</table>

**Notes**

This function is executed asynchronously, so it doesn’t return the result of the synchronization. To get that information, use the `SyncAlarmStatus` function.
Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>SyncAlarm()</code> // Synchronizes the database using the current date</td>
</tr>
<tr>
<td>Tag</td>
<td><code>SyncAlarm(&quot;10/20/2004&quot;)</code> // Synchronizes the database only for the day 10/20/2004</td>
</tr>
</tbody>
</table>

**SyncAlarmStatus**

Returns the status of a previously called `SyncAlarm` function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyncAlarmStatus</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Syntax**

`SyncAlarmStatus()`

This function takes no parameters.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Synchronization has finished.</td>
</tr>
<tr>
<td>2</td>
<td>Fail synchronizing</td>
</tr>
<tr>
<td>1</td>
<td>Still synchronizing</td>
</tr>
<tr>
<td>0</td>
<td>No synchronization is being executed.</td>
</tr>
<tr>
<td>-1</td>
<td>The format is not set to &quot;Database&quot;.</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>SyncAlarmStatus()</code></td>
</tr>
</tbody>
</table>

**SyncEvent**

Synchronizes the event database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyncEvent</td>
<td>Database/ERP</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Syntax**

`SyncEvent( optStrStartDate, optStrEndDate )`

**optStrStartDate**

String with the start date. If this parameter is not specified, then the current date is used.

**optStrEndDate**

String with the end date. If this parameter is not specified, then the function uses the same as the start date.
## Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fail to start synchronization; the database is probably being synchronized.</td>
</tr>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid group number</td>
</tr>
<tr>
<td>-2</td>
<td>The format is not set to &quot;Database&quot;.</td>
</tr>
<tr>
<td>-4</td>
<td>Start date specified is invalid.</td>
</tr>
<tr>
<td>-5</td>
<td>End date specified is invalid.</td>
</tr>
<tr>
<td>-6</td>
<td>Start date is greater than the end date.</td>
</tr>
</tbody>
</table>

## Notes

This function is executed asynchronously, so it doesn't return the result of the synchronization. To get that information, use the `SyncEventStatus` function.

## Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SyncEvent() // Synchronizes the database using the current date</td>
</tr>
<tr>
<td>Tag</td>
<td>SyncEvent(&quot;10/20/2004&quot;) // Synchronizes the database only for the day 10/20/2004</td>
</tr>
</tbody>
</table>

### SyncEventStatus

Returns the status of a previously called `SyncEvent` function.

#### Syntax

```plaintext
SyncEventStatus()
```

This function takes no parameters.

## Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Synchronization has finished.</td>
</tr>
<tr>
<td>2</td>
<td>Fail synchronizing</td>
</tr>
<tr>
<td>1</td>
<td>Still synchronizing</td>
</tr>
<tr>
<td>0</td>
<td>No synchronization is being executed.</td>
</tr>
<tr>
<td>-1</td>
<td>The format is not set to &quot;Database&quot;.</td>
</tr>
</tbody>
</table>

## Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SyncEventStatus()</td>
</tr>
</tbody>
</table>
**SyncTrend**

Synchronizes the trend database.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyncTrend</td>
<td>Database/ERP</td>
<td>Asynchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>

**Syntax**

```
SyncTrend( numGroup, optStrStartDate, optStrEndDate )
```

**numGroup**

Trend group/worksheet number.

**optStrStartDate**

String with the start date. If this parameter is not specified, then the current date is used.

**optStrEndDate**

String with the end date. If this parameter is not specified, then the function uses the same as the start date.

**Returned value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fail to start synchronization; the database is probably being synchronized.</td>
</tr>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>-1</td>
<td>Invalid group number</td>
</tr>
<tr>
<td>-2</td>
<td>The format is not set to &quot;Database&quot;.</td>
</tr>
<tr>
<td>-4</td>
<td>Start date specified is invalid.</td>
</tr>
<tr>
<td>-5</td>
<td>End date specified is invalid.</td>
</tr>
<tr>
<td>-6</td>
<td>Start date is greater than the end date.</td>
</tr>
</tbody>
</table>

**Notes**

This function is executed asynchronously, so it doesn't return the result of the synchronization. To get that information, use the **SyncTrendStatus** function.

**Examples**

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>SyncTrend(1) // Synchronizes the group 1 database using the current date</td>
</tr>
<tr>
<td>Tag</td>
<td>SyncTrend(1, &quot;10/20/2004&quot;) // Synchronizes the group 1 database only for the day 10/20/2004</td>
</tr>
</tbody>
</table>

**SyncTrendStatus**

Returns the status of a previously called **SyncTrend** function.

<table>
<thead>
<tr>
<th>Function</th>
<th>Group</th>
<th>Execution</th>
<th>Windows</th>
<th>Embedded</th>
<th>Thin Client</th>
<th>Mobile Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyncTrendStatus</td>
<td>Database/ERP</td>
<td>Synchronous</td>
<td>Supported</td>
<td>Supported</td>
<td>Executed on Server</td>
<td>Executed on Server</td>
</tr>
</tbody>
</table>
### Syntax

`SyncTrendStatus(numGroup)`  
**numGroup**  
Trend group/worksheet number.

### Returned value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Synchronization has finished.</td>
</tr>
<tr>
<td>2</td>
<td>Fail synchronizing</td>
</tr>
<tr>
<td>1</td>
<td>Still synchronizing</td>
</tr>
<tr>
<td>0</td>
<td>No synchronization is being executed.</td>
</tr>
<tr>
<td>-1</td>
<td>The format is not set to &quot;Database&quot;.</td>
</tr>
</tbody>
</table>

### Examples

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td><code>SyncTrendStatus( 1 )</code></td>
</tr>
</tbody>
</table>
Overview of VBScript

VBScript is a simple, standard and flexible scripting language that allows you to implement logics and algorithms within your project.

IWS implements Visual Basic Scripting Edition 5.5 or higher. Because IWS hosts VBScript, you can take advantage of every feature provided by this language, such as:

- Syntax, operators and functions.
- The ability to create new variables and procedures (functions and/or sub-routines).
- Access to properties, methods and/or events from COM objects, including ActiveX controls.
- The ability to execute the logics in any platform that supports VBScript, including Microsoft Windows-based PCs (running as the IWS project runtime server), Microsoft Windows Embedded devices (via EmbeddedView or CEView), and Internet Explorer (via the Thin Client).

**Note:** If you are not sure if the image loaded on your device supports VBScript, please consult the hardware manufacturer. The hardware manufacturer must enable the support for VBScript on the Windows Embedded device, so project runtime server will be able to execute the scripts configured in the VBScript language on the device.

The aim of this documentation is to provide an overview about the integration of VBScript with InduSoft Web Studio. Furthermore, it can be used as a quick reference for the most used features of the language. For a full description of the language as well as its interfaces and functions, please consult Microsoft. (At the time of this writing, the VBScript documentation could be accessed directly at the Microsoft Developer Network. This link, however, is beyond our control and may change without notice.)
VBScript Interfaces in the Software

The following table provides a summary of the VBScript interfaces supported by InduSoft Web Studio:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Scope for Procedures and Variables</th>
<th>Execution</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Procedures</td>
<td>Graphics and Tasks</td>
<td>-</td>
<td>Declaration of Procedures</td>
</tr>
<tr>
<td>Graphics Script</td>
<td>Graphics Script interface only</td>
<td>Server (Viewer) + Thin Clients</td>
<td>Declaration of Variables Declaration of Procedures Execution</td>
</tr>
<tr>
<td>Screen Script</td>
<td>Screen where the script is configured</td>
<td>Server (Viewer) + Thin Clients</td>
<td>Declaration of Variables Declaration of Procedures Execution</td>
</tr>
<tr>
<td>Command animation</td>
<td>Object where the script is configured</td>
<td>Server (Viewer) + Thin Clients</td>
<td>Declaration of Variables Execution</td>
</tr>
<tr>
<td>ActiveX Events</td>
<td>Object where the script is configured</td>
<td>Server (Viewer) + Thin Clients</td>
<td>Declaration of Variables Execution</td>
</tr>
<tr>
<td>Startup Script</td>
<td>All Script Sheets from Tasks</td>
<td>Server (BGTask)</td>
<td>Declaration of Variables Declaration of Procedures Execution</td>
</tr>
<tr>
<td>Script Groups</td>
<td>Script Group only</td>
<td>Server (BGTask)</td>
<td>Declaration of Variables Execution</td>
</tr>
</tbody>
</table>
The following illustration shows the scope of each VBScript interface and the order in which they are scanned by IWS:

The illustration shows that the Global Procedures are shared by the Graphic Module and the Background Task. However, the other VBScript interfaces are either from the Graphic Module or from the Background Task, and they do not share variables or procedures between them. They are independent of each other.

Note: Although the Graphics Script is scanned by IWS before the Screen Scripts, the procedures and variables declared in the Graphics Script interface are NOT available for any script interface configured on the screens. You must use the Global Procedures interface to implement procedures that must be available for all screens.

When writing your code in a VBScript interface, you can access any tag from the IWS tags database or any function from the Built-in Scripting Language by applying the "$" prefix to the tag/function name, as in the examples below:

$Time  'Returns the value of the tag Time from the tags database
$MyTag  'Returns the value of the tag MyTag from the tags database
$Open("main")  'Executes the Open() built-in function to open the "main" screen

Therefore, you can create scripts using built-in functions from IWS, tags from the IWS tags database, VBScript functions, VBScript variables, ActiveX properties, methods or events, and any other interface available. The IWS tags are shared by all modules from IWS, including the Graphic Module and the Background Task.

Global Procedures

This Procedures interface is used create a library of VBScript functions and sub-routines that can be called by any other scripting interface in InduSoft Web Studio. The procedures declared here are never directly executed during runtime; they must be explicitly called by another script.

Caution: InduSoft Web Studio will not prevent you from declaring two or more functions with the same name. (This includes functions imported from external files; see "Importing Functions from
an External File” below.) However, if you do, then your project may behave unexpectedly during runtime. Make sure that all of your functions are named correctly.

You can create as many Procedures worksheets as you want, and each worksheet can contain as many functions and sub-routines as you want.

To use the Procedures interface:

1. In the **Global** tab of the Project Explorer, do one of the following:
   - To edit the default Procedures worksheet, open the **Procedures** folder and then double-click **Main Procedures**; or
   - To create a new Procedures worksheet, right-click the **Procedures** folder and then click **Insert** on the shortcut menu.

   Either way, the worksheet is opened for editing.

![Main Procedures worksheet](image)

2. Declare your functions and sub-routines in the interface. For example:

   ```vbscript
   Option Explicit
   'Keep the Option Explicit statement in the first line of this interface.
   'Procedures with global scope can be implemented here
   'Global variables are NOT supported in this interface
   Sub MyMessage(message)
       MsgBox message, 0
   End Sub
   
   Function MyAdd(number1, number2)
       MyAdd = number1 + number2
       Call MyMessage("The sum is" & MyAdd & ".")
   End Function
   
   **Note:** You can declare local variables within each procedure, but you cannot declare global variables in this interface. In most cases, you should use tags instead.
3. Save your changes. The worksheet is added to the Procedures folder in the Project Explorer.

![Project Explorer](image)

**Procedure worksheets with declared functions**

**Organizing procedures into sub-folders**

If you have many procedures in a single Procedures worksheet, then you may choose to organize them into sub-folders. To organize procedures:

1. In the Procedures folder, open the worksheet that you want to organize.
2. In the worksheet, insert the following line before the procedures that you want to group together:

   `'$region:foldernamé`

   ...where foldernamé is the name of the sub-folder. For example:

   `'$region:My Subroutines
   Sub MyMessage(message)
       MsgBox message, 0
   End Sub`

   `'$region:My Functions
   Function MyAdd(number1, number2)
       MyAdd = number1 + number2
       Call MyMessage("The sum is" & MyAdd & ".")
   End Function`

3. Save your changes. The procedures are organized into sub-folders under the Procedures worksheet in the Project Explorer.

**Import functions from an external file**

You can also import functions from an external file and add them to a Procedures worksheet. This is useful if you have a library of existing functions that you want to reuse.

To import functions into a worksheet:

1. Save and close all open screens and worksheets.
2. In the Procedures folder, right-click the Procedures worksheet into which you want to import functions, and then click **Import** on the shortcut menu.
The *Import - Global Procedures* dialog box is displayed.

![Import - Global Procedures dialog box](image)

3. To the right of the **File** box, click **Browse**. A standard Windows file browser is displayed. Use it to locate and select a Procedures worksheet file. (This is a plain text file that has been saved with the `.gis` file extension.)

4. Select **Replace functions if they already exist** to overwrite functions in the worksheet with functions imported from the file, if the functions have the same names.

5. In the **Functions** area, do one of the following:
   - Click **All** to import all functions from the file; or
   - Click **Only** to import only selected functions from the files, and then select those functions in the list.

6. Click **Import**.

After the functions are imported, they should be displayed in the worksheet.

**Password protect a worksheet**

You can put a password on any of your Procedures worksheets to prevent them from being edited or analyzed by other users. To protect a Procedures worksheet:

1. In the **Procedures** folder, right-click the worksheet and then click **Password Protection** on the shortcut menu. A **Password Protection** dialog is displayed.

2. In the **New Password** box, type your password.

3. In the **Confirm Password** box, type your password again.

4. Click **OK**.

Once this is done, you will be prompted for the password whenever you attempt to open the worksheet.
**Graphic Module**

**GRAPHICS SCRIPT**

The Graphics Script interface can be edited by its icon from the Graphics tab of the Project Explorer:

This interface can be used to execute logics on the following events, based on pre-defined sub-routines:

- **Graphics_OnStart()**: The code configured within this sub-routine is automatically executed just once when the graphic module is started. This interface is useful for initializing variables or executing logics that must be implemented when running the project.

- **Graphics_WhileRunning()**: The code configured within this sub-routine is automatically executed continuously while the graphic module is running. The rate in which this sub-routine is called depends on the performance of the platform where the project is running.

- **Graphics_OnEnd()**: The code configured within this sub-routine is automatically executed just once when the graphic module is closed.

- **Graphics_OnScreenResize(width, height)**: The code configured within this sub-routine is automatically executed just once when the screen resolution of the runtime station changes. The new width and height of the screen (in pixels) are passed to the sub-routine as parameters.

  **Note**: The **Graphics_OnScreenResize()** sub-routine is intended for projects running Windows Embedded devices that can switch between Portrait and Landscape display modes. It is not supported for projects running on Windows PC or in the Thin Client.

  **Important**: Do not change the name of the predefined sub-routines. If you do, then the system will not be able to automatically execute them.

Example:

```
' Variables with local scope can be declared and initialized here
Dim MyDate
MyDate = Date()
Dim MyValue
MyValue = 100

' Procedures with local scope can be implemented here
Function MyNewProcedure(nCount)
    MyNewProcedure = nCount + 1
End Function

Function AreaRec(side1, side2)
    AreaRec = side1 * side2
End Function

Sub CheckHiLimit(myValue, myHiLimit)
    If myValue > myHiLimit Then
        MsgBox("Value out of range")
    End If
End Sub
```
'This procedure is executed just once when the graphic module is started
Sub Graphics_OnStart()
    MsgBox("Welcome to the system!")
End Sub

'This procedure is executed continuously while the graphic module is running
Sub Graphics_WhileRunning()
    If $UserName = "Guest" Then
        $MyFlag = 0
    End If
End Sub

'This procedure is executed just once when the graphic module is closed
Sub Graphics_OnEnd()
    $LogOff()
End Sub

When the Sub-routines Are Executed
On the Server (i.e., where InduSoft Web Studio, EmbeddedView, or CEView is installed and running):

- The graphic module is the Viewer task.
- The Graphics_OnStart() sub-routine is executed once on the Server when the Viewer task is launched.
- The Graphics_WhileRunning() sub-routine keeps being executed on the Server while the Viewer task is running. The Graphics_OnEnd() sub-routine is executed once on the Server when the Viewer task is shut down.

On the Thin Client or Secure Viewer:

- The graphic module is the ISSymbol control.
- The Graphics_OnStart() sub-routine is executed once on the Thin Client station after logging in successfully.
- The Graphics_WhileRunning() sub-routine keeps being executed on the Thin Client station while the ISSymbol control is hosted by the Web Browser.
- The Graphics_OnEnd() sub-routine is executed once on the Thin Client station when the Web Browser is shut down (or when the ISSymbol control is no longer hosted by the Web Browser).

The execution of the Graphic Script sub-routines on the Server is completely independent of the execution on the Thin Client and Secure Viewer stations.

Calling Graphics Script Procedures in Other VBScript Interfaces

The three predefined sub-routines are strictly local to the Graphic Script interface and are executed only on the events described above. Other procedures defined in the interface, however — under the Procedures with local scope heading — may be called in any other Screen Script or Command animation. The procedures are called by using the syntax Graphics.procedure_name.

Taking the function MyNewProcedure that was declared in the example above, you could place a Button object on your project screen and then apply a Command animation to it with the following line:


SCREEN SCRIPT

Each project screen has an associated screen script. To edit the screen script for a given project screen, open the screen worksheet for editing and then do one of the following:

- On the Graphics tab of the ribbon, in the Screen group, click Script; or
- Right-click in the screen worksheet, and then click Screen Script on the shortcut menu.

This interface can be used to execute logics on the following events, based on preconfigured sub-routines:

- Screen_IIsClosedByReplace(): This procedure determines whether the screen is automatically closed when another screen is opened to replace it. If the procedure is given a value of 0 or FALSE, then automatic
closing is disabled. When the function is given a positive value (e.g., 1) or **TRUE**, or if the procedure is not declared at all, then automatic closing is enabled.

- **Screen_OnOpen()**: The code configured within this sub-routine is automatically executed just once when the screen is opened.

- **Screen_WhileOpen()**: The code configured within this sub-routine is automatically executed continuously while its screen is open. The rate in which this sub-routine is called depends on the performance of the platform where the project is running.

- **Screen_OnClose()**: The code configured within this sub-routine is automatically executed just once when the screen is closed.

The variables and procedures declared in this interface are available for the VBScript interfaces of the screen where the Screen Script is configured.

**Caution:** Do not change the names of the preconfigured sub-routines described above. If you do, then the system will not be able to call them.

**Note:**
- The execution of the Screen Script sub-routines on the server is totally independent of the execution on the Thin Client stations. In other words, these sub-routines are executed asynchronously.
- The procedures and/or variables declared in the Screen Script interface have local scope. They can be called only from the specific screen on which they are declared.

Example:

```vbnet
'Variables available on this screen can be declared and initialized here
Dim Counter

'Procedures available on this screen can be implemented here
Function AreaCircle(radius)
    AreaCircle = Sqr(radius) * $Pi()
End Function

Sub CheckLoLimit (myValue, myLoLimit)
    If myValue < myLoLimit Then
        MsgBox("Value out of range")
    End If
End Sub

'This procedure determines whether the screen is automatically closed
Function Screen_IsClosedByReplace()
    Screen_IsClosedByReplace = $ReplaceModeTag
End Function

'This procedure is executed just once when this screen is open
Sub Screen_OnOpen()
    MsgBox("The screen was open!")
End Sub

'This procedure is executed continuously while this screen is open
Sub Screen_WhileOpen()
    If Counter < 100 Then
        Counter = Counter + 1
    Else
        Counter = 0
    End If
    $SimulationTag = Counter
End Sub

'This procedure is executed just once when this screen is closed
Sub Screen_OnClose()
    MsgBox("The screen will be closed!")
```
COMMAND ANIMATION

On the Graphics tab, in the Animations group, click Command to add the animation to a selected object or group of objects. The animation enables you to click on the object or press a pre-defined key to execute the command at runtime. Double-click on the object to view its object properties.

Object Properties: Command

The Command animation provides one tag for each one of the events supported by it. Notice that more than one event can be configured simultaneously for the same Command animation:

- **On Down**: Executes the command/script once when the user clicks on the object with the left mouse button.
- **On While**: Keeps executing the command/script continuously while the mouse pointer is pressed on the object. The period (in milliseconds) of execution for the command/script is set in the Rate field from the Configuration dialog screen, except for the VBScript option, which is executed as fast as possible.
- **On Up**: Executes the command/script once when the user releases the left mouse button on the object.
- **On Right Down**: Executes the command/script once when the user clicks on the object with the right mouse button.
- **On Right Up**: Executes the command/script once when the user releases the right mouse button on the object.
- **On Double Click**: Executes the command/script once when the user double-clicks on the object with the left mouse button.
- **On Touch, On Touch Start, On Touch Delta, On Touch Complete**: These events are used for multi-touch gestures. For more information, see About Touch Events on page 269.

**Note**: The events On Right Down, On Right Up and On Double Click are not supported in projects running on Windows Embedded target systems.

**Tip**: An asterisk (*) on an event tab indicates that something is configured for that event. This makes it easier to see at a glance which events are configured.

**Type menu**: This setting defines the type of action that must be executed by the event of the Command animation. Notice that each event has its own type. Therefore, the same Command animation can be configured with different types of action for different events. The following types are supported:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in Language</td>
<td>Allows you to configure a script using the IWS built-in language. When this type is selected, the user can configure up to 12 expressions for each event in the Expression column. The expressions are executed sequentially from the first row until the last</td>
</tr>
</tbody>
</table>
### Type

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>One when the event is triggered. The result of each expression is written to the tag configured in the <strong>Tag</strong> column (if any). Consult the <em>Built-in Scripting Language</em> chapter for more information.</td>
</tr>
</tbody>
</table>

**VBScript**

- Allows you to configure a script using the standard VBScript language. When this type is selected, the user can configure a script in the VBScript editor for the **Command** animation. Consult the *VBScript* chapter for further information about the VBScript language.

**Open Screen**

- Allows you to configure the **Command** animation to open a specific screen when the event is triggered during runtime. This type is equivalent to the **Open** function. You can either type the screen name in the **Open Screen** field or browse it. Furthermore, you can type a string tag between curly brackets {{TagName}} in this field. When the event is executed, the project will attempt to open the named screen.

**Close Screen**

- Allows you to configure the **Command** animation to close a specific screen when the event is triggered during runtime. This type is equivalent to the **Close** function. You can either type the screen name in the **Close Screen** field or browse it. You can also type a string tag between curly brackets {{TagName}} in this field. When the event is executed, the project will attempt to close the named screen.

**Set Tag**

- Allows you to configure the **Command** animation to set a tag when the event is triggered during runtime. You can either type the tag name in the **Set Tag** field or browse it. When the event is executed, the project will write the value 1 to the tag configured in this field.

**Reset Tag**

- Allows you to configure the **Command** animation to reset a tag when the event is triggered during runtime. You can either type the tag name in the **Reset Tag** field or browse it. When the event is executed, the project will write the value 0 to the tag configured in this field.

**Toggle Tag**

- Allows you to configure the **Command** animation to toggle a tag when the event is triggered during runtime. You can either type the tag name in the **Toggle Tag** field or browse it. When the event is executed, the project will toggle the value of the tag configured in this field.

---

**Config** button: Launches the *Configuration* dialog, where the Command animation can be fully configured.

**Back to button**: Click to go back to the object properties of the underlying Button object.

**Configuration dialog**

This dialog allows you to fully configure the Command animation...

![Configuration dialog](image)

The event tabs (e.g., On Down, On While, etc.) and the **Type** menu are the same as in the *Object Properties* dialog described above. The remaining settings are shared for all events:
Options pane:

• **Enable Focus** checkbox: When this option is checked, the object that the Command animation was applied to can receive the focus during runtime by the navigation keys.

• **Force** checkbox: When this option is selected, any project tag that receives a value will trigger events as if the tag changed, even if the new value is equal to the old value. For example, if a tag has a value of 0 and the Command animation runs a procedure that writes 0 to that tag, all other tasks in the project runtime will recognize that the tag changed, even though it did not. This option is useful for making sure that events triggered by tag changes (e.g., Write on Tag Change on a communication driver) are always triggered when the Command animation is used.

Please keep in mind that if the tag's value does not actually change, the tag's timestamp (tagname->Timestamp) is not updated either.

**Force** applies to both the procedure run by the Command animation itself and any global procedures called in that procedure, as long as they are run on the project runtime client where the Command animation is used (i.e., on the device where the button is pushed).

**Force** does not apply to global procedures that are run on the project runtime server using the function RunGlobalProcedureOnServer, even if the function is called in the procedure run by the Command animation. If you want to force tag changes in global procedures run on the server, use the function ForceTagChange.

**Note:** For projects created with InduSoft Web Studio v6.1+SP3 or earlier, **Force** is enabled by default and cannot be disabled.

• **Beep** checkbox: When this option is checked, a short beep is played when the Command is executed. This option is useful to provide an audio feedback to the user, indicating that the Command was executed. It does not indicate, however, if the action triggered by the Command animation was successful or not.

• **Release** checkbox: When this option is checked, the On Up event is executed when you drag the cursor (or your finger) out of the object area (whether the button was released or not). This option is useful to make sure that the On Up event will always be executed after an On Down event, even if the user releases the mouse cursor out of the object area before releasing it.

• **Confirm** checkbox: When this option is checked, user will have to answer a confirmation question before executing the command. This option is useful for decreasing the accidental triggering of critical events during runtime.

• **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.

• **Key Only** checkbox: When this option is checked, the user can only use the keyboard shortcut (configured in the Key pane described below) to execute commands.

• **Disable**: Disables action by the user when the result of the expression configured in this field is TRUE (value different from 0).

• **Security**: Security access level required to use the Command animation.

• **Key** group: Shortcut used to trigger the events On Down, While Down and On Up using a keyboard. (In other words, pressing this keyboard shortcut is the same as clicking the left mouse button.) This option is especially useful when creating projects for runtime devices that do not provide a mouse or touch-screen interface — the keyboard is the only physical interface available to interact with your project during runtime.

• **Shift, Ctrl, or Alt** boxes: Click to create a key combination key, meaning the Shift, Ctrl and/or Alt key must be pressed with the key specified in the drop-down list.

• Click the browse button (…) to open the Key Modifier dialog, which enables you to modify your combination keys. You can choose Left, Right or Left or Right to specify the position on the keyboard of the Shift, Ctrl or Alt key in the key combination. If you choose Left or Right, the command will be executed any time either of these keys is pressed in combination with the key specified in the drop-down list.

**Tip:** If you have defined custom keys for your project, you can select them in this list. For more information, see Define custom keys for selected screen objects.
ACTIVEX EVENTS
To edit the ActiveX Events interface, select the Script option from the Events tab of the ActiveX object inserted on the screen.

1. Click the ActiveX Control icon in the Active Objects toolbar.

The *Insert ActiveX Control* dialog opens.

2. Select the ActiveX Control that you wish to use and then click **OK**.

3. The object that symbolizes the selected ActiveX Control will display. Right-click on this object to open the *Object Properties* dialog.

4. Click the **Configuration** button. The *Configuration* dialog will open. Click the Events tab.
5. Click the ... button in the Script column.

Use this interface to execute logics when an ActiveX object triggers an event.

Variables declared in this interface are available for this interface only (local scope). In other words, they are not available for any other object in the project.

You cannot implement procedures in this interface. You can, however, call procedures implemented in the Global Procedures or in the Screen Script interface for the same screen where the ActiveX object is configured.

**Note:** For more information, see ActiveX Control object.

Example:

```
'The script below will be executed when the Calendar Control ActiveX
'triggers its "AfterUpdate" event
$MyYear = CalendarControl1.Year
$MyMonth = CalendarControl1.Month
$MyDay = CalendarControl1.Day
```

**Background Task**

**SCRIPT WORKSHEET**

A Script worksheet is used to implement program logic (using VBScript) that should be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.

**Note:** The Script worksheet is functionally similar to the Math worksheet, except that it uses VBScript instead of the Built-in Scripting Language.

To create a new Script worksheet, do one of the following:

- On the Insert tab of the ribbon, in the Task Worksheets group, click **Script**;
- Right-click the **Script** folder in the Project Explorer, and then click **Insert** on the shortcut menu; or
- Click **New** on the Application menu, click the **File** tab, and then select **Script Worksheet**.

To edit an existing Script worksheet, double-click it in the Project Explorer.
The code configured in each Script worksheet is executed by the Background Task. The project scans the worksheets sequentially (based on the worksheet number) and executes only the groups in which the condition configured in the Execution field of the worksheet is TRUE (i.e., non-zero).

**Note:** You must use the syntax supported by the Built-in Scripting Language in the Execution field. Only the body of the worksheet supports VBScript.

Variables declared in the worksheet have local scope for that specific group only. They are not available for any other VBScript interface.

You cannot define procedures (i.e., functions and subs) in the Script worksheet. However, you can call procedures defined in the Global Procedures or in the Startup Script.

Example:

```vbnet
'Variables available only for this group can be declared here
Dim myVar, myTest
myTest = 1

'The code configured here is executed while the condition configured in the Execution field is TRUE
myVar = $FindFile("c:\*.txt")
If MyVar > 0 Then
    $TagNumOfFiles = myVar
End If
```

**Caution:** When any Script worksheet is saved during runtime (on-line configuration), the Startup Script will be executed again and the current value of the local variables of any Script worksheet will be reset.

### STARTUP SCRIPT WORKSHEET

The Startup Script worksheet is a VBScript interface that is automatically executed when the project is run. To edit the Startup Script worksheet, double-click it in the Project Explorer. (It is located on the Tasks tab, in the Script folder.) The worksheet is displayed:

The code configured in this worksheet is executed just once when the Background Task module (BGTask) is started. This interface is useful for initializing variables or executing logics that must be implemented when the project is run.

You can declare and initialize variables and define procedures. However, variables or procedures declared in this interface will be available ONLY to the Script worksheets executed by the Background Task module — they are not available to any VBScript interface from the Graphic Module.

Example:

```vbnet
'Variables available for all Script groups from the Script task can be declared and initialized here
Dim MyVar, Counter
MyVar = 100

'Procedures available for all Script groups from the Script task can be implemented here
```
Function AreaEquTriangle(base, high)
    AreaEquTriangle = (base * high) / 2
End Function

Sub CheckLimits(myValue, myHiLimit, myLoLimit)
    If (myValue > myHiLimit Or myValue < myLoLimit) Then
        MsgBox("Value out of range")
    End If
End Sub

' The code configured here is executed just once when the Background task is started
If $GetOS() = 3 Then
    MsgBox("Welcome! This project is running under Microsoft Windows Embedded operating system.")
Else
    MsgBox("Welcome! This project is running under Microsoft Windows desktop operating system.")
End If
## Language Reference

### Operators

#### Arithmetic Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Exponentiation</td>
<td>Raises a number to the power of an exponent.</td>
</tr>
<tr>
<td>-</td>
<td>Unary negation</td>
<td>Finds the difference between two numbers or indicates the negative value of a numeric expression.</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>Multiplies two numbers.</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>Divides two numbers and returns a floating-point result.</td>
</tr>
<tr>
<td>\</td>
<td>Integer division</td>
<td>Divides two numbers and returns an integer result.</td>
</tr>
<tr>
<td>Mod</td>
<td>Modulus arithmetic</td>
<td>Divides two numbers and returns only the remainder.</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
<td>Finds the sum of two numbers.</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>Finds the difference between two numbers or indicates the negative value of a numeric expression.</td>
</tr>
<tr>
<td>&amp;</td>
<td>String concatenation</td>
<td>Forces string concatenation of two expressions.</td>
</tr>
</tbody>
</table>

#### Comparison Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equality</td>
<td>Comparison is True if the first expression is equal to the second expression.</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Inequality</td>
<td>Comparison is True if the first expression is different from the second expression.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>Comparison is True if the first expression is less than the second expression.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>Comparison is True if the first expression is greater than the second expression.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td>Comparison is True if the first expression is less than or equal to the second expression.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>Comparison is True if the first expression is greater than or equal to the second expression.</td>
</tr>
<tr>
<td>Is</td>
<td>Object equivalence</td>
<td>Compares two object reference variables. Comparison is True if both object names refer to the same object.</td>
</tr>
</tbody>
</table>

#### Logical Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not</td>
<td>Logical negation</td>
<td>Performs logical negation on an expression.</td>
</tr>
<tr>
<td>And</td>
<td>Logical conjunction</td>
<td>Performs a logical conjunction on two expressions.</td>
</tr>
<tr>
<td>Or</td>
<td>Logical disjunction</td>
<td>Performs a logical disjunction on two expressions.</td>
</tr>
<tr>
<td>Xor</td>
<td>Logical exclusion</td>
<td>Performs a logical exclusion on two expressions.</td>
</tr>
<tr>
<td>Eqv</td>
<td>Logical equivalence</td>
<td>Performs a logical equivalence on two expressions.</td>
</tr>
<tr>
<td>Imp</td>
<td>Logical implication</td>
<td>Performs a logical implication on two expressions.</td>
</tr>
</tbody>
</table>

#### Assignment Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Assignment</td>
<td>Assigns a value to a variable or property.</td>
</tr>
</tbody>
</table>
## Constants

### Color Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbBlack</td>
<td>&amp;h00</td>
<td>Black</td>
</tr>
<tr>
<td>vbRed</td>
<td>&amp;hFF</td>
<td>Red</td>
</tr>
<tr>
<td>vbGreen</td>
<td>&amp;hFF00</td>
<td>Green</td>
</tr>
<tr>
<td>vbYellow</td>
<td>&amp;hFFFF</td>
<td>Yellow</td>
</tr>
<tr>
<td>vbBlue</td>
<td>&amp;hFF0000</td>
<td>Blue</td>
</tr>
<tr>
<td>vbMagenta</td>
<td>&amp;hFF00FF</td>
<td>Magenta</td>
</tr>
<tr>
<td>vbCyan</td>
<td>&amp;hFFFF00</td>
<td>Cyan</td>
</tr>
<tr>
<td>vbWhite</td>
<td>&amp;hFFFFFF</td>
<td>White</td>
</tr>
</tbody>
</table>

### Comparison Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbBinaryCompare</td>
<td>0</td>
<td>Perform a binary comparison</td>
</tr>
<tr>
<td>vbTextCompare</td>
<td>1</td>
<td>Perform a textual comparison</td>
</tr>
</tbody>
</table>

### Date & Time Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbSunday</td>
<td>1</td>
<td>Sunday</td>
</tr>
<tr>
<td>vbMonday</td>
<td>2</td>
<td>Monday</td>
</tr>
<tr>
<td>vbTuesday</td>
<td>3</td>
<td>Tuesday</td>
</tr>
<tr>
<td>vbWednesday</td>
<td>4</td>
<td>Wednesday</td>
</tr>
<tr>
<td>vbThursday</td>
<td>5</td>
<td>Thursday</td>
</tr>
<tr>
<td>vbFriday</td>
<td>6</td>
<td>Friday</td>
</tr>
<tr>
<td>vbSaturday</td>
<td>7</td>
<td>Saturday</td>
</tr>
<tr>
<td>vbUseSystemDayOfWeek</td>
<td>0</td>
<td>Use the day of the week specified in your system settings for the first day</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of the week.</td>
</tr>
<tr>
<td>vbFirstJan1</td>
<td>1</td>
<td>Use the week in which January 1 occurs (default).</td>
</tr>
<tr>
<td>vbFirstFourDays</td>
<td>2</td>
<td>Use the first week that has at least four days in the new year.</td>
</tr>
<tr>
<td>vbFirstFullWeek</td>
<td>3</td>
<td>Use the first full week of the year.</td>
</tr>
</tbody>
</table>

### Date Format Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbGeneralDate</td>
<td>0</td>
<td>Display a date and/or time. For real numbers, display a date and time. If</td>
</tr>
<tr>
<td></td>
<td></td>
<td>there is no fractional part, display only a date. If there is no integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>part, display time only. Date and time display is determined by your system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>settings.</td>
</tr>
<tr>
<td>vbLongDate</td>
<td>1</td>
<td>Display a date using the long date format specified in your computer's</td>
</tr>
<tr>
<td></td>
<td></td>
<td>regional settings.</td>
</tr>
<tr>
<td>vbShortDate</td>
<td>2</td>
<td>Display a date using the short date format specified in your computer's</td>
</tr>
<tr>
<td></td>
<td></td>
<td>regional settings.</td>
</tr>
<tr>
<td>vbLongTime</td>
<td>3</td>
<td>Display a time using the long time format specified in your computer's</td>
</tr>
<tr>
<td></td>
<td></td>
<td>regional settings.</td>
</tr>
<tr>
<td>vbShortTime</td>
<td>4</td>
<td>Display a time using the short time format specified in your computer's</td>
</tr>
<tr>
<td></td>
<td></td>
<td>regional settings.</td>
</tr>
</tbody>
</table>
### Miscellaneous Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbObjectError</td>
<td>-2147221504</td>
<td>User-defined error numbers should be greater than this value.</td>
</tr>
</tbody>
</table>

### Box Constants – Buttons & Icons

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbOKOnly</td>
<td>0</td>
<td>Display OK button only.</td>
</tr>
<tr>
<td>vbOKCancel</td>
<td>1</td>
<td>Display OK and Cancel buttons.</td>
</tr>
<tr>
<td>vbAlertRetryIgnore</td>
<td>2</td>
<td>Display Abort, Retry, and Ignore buttons.</td>
</tr>
<tr>
<td>vbYesNoCancel</td>
<td>3</td>
<td>Display Yes, No, and Cancel buttons.</td>
</tr>
<tr>
<td>vbYesNo</td>
<td>4</td>
<td>Display Yes and No buttons.</td>
</tr>
<tr>
<td>vbRetryCancel</td>
<td>5</td>
<td>Display Retry and Cancel buttons.</td>
</tr>
<tr>
<td>vbCritical</td>
<td>16</td>
<td>Display Critical Message icon.</td>
</tr>
<tr>
<td>vbQuestion</td>
<td>32</td>
<td>Display Warning Query icon.</td>
</tr>
<tr>
<td>vbExclamation</td>
<td>48</td>
<td>Display Warning Message icon.</td>
</tr>
<tr>
<td>vbInformation</td>
<td>64</td>
<td>Display Information Message icon.</td>
</tr>
<tr>
<td>vbDefaultButton1</td>
<td>0</td>
<td>First button is the default.</td>
</tr>
<tr>
<td>vbDefaultButton2</td>
<td>256</td>
<td>Second button is the default.</td>
</tr>
<tr>
<td>vbDefaultButton3</td>
<td>512</td>
<td>Third button is the default.</td>
</tr>
<tr>
<td>vbDefaultButton4</td>
<td>768</td>
<td>Fourth button is the default.</td>
</tr>
<tr>
<td>vbApplicationModal</td>
<td>0</td>
<td>Application modal. The user must respond to the message box before continuing work in the current application.</td>
</tr>
<tr>
<td>vbSystemModal</td>
<td>4096</td>
<td>System modal. On Win16 systems, all programs are suspended until the user responds to the message box. On Win32 systems, this constant provides a program modal message box that always remains on top of any other programs you may have running.</td>
</tr>
</tbody>
</table>

### Box Constants – Selected Button

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbOK</td>
<td>1</td>
<td>OK button was clicked.</td>
</tr>
<tr>
<td>vbCancel</td>
<td>2</td>
<td>Cancel button was clicked.</td>
</tr>
<tr>
<td>vbAbort</td>
<td>3</td>
<td>Abort button was clicked.</td>
</tr>
<tr>
<td>vbRetry</td>
<td>4</td>
<td>Retry button was clicked.</td>
</tr>
<tr>
<td>vbIgnore</td>
<td>5</td>
<td>Ignore button was clicked.</td>
</tr>
<tr>
<td>vbYes</td>
<td>6</td>
<td>Yes button was clicked.</td>
</tr>
<tr>
<td>vbNo</td>
<td>7</td>
<td>No button was clicked.</td>
</tr>
</tbody>
</table>

### String Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbCr</td>
<td>Chr(13)</td>
<td>Carriage return</td>
</tr>
<tr>
<td>VbCrLf</td>
<td>Chr(13) &amp; Chr(10)</td>
<td>Carriage return...linefeed combination</td>
</tr>
<tr>
<td>vbFormFeed</td>
<td>Chr(12)</td>
<td>Form feed; not useful in Microsoft Windows</td>
</tr>
</tbody>
</table>
Overview of VBScript

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbLf</td>
<td>Chr(10)</td>
<td>Line feed</td>
</tr>
<tr>
<td>vbNewLine</td>
<td>Chr(13) &amp; Chr(10) or Chr(10)</td>
<td>Platform-specific newline character; whatever is appropriate for the platform</td>
</tr>
<tr>
<td>vbNullChar</td>
<td>Chr(0)</td>
<td>Character having the value 0</td>
</tr>
<tr>
<td>vbNullString</td>
<td>String having value 0</td>
<td>Not the same as a zero-length string (“”); used for calling external procedures</td>
</tr>
<tr>
<td>vbTab</td>
<td>Chr(9)</td>
<td>Horizontal tab</td>
</tr>
<tr>
<td>vbVerticalTab</td>
<td>Chr(11)</td>
<td>Vertical tab; not useful in Microsoft Windows</td>
</tr>
</tbody>
</table>

Tristate Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbUseDefault</td>
<td>−2</td>
<td>Use default from computer's regional settings.</td>
</tr>
<tr>
<td>vbTrue</td>
<td>−1</td>
<td>TRUE</td>
</tr>
<tr>
<td>vbFalse</td>
<td>0</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

VarType Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vbEmpty</td>
<td>0</td>
<td>Uninitialized (default)</td>
</tr>
<tr>
<td>vbNull</td>
<td>1</td>
<td>Contains no valid data</td>
</tr>
<tr>
<td>vbInteger</td>
<td>2</td>
<td>Integer subtype</td>
</tr>
<tr>
<td>vbLong</td>
<td>3</td>
<td>Long subtype</td>
</tr>
<tr>
<td>vbSingle</td>
<td>4</td>
<td>Single subtype</td>
</tr>
<tr>
<td>vbDouble</td>
<td>5</td>
<td>Double subtype</td>
</tr>
<tr>
<td>vbCurrency</td>
<td>6</td>
<td>Currency subtype</td>
</tr>
<tr>
<td>vbDate</td>
<td>7</td>
<td>Date subtype</td>
</tr>
<tr>
<td>vbString</td>
<td>8</td>
<td>String subtype</td>
</tr>
<tr>
<td>vbObject</td>
<td>9</td>
<td>Object</td>
</tr>
<tr>
<td>vbError</td>
<td>10</td>
<td>Error subtype</td>
</tr>
<tr>
<td>vbBoolean</td>
<td>11</td>
<td>Boolean subtype</td>
</tr>
<tr>
<td>vbVariant</td>
<td>12</td>
<td>Variant (used only for arrays of variants)</td>
</tr>
<tr>
<td>vbDataObject</td>
<td>13</td>
<td>Data access object</td>
</tr>
<tr>
<td>vbDecimal</td>
<td>14</td>
<td>Decimal subtype</td>
</tr>
<tr>
<td>vbByte</td>
<td>17</td>
<td>Byte subtype</td>
</tr>
<tr>
<td>vbArray</td>
<td>8192</td>
<td>Array</td>
</tr>
</tbody>
</table>

Objects and Collections

Class Object

The object created using the Class statement. Provides access to the events of the class.

Debug Object

An intrinsic global object that can send output to a script debugger, such as the Microsoft Script Debugger.

Err Object
Contains information about runtime errors. Accepts the Raise and Clear methods for generating and clearing runtime errors.

**Match Object**
Provides access to the read-only properties of a regular expression match.

**Matches Collection**
Collection of regular expression Match objects.

**Regular Expression (RegExp) Object**
Provides simple regular expression support.

**SubMatches Collection**
Collection of regular expression submatch strings.

**Properties**

**Description**
Returns or sets a descriptive string associated with an error.

**FirstIndex**
Returns the position in a search string where a match occurs.

**Global**
Sets or returns a Boolean value that indicates if a pattern should match all occurrences in an entire search string or just the first one.

**HelpContext**
Sets or returns a context ID for a topic in a Help File.

**HelpFile**
Sets or returns a fully qualified path to a Help File.

**IgnoreCase**
Sets or returns a Boolean value that indicates if a pattern search is case-sensitive or not.

**Length**
Sets or returns a Boolean value that indicates if a pattern search is case-sensitive or not.

**Number**
Returns or sets a numeric value specifying an error. Number is the Err object’s default property.

**Pattern**
Sets or returns the regular expression pattern being searched for.

**Source**
Returns or sets the name of the object or application that originally generated the error.

**Value**
Returns the value or text of a match found in a search string.

**Statements**

**Call**
Transfers control to a Sub or Function procedure.

**Class**
Declares the name of a class, as well as a definition of the variables, properties, and methods that comprise the class.

**Const**
Declares constants for use in place of literal values.

**Dim**
Declares variables and allocates storage space.
Do...Loop  
Repeats a block of statements while a condition is True or until a condition becomes True.

Erase  
Reinitializes the elements of fixed-size arrays and deallocates dynamic-array storage space.

Execute  
Executes one or more specified statements.

ExecuteGlobal  
Executes one or more specified statements in the global namespace of a script.

Exit  
Exits a block of Do...Loop, For...Next, Function, or Sub code.

For Each...Next  
Repeats a group of statements for each element in an array or collection.

For...Next  
Repeats a group of statements a specified number of times.

Function  
Declares the name, arguments, and code that form the body of a Function procedure.

If...Then...Else  
Conditionally executes a group of statements, depending on the value of an expression.

Option Explicit  
Forces explicit declaration of all variables in a script.

Private  
Declares private variables and allocates storage space. Declares, in a Class block, a private variable.

Property Get  
Declares, in a Class block, the name, arguments, and code that form the body of a Property procedure that gets (returns) the value of a property.

Property Let  
Declares, in a Class block, the name, arguments, and code that form the body of a Property procedure that assigns (sets) the value of a property.

Property Set  
Declares, in a Class block, the name, arguments, and code that form the body of a Property procedure that sets a reference to an object.

Public  
Declares public variables and allocates storage space. Declares, in a Class block, a private variable.

Randomize  
Initializes the random-number generator.

ReDim  
Declares dynamic-array variables, and allocates or reallocates storage space at procedure level.

Rem  
Includes explanatory remarks in a program.

Select  
Executes one of several groups of statements, depending on the value of an expression.

Set  
Assigns an object reference to a variable or property, or associates a procedure reference with an event.
Stop
Suspends execution.

Sub
Declares the name, arguments, and code that form the body of a Sub procedure.

While
Executes a series of statements as long as a given condition is True.

With
Executes a series of statements on a single object.

Methods
Clear
Clears all property settings of the Err object.

Execute
Executes a regular expression search against a specified string.

Raise
Generates a runtime error.

Replace
Replaces text found in a regular expression search.

Test
Executes a regular expression search against a specified string and returns a Boolean value that indicates if a pattern match was found.

Write
Sends strings to the script debugger.

WriteLine
Sends strings to the script debugger, followed by a newline character.

Functions

<table>
<thead>
<tr>
<th>Function Names</th>
<th>Array</th>
<th>Asc</th>
<th>Atn</th>
<th>Abs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBool</td>
<td>CByte</td>
<td>CCur</td>
<td>CDate</td>
<td>CDate</td>
</tr>
<tr>
<td>CDbI</td>
<td>Chr</td>
<td>CInt</td>
<td>CLng</td>
<td>CLng</td>
</tr>
<tr>
<td>Conversions</td>
<td>Cos</td>
<td>CreateObject</td>
<td>CSng</td>
<td>CSng</td>
</tr>
<tr>
<td>CStr</td>
<td>Date</td>
<td>DateAdd</td>
<td>DateDiff</td>
<td>DateDiff</td>
</tr>
<tr>
<td>DatePart</td>
<td>DateSerial</td>
<td>DateValue</td>
<td>Day</td>
<td>Day</td>
</tr>
<tr>
<td>Derived Math</td>
<td>Escape</td>
<td>Eval</td>
<td>Exp</td>
<td>Exp</td>
</tr>
<tr>
<td>Filter</td>
<td>FormatCurrency</td>
<td>FormatDateTime</td>
<td>FormatNumber</td>
<td>FormatNumber</td>
</tr>
<tr>
<td>FormatPercent</td>
<td>GetLocale</td>
<td>GetObject</td>
<td>GetRef</td>
<td>GetRef</td>
</tr>
<tr>
<td>Hex</td>
<td>Hour</td>
<td>InputBox</td>
<td>InStr</td>
<td>InStr</td>
</tr>
<tr>
<td>InStrRev</td>
<td>Int, Fix</td>
<td>IsArray</td>
<td>IsDate</td>
<td>IsDate</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>IsNull</td>
<td>IsNumeric</td>
<td>IsObject</td>
<td>IsObject</td>
</tr>
<tr>
<td>Join</td>
<td>LBound</td>
<td>LCase</td>
<td>Left</td>
<td>Left</td>
</tr>
<tr>
<td>Len</td>
<td>LoadPicture</td>
<td>Log</td>
<td>LTrim; RTrim; and Trim</td>
<td>LTrim; RTrim; and Trim</td>
</tr>
<tr>
<td>Maths</td>
<td>Mid</td>
<td>Minute</td>
<td>Month</td>
<td>Month</td>
</tr>
</tbody>
</table>
### Function Names

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Name</th>
<th>Function Name</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>MonthName</td>
<td>MsgBox</td>
<td>Now</td>
<td>Oct</td>
</tr>
<tr>
<td>Replace</td>
<td>RGB</td>
<td>Right</td>
<td>Rnd</td>
</tr>
<tr>
<td>Round</td>
<td>ScriptEngine</td>
<td>ScriptEngineBuildVersion</td>
<td>ScriptEngineMajorVersion</td>
</tr>
<tr>
<td>ScriptEngineMinorVersion</td>
<td>Second</td>
<td>SetLocale</td>
<td>Sgn</td>
</tr>
<tr>
<td>Sin</td>
<td>Space</td>
<td>Split</td>
<td>Sqr</td>
</tr>
<tr>
<td>StrComp</td>
<td>String</td>
<td>StrReverse</td>
<td>Tan</td>
</tr>
<tr>
<td>Time</td>
<td>Timer</td>
<td>TimeSerial</td>
<td>TimeValue</td>
</tr>
<tr>
<td>TypeName</td>
<td>UBound</td>
<td>UCase</td>
<td>Unescape</td>
</tr>
<tr>
<td>VarType</td>
<td>Weekday</td>
<td>WeekdayName</td>
<td>Year</td>
</tr>
</tbody>
</table>

### Keywords

#### Empty

The Empty keyword is used to indicate an uninitialized variable value. This is not the same thing as Null.

#### False

The False keyword has a value equal to 0.

#### Nothing

The Nothing keyword in VBScript is used to disassociate an object variable from any actual object.

#### Null

The Null keyword is used to indicate that a variable contains no valid data. This is not the same thing as Empty.

#### True

The True keyword has a value equal to -1.

### Errors

#### VBScript Runtime Errors

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Invalid procedure call or argument</td>
</tr>
<tr>
<td>6</td>
<td>Overflow</td>
</tr>
<tr>
<td>7</td>
<td>Out of memory</td>
</tr>
<tr>
<td>9</td>
<td>Subscript out of range</td>
</tr>
<tr>
<td>10</td>
<td>This array is fixed or temporarily locked</td>
</tr>
<tr>
<td>11</td>
<td>Division by zero</td>
</tr>
<tr>
<td>13</td>
<td>Type mismatch</td>
</tr>
<tr>
<td>14</td>
<td>Out of string space</td>
</tr>
<tr>
<td>17</td>
<td>Can't perform requested operation</td>
</tr>
<tr>
<td>28</td>
<td>Out of stack space</td>
</tr>
<tr>
<td>35</td>
<td>Sub or function not defined</td>
</tr>
<tr>
<td>48</td>
<td>Error in loading DLL</td>
</tr>
<tr>
<td>51</td>
<td>Internal error</td>
</tr>
</tbody>
</table>
### Error Number | Description
--- | ---
91 | Object variable not set
92 | For loop not initialized
94 | Invalid use of Null
424 | Object required
429 | ActiveX component can't create object
430 | Class doesn't support Automation
432 | File name or class name not found during Automation operation
438 | Object doesn't support this property or method
445 | Object doesn't support this action
447 | Object doesn't support current locale setting
448 | Named argument not found
449 | Argument not optional
450 | Wrong number of arguments or invalid property assignment
451 | Object not a collection
458 | Variable uses an Automation type not supported in VBScript
462 | The remote server machine does not exist or is unavailable
481 | Invalid picture
500 | Variable is undefined
502 | Object not safe for scripting
503 | Object not safe for initializing
504 | Object not safe for creating
505 | Invalid or unqualified reference
506 | Class not defined
507 | An exception occurred
5008 | Illegal assignment
5017 | Syntax error in regular expression
5018 | Unexpected quantifier
5019 | Expected '[' in regular expression
5020 | Expected ']' in regular expression
5021 | Invalid range in character set

### VBScript Syntax Errors

| Error Number | Description |
--- | ---|
1001 | Out of memory |
1002 | Syntax error |
1005 | Expected '(' |
1006 | Expected ')' |
1010 | Expected identifier |
1011 | Expected "=" |
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1012</td>
<td>Expected 'If'</td>
</tr>
<tr>
<td>1013</td>
<td>Expected 'To'</td>
</tr>
<tr>
<td>1014</td>
<td>Expected 'End'</td>
</tr>
<tr>
<td>1015</td>
<td>Expected 'Function'</td>
</tr>
<tr>
<td>1016</td>
<td>Expected 'Sub'</td>
</tr>
<tr>
<td>1017</td>
<td>Expected 'Then'</td>
</tr>
<tr>
<td>1018</td>
<td>Expected 'Wend'</td>
</tr>
<tr>
<td>1019</td>
<td>Expected 'Loop'</td>
</tr>
<tr>
<td>1020</td>
<td>Expected 'Next'</td>
</tr>
<tr>
<td>1021</td>
<td>Expected 'Case'</td>
</tr>
<tr>
<td>1022</td>
<td>Expected 'Select'</td>
</tr>
<tr>
<td>1023</td>
<td>Expected expression</td>
</tr>
<tr>
<td>1024</td>
<td>Expected statement</td>
</tr>
<tr>
<td>1025</td>
<td>Expected end of statement</td>
</tr>
<tr>
<td>1026</td>
<td>Expected integer constant</td>
</tr>
<tr>
<td>1027</td>
<td>Expected 'While' or 'Until'</td>
</tr>
<tr>
<td>1028</td>
<td>Expected 'While', 'Until', or end of statement</td>
</tr>
<tr>
<td>1029</td>
<td>Expected 'With'</td>
</tr>
<tr>
<td>1030</td>
<td>Identifier too long</td>
</tr>
<tr>
<td>1037</td>
<td>Invalid use of 'Me' keyword</td>
</tr>
<tr>
<td>1038</td>
<td>'loop' without 'do'</td>
</tr>
<tr>
<td>1039</td>
<td>Invalid 'exit' statement</td>
</tr>
<tr>
<td>1040</td>
<td>Invalid 'for' loop control variable</td>
</tr>
<tr>
<td>1041</td>
<td>Name redefined</td>
</tr>
<tr>
<td>1042</td>
<td>Must be first statement on the line</td>
</tr>
<tr>
<td>1044</td>
<td>Cannot use parentheses when calling a Sub</td>
</tr>
<tr>
<td>1045</td>
<td>Expected literal constant</td>
</tr>
<tr>
<td>1046</td>
<td>Expected 'in'</td>
</tr>
<tr>
<td>1047</td>
<td>Expected 'Class'</td>
</tr>
<tr>
<td>1048</td>
<td>Must be defined inside a Class</td>
</tr>
<tr>
<td>1049</td>
<td>Expected Let or Set or Get in property declaration</td>
</tr>
<tr>
<td>1050</td>
<td>Expected 'Property'</td>
</tr>
<tr>
<td>1051</td>
<td>Number of arguments must be consistent across properties specification</td>
</tr>
<tr>
<td>1052</td>
<td>Cannot have multiple default property/method in a Class</td>
</tr>
<tr>
<td>1053</td>
<td>Class initialize or terminate do not have arguments</td>
</tr>
<tr>
<td>1054</td>
<td>Property Set or Let must have at least one argument</td>
</tr>
<tr>
<td>1055</td>
<td>Unexpected 'Next'</td>
</tr>
<tr>
<td>1057</td>
<td>'Default' specification must also specify 'Public'</td>
</tr>
<tr>
<td>1058</td>
<td>'Default' specification can only be on Property Get</td>
</tr>
</tbody>
</table>
Tips & Tricks

**VBScript Editor IntelliSense**

IntelliSense provides an array of options that make language references easily accessible. When coding, you do not need to leave the Code Editor or the Immediate Mode command window to perform searches on language elements. You can keep your context, find the information you need, insert language elements directly into your code, and even have IntelliSense complete your typing for you.

IntelliSense comprises the following options...

**List Members**

You can display a list of valid members from class tags, fields from any tag, properties/methods from an ActiveX object, or functions from the Built-in Scripting Language. Selecting from the list inserts the member into your code.

When you type the $ character on any VBScript interface, a list box will automatically open with the list of all tags available for the current project as well as all functions from the Built-In Scripting Language.

![List Members Example](image)

When you type the name of a class tag followed by the dot character ( . ) on any VBScript interface, a list box will automatically open with the list of all members from the class tag:

![Class Members Example](image)

When you type the name of a tag followed by the hyphen and greater than characters ( - > ) on any VBScript interface, a list box will automatically open with the list of all fields available for this tag:

![Field Members Example](image)

The items are displayed in alphabetic order, and each item has an icon to identify its main type, as follows:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Boolean Tag</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Integer Tag</td>
</tr>
</tbody>
</table>
Overview of VBScript

<table>
<thead>
<tr>
<th>Icon</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="Image" /></td>
<td>Real Tag</td>
</tr>
<tr>
<td><img src="Image" alt="Image" /></td>
<td>String Tag</td>
</tr>
<tr>
<td><img src="Image" alt="Image" /></td>
<td>Class Tag</td>
</tr>
<tr>
<td><img src="Image" alt="Image" /></td>
<td>Function from the Built-in Scripting Language</td>
</tr>
</tbody>
</table>

When you type the name of an ActiveX control that is inserted on the screen followed by the dot character (.) on any VBScript interface from the screen where the ActiveX object is inserted, a list box will automatically open with the list of all properties and methods from the object:

The items are displayed in alphabetic order, and each item has an icon to identify its main type, as follows:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="Image" /></td>
<td>Property from the ActiveX object</td>
</tr>
<tr>
<td><img src="Image" alt="Image" /></td>
<td>Method from the ActiveX object</td>
</tr>
</tbody>
</table>

**Parameter Quick Info**

The Quick Info option displays pop-up boxes with the information about the functions from the Built-in Scripting Language. The information includes all the parameters supported by this function, with the currently configured one in bold text.

**Complete Word**

Complete word finishes a tag, member, field, function, or ActiveX property/method name once you have entered enough characters to disambiguate the term. After you type the first few letters of the name, you can press Ctrl+Space to complete the name automatically.

**VBScript Compared to VBA**

While VBScript and Visual Basic for Applications (VBA) are similar and are both based on the Visual Basic standard language, there are advantages to using VBScript for IWS users:

- VBScript is supported for the Microsoft Windows Embedded operating system, and VBA is not.
- VBScript brings active scripting to a wide variety of environments, including Web client scripting in Microsoft Internet Explorer. This prevents operations that may present risks for the Thin Client user, such as direct access to local files.
- VBScript was designed to be simple and easy to learn, with some standards from VBA modified in VBScript to make it more straightforward. For example, in VBScript the user does not have to worry about the type of each variable when declaring them because VBScript assumes the proper type for each variable automatically.

The following table lists VBScript features that VBA does not have.
### Overview of VBScript

<table>
<thead>
<tr>
<th>Category</th>
<th>Feature/Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declarations</td>
<td>Class</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Eval, Execute</td>
</tr>
<tr>
<td>Objects</td>
<td>RegExp</td>
</tr>
<tr>
<td>Script Engine Identification</td>
<td>ScriptEngine,</td>
</tr>
<tr>
<td></td>
<td>ScriptEngineBuildVersion</td>
</tr>
<tr>
<td></td>
<td>ScriptEngineMajorVersion</td>
</tr>
</tbody>
</table>

The following table lists VBA features that VBScript does not have.

<table>
<thead>
<tr>
<th>Category</th>
<th>Omitted Feature/Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array Handling</td>
<td>Option Base</td>
</tr>
<tr>
<td></td>
<td>Declaring arrays with lower bound &lt;&gt; 0</td>
</tr>
<tr>
<td>Collection</td>
<td>Add, Count, Item, Remove</td>
</tr>
<tr>
<td></td>
<td>Access to collections using ! character</td>
</tr>
<tr>
<td>Conditional Compilation</td>
<td>#Const</td>
</tr>
<tr>
<td></td>
<td>#If...Then...#Else</td>
</tr>
<tr>
<td>Control Flow</td>
<td>DoEvents</td>
</tr>
<tr>
<td></td>
<td>GoSub...Return, GoTo</td>
</tr>
<tr>
<td></td>
<td>On Error GoTo</td>
</tr>
<tr>
<td></td>
<td>On...GoSub, On...GoTo</td>
</tr>
<tr>
<td></td>
<td>Line numbers, Line labels</td>
</tr>
<tr>
<td>Conversion</td>
<td>CVar, CVDate</td>
</tr>
<tr>
<td></td>
<td>Str, Val</td>
</tr>
<tr>
<td>Data Types</td>
<td>All intrinsic data types except Variant</td>
</tr>
<tr>
<td></td>
<td>Type...End Type</td>
</tr>
<tr>
<td>Date/Time</td>
<td>Date statement, Time statement</td>
</tr>
<tr>
<td>DDE</td>
<td>LinkExecute, LinkPoke, LinkRequest, LinkSend</td>
</tr>
<tr>
<td>Debugging</td>
<td>Debug.Print</td>
</tr>
<tr>
<td></td>
<td>End, Stop</td>
</tr>
<tr>
<td>Declaration</td>
<td>Declare (for declaring DLLs)</td>
</tr>
<tr>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>ParamArray</td>
</tr>
<tr>
<td></td>
<td>Static</td>
</tr>
<tr>
<td>Error Handling</td>
<td>Err</td>
</tr>
<tr>
<td></td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td>Resume, Resume Next</td>
</tr>
<tr>
<td>File Input/Output</td>
<td>All traditional Basic file I/O</td>
</tr>
<tr>
<td>Financial</td>
<td>All financial functions</td>
</tr>
<tr>
<td>Object Manipulation</td>
<td>TypeOf</td>
</tr>
</tbody>
</table>
### Screen Events

In addition to the Screen Script, you can configure logics using the Built-in Scripting Language for the On Open, While Open and On Close events for the screen (see the Screen Logic interface from the Screen Attributes dialog). If you configure the Screen Script (VBScript language) and the Screen Logic (Built-in Scripting Language), IWS will respect the following execution order:

<table>
<thead>
<tr>
<th>Event</th>
<th>Order of execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>When opening the screen</td>
<td>• Screen_OnOpen() sub-routine from the Screen Script interface (VBScript language)</td>
</tr>
<tr>
<td></td>
<td>• On Open from the Screen Logic interface (Built-in Scripting Language)</td>
</tr>
<tr>
<td>When closing the screen</td>
<td>• On Close from the Screen Logic interface (Built-in Scripting Language)</td>
</tr>
<tr>
<td></td>
<td>• Screen_OnClose() sub-routine from the Screen Script interface (VBScript language)</td>
</tr>
</tbody>
</table>

### MsgBox and InputBox Functions

The `MsgBox()` and `InputBox()` functions from the VBScript language allow you to display pop-up messages during runtime. These functions are synchronous. When either one is executed, the remaining instructions from the code will not be executed before the pop-up messages launched by the functions are closed.

**Note:** The text displayed in these pop-up messages are not affected by the Translation Tool of IWS, unless you configure the text explicitly using the `$Ext()` function from the Built-in Scripting Language.

### VBScript Procedures

In VBScript, there are two kinds of procedures; the Sub procedure and the Function procedure.

**Sub Procedures**

A Sub procedure is a series of VBScript statements (enclosed by `Sub` and `End Sub` statements) that perform actions but don’t return a value. A Sub procedure can take arguments (constants, variables, or expressions that are passed by a calling procedure). If a Sub procedure has no arguments, its `Sub` statement must include an empty set of parentheses `()`. 
The following Sub procedure uses two intrinsic (built-in) VBScript functions, `MsgBox` and `InputBox`, to prompt a user for information. It then displays the results of a calculation based on that information. The calculation is performed in a Function procedure created with VBScript. The Function procedure is shown after the following discussion.

```
Sub ConvertTemp()
    temp = InputBox("Please enter the temperature in degrees F.", 1)
    MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub
```

### Function Procedures

A Function procedure is a series of VBScript statements enclosed by the `Function` and `End Function` statements. A Function procedure is similar to a Sub procedure, but can also return a value. A Function procedure can take arguments (constants, variables or expressions that are passed to it by a calling procedure). If a Function procedure has no arguments, its `Function` statement must include an empty set of parentheses. A Function returns a value by assigning a value to its name in one or more statements of the procedure. The return type of a Function is always a Variant.

In the following example, the `Celsius` function calculates degrees Celsius from degrees Fahrenheit. When the function is called from the `ConvertTemp` Sub procedure, a variable containing the argument value is passed to the function. The result of the calculation is returned to the calling procedure and displayed in a message box.

```
Sub ConvertTemp()
    temp = InputBox("Please enter the temperature in degrees F.", 1)
    MsgBox "The temperature is " & Celsius(temp) & " degrees C."
End Sub

Function Celsius(fDegrees)
    Celsius = (fDegrees - 32) * 5 / 9
End Function
```

### Getting Data Into and Out of Procedures

Each piece of data is passed into your procedures using an argument. Arguments serve as placeholders for the data you want to pass into your procedure. You can name your arguments any valid variable name. When you create a procedure using either the `Sub` statement or the `Function` statement, parentheses must be included after the name of the procedure. Any arguments are placed inside these parentheses, separated by commas. For example, in the following example, `fDegrees` is a placeholder for the value being passed into the `Celsius` function for conversion.

```
Function Celsius(fDegrees)
    Celsius = (fDegrees - 32) * 5 / 9
End Function
```

To get data out of a procedure, you must use a Function. Remember, a Function procedure can return a value; a Sub procedure cannot.

### Using Sub and Function Procedures in Code

A Function in your code must always be used on the right side of a variable assignment or in an expression. For example:

```
Temp = Celsius(fDegrees)
```

or

```
MsgBox "The Celsius temperature is " & Celsius(fDegrees) & " degrees."
```
To call a Sub procedure from another procedure, type the name of the procedure along with values for any required arguments, each separated by a comma. The `Call` statement is not required, but if you do use it, you must enclose any arguments in parentheses.

The following example shows two calls to the `MyProc` procedure. One uses the `Call` statement in the code; the other doesn’t. Both do exactly the same thing.

```vbs
Call MyProc(firstarg, secondarg)
MyProc firstarg, secondarg
```

Notice that the parentheses are omitted in the call when the `Call` statement isn’t used.

**Creating Constants**

A constant is a meaningful name that takes the place of a number or string and never changes. VBScript defines a number of intrinsic constants.

You create user-defined constants in VBScript using the `Const` statement. Using the `Const` statement, you can create string or numeric constants with meaningful names and assign them literal values. For example:

```vbs
Const MyString = "This is my string."
Const MyAge = 49
```

Note that the string literal is enclosed in quotation marks (" "). Quotation marks are the most obvious way to differentiate string values from numeric values. You represent Date literals and time literals by enclosing them in number signs (#). For example:

```vbs
Const CutoffDate = #6-1-97#
```

You may want to adopt a naming scheme to differentiate constants from variables. This will prevent you from trying to reassign constant values while your script is running. For example, you might want to use a "vb" or "con" prefix on your constant names, or you might name your constants in all capital letters. Differentiating constants from variables eliminates confusion as you develop more complex scripts.

**Declaring Variables**

A variable is a convenient placeholder that refers to a computer memory location where you can store program information that may change during the time your script is running. In VBScript, variables are always of one fundamental data type, Variant.

You declare variables explicitly in your script using the `Dim` statement, the `Public` statement, and the `Private` statement. For example:

```vbs
Dim DegreesFahrenheit
```

You declare multiple variables by separating each variable name with a comma. For example:

```vbs
Dim Top, Bottom, Left, Right
```

You can also declare a variable implicitly by simply using its name in your script. That is not generally a good practice because you could misspell the variable name in one or more places, causing unexpected results when your script is run. For that reason, the `Option Explicit` statement is configured by default in the Global Procedures interface to require explicit declaration of all variables. Unless you delete this statement, you need to declare all variables explicitly; otherwise, VBScript will generate errors during runtime indicating that the variable does not exist.

An expression should have the variable on the left side and the value you want to assign to the variable on the right. For example:

```vbs
MyVar = 100
```
**Scope and Lifetime of Variables**

A variable’s scope is determined by where you declare it. When you declare a variable within a procedure, only code within that procedure can access or change the value of that variable. It has local scope and is a procedure-level variable. If you declare a variable outside a procedure, you make it recognizable to all the procedures in your script. This is a script-level variable, and it has script-level scope.

The lifetime of a variable depends on how long it exists. The lifetime of a script-level variable extends from the time it is declared until the time the script is finished running. At procedure level, a variable exists only as the procedure runs. When the procedure exits, the variable is destroyed. Local variables are ideal as temporary storage space when a procedure is executing. You can have local variables of the same name in several different procedures because each is recognized only by the procedure in which it is declared.

**Boolean tags and Boolean variables**

By default, Boolean tags (i.e., project tags of Boolean type) in InduSoft Web Studio are handled differently than Boolean variables in VBScript. The Boolean states of FALSE and TRUE have the same meanings, but the numerical values of TRUE are different, as shown in the table below:

<table>
<thead>
<tr>
<th>Boolean state</th>
<th>Numerical value in a...</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IWS project tag</td>
<td>VBScript variable</td>
</tr>
<tr>
<td>FALSE</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TRUE</td>
<td>1 (or non-zero)</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Note:** `False` and `True` are reserved as **keywords** in VBScript.

This can cause problems during run time as logical and arithmetic operations — especially the **NOT operator** — might change tag values in unexpected ways. To prevent these problems and ensure that Boolean tags have the correct values at all times, the VBScript interface in InduSoft Web Studio has been modified to preprocess Boolean tags and handle them like Boolean variables. In other words, while a Boolean tag has an actual value of 1 in the tags database, it is handled as if it has a value of -1 in VBScript so that you can perform operations using both Boolean tags and Boolean variables.

This modification was done in a previous version of InduSoft Web Studio — specifically, in InduSoft Web Studio v6.1+SP4. Projects that were developed with earlier versions of InduSoft Web Studio and then upgraded to the latest version have not been modified, in order to maintain backward compatibility. If you developed your project with an earlier version of InduSoft Web Studio, you can do the modification yourself by manually editing your project file (`projectname.APP`) to add the following entry:

```
[Script]
VBBoolean=1
```

After you do this, however, you must thoroughly test your project to make sure that your VBScript code still behaves as expected.

**Note:** Be careful when defining a custom property on a **Linked Symbol** using the `#Label:@Pointer` syntax. For example:

```vbnet
'The following statements are valid
If $MyBoolean = 1 Then
  End If

If $MyBoolean = True Then
  End If

If #Mne:@MyPointer = True Then
  End If

'The following statement is invalid
If #Mne:@MyPointer = 1 Then
```
Writing Real Values to Integer Tags

By default, a Real (i.e., floating point) value is truncated at the decimal point when it is written to an Integer tag. This behavior is the same in both the Built-in Scripting Language and in VBScript.

You can change this behavior in VBScript, however, by disabling the "TruncRealToInt" runtime setting. To do this, edit your project file (`project_name.app`) to change the following line:

```vbnet
[Script]
TruncRealToInt=value
```

If `TruncRealToInt` is set to 1, then the project will behave as described above: Real values will be truncated at the decimal point without rounding. (For example, a value of 5.56 will be written as 5 to an Integer tag.) This is the default setting for projects created with InduSoft Web Studio v6.1+SP4 or earlier and then updated to v6.1+SP5, in order to maintain backward compatibility.

If `TruncRealToInt` is set to 0, then VBScript functions and operations will round Real values to the nearest integer. (For example, a value of 5.56 will be written as 6 to an Integer tag.) This is the default setting for projects created with InduSoft Web Studio v6.1+SP5 or later.

**Note:** This setting only affects the behavior of VBScript in IWS. It does not affect the behavior of the Built-in Scripting Language.

Precedence of VBScript Operators

VBScript has a full range of operators, including arithmetic operators, comparison operators, concatenation operators, and logical operators.

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called "operator precedence." You can use parentheses to override the order of precedence and force some parts of an expression to be evaluated before others. Operations within parentheses are always performed before those outside. Within parentheses, however, standard operator precedence is maintained.

When expressions contain operators from more than one category, arithmetic operators are evaluated first, comparison operators are evaluated next, and logical operators are evaluated last. Comparison operators all have equal precedence; that is, they are evaluated in the left-to-right order in which they appear. Arithmetic and logical operators are evaluated in the following order of precedence.

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>Comparison</th>
<th>Logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation (-)</td>
<td>Equality (=)</td>
<td>Not</td>
</tr>
<tr>
<td>Exponentiation (^)</td>
<td>Inequality (&lt;&gt;</td>
<td>And</td>
</tr>
<tr>
<td>Multiplication and division (*, /)</td>
<td>Less than (&lt;)</td>
<td>Or</td>
</tr>
<tr>
<td>Integer division ()</td>
<td>Greater than (&gt;</td>
<td>Xor</td>
</tr>
<tr>
<td>Modulus arithmetic (Mod)</td>
<td>Less than or equal to (&lt;=)</td>
<td>Eqv</td>
</tr>
<tr>
<td>Addition and subtraction (+, -)</td>
<td>Greater than or equal to (&gt;=)</td>
<td>Imp</td>
</tr>
<tr>
<td>String concatenation (&amp;)</td>
<td>Is</td>
<td>&amp;</td>
</tr>
</tbody>
</table>

When multiplication and division occur together in an expression, each operation is evaluated as it occurs from left to right. Likewise, when addition and subtraction occur together in an expression, each operation is evaluated in order of appearance from left to right.

The string concatenation (&) operator is not an arithmetic operator, but in precedence it falls after all arithmetic operators and before all comparison operators. The Is operator is an object reference comparison operator. It does not compare objects or their values; it checks only to determine if two object references refer to the same object.
Logical Operator NOT

The logical operator NOT behaves differently in VBScript than it does in the built-in scripting language.

NOT Operator in VBScript

In VBScript, the NOT operator inverts the bits of a given numeric value, producing its complement number according to the “two's complement” system of signed numbers that is used by computers. The table below illustrates the behavior of the NOT operator in VBScript for the syntax...

\[
\text{result} = \text{NOT expression}
\]

<table>
<thead>
<tr>
<th>If expression is...</th>
<th>Then result is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>2</td>
</tr>
<tr>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>2</td>
<td>-3</td>
</tr>
<tr>
<td>3</td>
<td>-4</td>
</tr>
</tbody>
</table>

Note: By default, when you attempt to write any numeric value other than 0 to a Boolean tag, the tag automatically assumes a value of 1. Therefore, if VBScript's NOT operator is applied to a Boolean tag with a value of 1, then the value of the tag does not change; the operator returns a value of −2, but the tag cannot accept this value so it again assumes a value of 1.

You can configure IWS to treat Boolean tags like Boolean variables in VBScript, so that the NOT operator in VBScript will work as expected. For more information, please see Boolean Tags and Boolean Variables.

NOT Operator in Built-in Language

In contrast, the NOT operator in the Built-in Scripting Language toggles the given numeric value as if it is a natural boolean. The table below illustrates the behavior of the NOT operator in the Built-in Scripting Language for the syntax...

\[
\text{result} = \text{NOT expression}
\]

<table>
<thead>
<tr>
<th>If expression is...</th>
<th>Then result is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>≠0</td>
<td>0</td>
</tr>
</tbody>
</table>

Using Conditional Statements

You can control the flow of your script with conditional statements and looping statements. Using conditional statements, you can write VBScript code that makes decisions and repeats actions. The following conditional statements are available in VBScript:

- If…Then…Else statement
- Select Case statement

Making Decisions Using If…Then…Else

The If…Then…Else statement is used to evaluate whether a condition is True or False and, depending on the result, to specify one or more statements to run. Usually the condition is an expression that uses
a comparison operator to compare one value or variable with another. For information about comparison operators, see Comparison Operators.

**If...Then...Else** statements can be nested to as many levels as you need.

**Running Statements if a Condition is True**

To run only one statement when a condition is True, use the single-line syntax for the If...Then...Else statement. The following example shows the single-line syntax. Notice that this example omits the Else keyword:

```vbscript
Sub FixDate()
    Dim myDate
    myDate = #2/13/95#
    If myDate < Now Then myDate = Now
End Sub
```

To run more than one line of code, you must use the multiple-line (or block) syntax. This syntax includes the End If statement, as shown in the following example:

```vbscript
Sub AlertUser(value)
    If value = 0 Then
        AlertLabel.ForeColor = vbRed
        AlertLabel.Font.Bold = True
        AlertLabel.Font.Italic = True
    End If
End Sub
```

**Running Certain Statements if a Condition is True and Running Others if a Condition is False**

You can use an If...Then...Else statement to define two blocks of executable statements: one block to run if the condition is True, and the other block to run if the condition is False:

```vbscript
Sub AlertUser(value)
    If value = 0 Then
        MsgBox value
    ElseIf value = 1 Then
        MsgBox value
    ElseIf value = 2 Then
        MsgBox value
    Else
        MsgBox "Value out of range!"
    End If
End Sub
```

**Deciding Between Several Alternatives**

A variation on the If...Then...Else statement allows you to choose from several alternatives. Adding ElseIf clauses expands the functionality of the If...Then...Else statement, so you can control program flow based on different possibilities. For example:

```vbscript
Sub ReportValue(value)
    If value = 0 Then
        MsgBox value
    ElseIf value = 1 Then
        MsgBox value
    ElseIf value = 2 then
        MsgBox value
    Else
        MsgBox "Value out of range!"
    End If
End Sub
```
You can add as many ElseIf clauses as you need to provide alternative choices, but extensive use of the ElseIf clauses often becomes cumbersome. A better way to choose between several alternatives is the Select Case statement.

Making Decisions with Select Case

The Select Case structure provides an alternative to If…Then…ElseIf for selectively executing one block of statements from among multiple blocks of statements. A Select Case statement provides capability similar to the If…Then…Else statement, but it makes code more efficient and readable.

A Select Case structure works with a single test expression that is evaluated once, at the top of the structure. The result of the expression is then compared to the values for each Case in the structure. If there is a match, the block of statements associated with that Case is executed, as in the following example:

```vbnet
Select Case Document.Form1.CardType.Options(SelectedIndex).Text
    Case "MasterCard"
        DisplayMCLogo
        ValidateMCAccount
    Case "Visa"
        DisplayVisaLogo
        ValidateVisaAccount
    Case "American Express"
        DisplayAMEXCOLogo
        ValidateAMEXCOAccount
    Case Else
        DisplayUnknownImage
        PromptAgain
End Select
```

Notice that the Select Case structure evaluates an expression once at the top of the structure. In contrast, the If…Then…ElseIf structure can evaluate a different expression for each ElseIf statement. You can replace an If…Then…ElseIf structure with a Select Case structure only if each ElseIf statement evaluates the same expression.

Looping Through Code

Looping allows you to run a group of statements repeatedly. Some loops repeat statements until a condition is False; others repeat statements until a condition is True. There are also loops that repeat statements a specific number of times.

The following looping statements are available in VBScript:

- **Do…Loop**: Loops while or until a condition is True
- **While…Wend**: Loops while a condition is True
- **For…Next**: Uses a counter to run statements a specified number of times

Using Do Loops

You can use Do…Loop statements to run a block of statements an indefinite number of times. The statements are repeated either while a condition is True or until a condition becomes True.

Repeating Statements While a Condition is True

Use the While keyword to check a condition in a Do…Loop statement. You can check the condition before you enter the loop (as shown in the following ChkFirstWhile example), or you can check it after the loop has run at least once (as shown in the ChkLastWhile example). In the ChkFirstWhile procedure, if myNum is set to 9 instead of 20, the statements inside the loop will never run. In the ChkLastWhile procedure, the statements inside the loop run only once because the condition is already False.

```vbnet
Sub ChkFirstWhile()
    Dim counter, myNum
    counter = 0
    myNum = 20
    Do While myNum > 10
        myNum = myNum - 1
End Sub
```
counter = counter + 1
Loop
MsgBox "The loop made " & counter & " repetitions."
End Sub

Sub ChkLastWhile()
Dim counter, myNum
counter = 0
myNum = 9
Do
    myNum = myNum - 1
    counter = counter + 1
Loop While myNum > 10
MsgBox "The loop made " & counter & " repetitions."
End Sub

Repeating a Statement Until a Condition Becomes True

There are two ways to use the Until keyword to check a condition in a Do...Loop statement. You can check the condition before you enter the loop (as shown in the following ChkFirstUntil example), or you can check it after the loop has run at least once (as shown in the ChkLastUntil example). As long as the condition is False, the looping occurs.

Sub ChkFirstUntil()
Dim counter, myNum
counter = 0
myNum = 20
Do Until myNum = 10
    myNum = myNum - 1
    counter = counter + 1
Loop
MsgBox "The loop made " & counter & " repetitions."
End Sub

Sub ChkLastUntil()
Dim counter, myNum
counter = 0
myNum = 1
Do
    myNum = myNum + 1
    counter = counter + 1
Loop Until myNum = 10
MsgBox "The loop made " & counter & " repetitions."
End Sub

Exiting a Do...Loop Statement from Inside the Loop

You can exit a Do...Loop by using the Exit Do statement. Because you usually want to exit only in certain situations, such as to avoid an endless loop, you should use the Exit Do statement in the True statement block of an If...Then...Else statement. If the condition is False, the loop runs as usual.

In the following example, myNum is assigned a value that creates an endless loop. The If...Then...Else statement checks for this condition, preventing the endless repetition.

Sub ExitExample()
Dim counter, myNum
counter = 0
myNum = 9
Do Until myNum = 10
    myNum = myNum - 1
    counter = counter + 1
    If myNum < 10 Then Exit Do
Loop
MsgBox "The loop made " & counter & " repetitions."
End Sub
Using While...Wend

The *While...Wend* statement is provided in VBScript for those who are familiar with its usage. However, because of the lack of flexibility in *While...Wend*, it is recommended that you use *Do...Loop* instead.

Using For...Next

You can use *For...Next* statements to run a block of statements a specific number of times. For loops, use a counter variable whose value increases or decreases with each repetition of the loop.

The following example causes a procedure called *MyProc* to execute 50 times. The For statement specifies the counter variable *x* and its start and end values. The *Next* statement increments the counter variable by 1.

```vbnet
Sub DoMyProc50Times()
    Dim x
    For x = 1 To 50
        MyProc
    Next
End Sub
```

Using the *Step* keyword, you can increase or decrease the counter variable by the value you specify. In the following example, the counter variable *j* is incremented by 2 each time the loop repeats. When the loop is finished, the total is the sum of 2, 4, 6, 8, and 10.

```vbnet
Sub TwosTotal()
    Dim j, total
    For j = 2 To 10 Step 2
        total = total + j
    Next
    MsgBox "The total is " & total
End Sub
```

To decrease the counter variable, use a negative *Step* value. You must specify an end value that is less than the start value. In the following example, the counter variable *myNum* is decreased by 2 each time the loop repeats. When the loop is finished, the total is the sum of 16, 14, 12, 10, 8, 6, 4, and 2.

```vbnet
Sub NewTotal()
    Dim myNum, total
    For myNum = 16 To 2 Step -2
        total = total + myNum
    Next
    MsgBox "The total is " & total
End Sub
```

You can exit any *For...Next* statement before the counter reaches its end value by using the *Exit For* statement. Because you usually want to exit only in certain situations, such as when an error occurs, you should use the *Exit For* statement in the *True* statement block of an *If...Then...Else* statement. If the condition is *False*, the loop runs as usual.

Support for ActiveX Controls

Using the VBScript interfaces for the Graphic module (Graphics Script, Screen Script, Command animation, and ActiveX Events), you can use this syntax to access properties and methods directly from any ActiveX Control object inserted in the screen where the object is configured.

IWS will assign a unique name to the object on the screen. You can use the Name property (in the *Object Properties* dialog) to modify this name.

After inserting an ActiveX Control object on the screen, you can access properties and methods from this object from any VBScript interface associated with this screen. Use the syntax *Object_Name.Properties_or_Method_Name*. Examples:

```vbnet
//Access the value of the property Day from the CalendarControl1 ActiveX object
CalendarControl1.Day
```
//Triggers the method AboutBox from the CalendarControll ActiveX object
CalendarControll1.AboutBox

**Support for VBScript on Windows Embedded**

EmbeddedView and CEView (i.e., the IWS project runtime engines for Windows Embedded Standard and Windows Embedded Compact, respectively) can execute the VBScript in your project, but only if the Windows Embedded device itself supports VBScript. The device's OS image must be configured to enable support for VBScript, and that is typically done by the manufacturer of the device. Resources are limited on embedded devices, so manufacturers often disable system components that they think will not be needed. If you do not know whether your Windows Embedded device supports VBScript, please contact the manufacturer or consult the manufacturer's documentation.

**Note:** The manufacturer can specifically enable/disable support for the VBScript functions `MsgBox` and `InputBox`. 
Debugging VBScript

The development application provides additional tools for debugging your VBScript code.

**Tip:** These tools are based on the debugging tools in Microsoft Visual Studio, so if you are experienced with Visual Studio, then these tools should be familiar to you as well.

The tools are available in the following **VBScript interfaces** in the development application:
- Procedures, in the **Global** tab of the Project Explorer;
- The Graphics Script, in the **Graphics** tab of the Project Explorer;
- Script groups, including the Startup Script, in the **Tasks** tab of the Project Explorer; and
- The Screen Script that is attached to each Screen worksheet.

At this time, debugging is not supported in the following VBScript interfaces:
- The Command animation that is attached to each Button object; and
- ActiveX and .NET Control objects.

Generally speaking, you can debug your code by running your project in **Debug** mode (as opposed to the usual **Run** mode) and then observing how the values of project tags, locally-declared variables, and entire functions change as you step through the code. You can control the stepping by using the **Debug** tab of the ribbon, and you can observe the changing values in the **Database Spy** window.

**About the Debug tab**

The **Debug** tab of the ribbon is used to debug the VBScript code in your project.

**Note:** This tab of the ribbon is contextual: it appears only when you use a VBScript interface to view or edit your code.

The tools are organized into the following groups:

**Debug Options**

These are general options for the debugging tools.

First, the **Restore Default** tool restores the development environment to its default layout. It is the same as the **Restore Default** tool on the **View** tab of the ribbon, and it is useful when you have extensively resized and rearranged the windows to facilitate debugging.

Second, the **Debug**... options determine exactly which parts of the project runtime will be debugged. Normally, both parts are debugged at the same time, but if you want one to run without interruption while you focus on the other, then you can control that here.

**Local Management**

These tools are the same as in the **Local Management** group on the **Home** tab of the ribbon. You can use them to run and stop your project, as well as to check the states of the many tasks and modules that make up the project runtime.

**Debug Tools**

These tools control the actual code stepping when you run your project in Debug mode. For more information about these tools, see the rest of the “Debugging VBScript” section.
Set break points in your VBScript code

Set break points in your VBScript code to indicate where run-time execution should be suspended.

As you develop your code, you will identify important sections where you want to focus your attention while debugging. For example, if your code includes a critical function that updates the values of several project tags, then you may want to suspend execution at the beginning of that function and observe the value changes as they happen. That is where you should set a break point.

A break point is a signal that tells the debugger to temporarily suspend execution of your project. When execution is suspended at a break point, your project is said to be in break mode. Entering break mode does not stop or end the execution of your project; execution can be resumed at any time.

You can think of break mode as being like a time-out. All the elements remain (functions, variables, and objects remain in memory, for example), but their movements and activities are suspended. During break mode, you can examine the elements’ positions and states to look for violations or bugs. You can also make adjustments to the project while in break mode; for example, you can change the value of a variable.

Break points provide a powerful tool that enables you to suspend execution where and when you need to. Rather than stepping through your code line by line, you can allow your project to run until it hits a break point, and then start to debug. This speeds up the debugging process. Without this ability, it would be almost impossible to debug a large project.

To set one or more break points in your VBScript code:

1. Open the VBScript worksheet that you want to debug.
   The worksheet must be in one of the VBScript interfaces that supports debugging. For more information, see Debugging VBScript on page 983.
   The worksheet is opened for editing.

2. In the worksheet, find the line of code where you want to set a break point.

   ![Example of a VBScript interface]

   **Example of a VBScript interface**

   1. [Delete Commodity]
   2. Function B_BBCommodityDelete(sCode, sUser, sStation)
   3. On Error Resume Next
   4. Dim objRS, sResult, SqlStatement, i
   5. Set objCon = Nothing
   6. Er Clear
   7. Dim objCon
   8. Set objCon = CreateObject("ADODB.Connection")
   9. sResult = "Failed"
   10. 'Create DB connection
   11. Er Clear 'clear error if TypeNum fails above
   12. DBConnect(objCon) 'open database connection
   13. If Err.Number = 0 Then
   14. 'Create DB recordset

   **Tip:** To show/hide line numbers in the VBScript interface, select **Line Number** on the **View** tab of the ribbon. The option is set independently for each worksheet; there is no global setting.

3. Do one of the following:
   * Click in the light blue gutter to the left of that line; or
   * Click in the line to place your cursor, and then click **Break Point** on the ribbon.
A red break point symbol is inserted. If a break point cannot be inserted exactly where you clicked — for example, if it is a `Dim` statement that does not actually change any values — then it will be automatically inserted at the next possible line.

4. Repeat these steps for each break point that you want to set.

5. Save and close all open VBScript worksheets.

Any break points that you set are saved with the worksheet(s).

**Run your project in Debug mode**

Run your project in Debug mode in order to use the other debugging tools.

A project can be run either in normal Run mode or in Debug mode. You must run your project in Debug mode in order to use the other debugging tools such as break points, stepping, and the VBScript-specific tabs in the Database Spy window.

To run your project in Debug mode:

1. On Debug tab of the ribbon, in the Local Management group, click Debug.

The project runtime is started in Debug mode. If the Debug Viewer option (on the Debug tab, in the Debug Options group) is selected and the Viewer task is set to start up, then the Secure Viewer is also started.

**Note:** When a project is run locally, the standard Viewer is normally used. In contrast, when a project is run in Debug mode, the Secure Viewer is used for better thread and process management.

The VBScript code is executed up to the first break point that you set, assuming you set break points. (The “first” break point is the first one that comes in the logical, sequential execution of the code, not literally the first one that you set.) At that point, the project automatically enters break mode: execution is...
suspended and a yellow arrow is displayed in the VBScript interface to show exactly where in the code that execution was suspended.

If you did not set break points, then the project will continue to run until you click either Stop (to stop the project runtime) or Break (to manually enter break mode) on the Debug tab.

2. Once the project is in break mode, do one of the following:
   - Click Continue on the Debug tab to resume execution and continue to the next break point.
   - Check the Database Spy window to see the current state of the project. For more information, see Observe the current state in the Database Spy window on page 987.
   - Click Step Into, Step Over, or Step Out on the Debug tab to step through the code one line at a time. For more information, see Step through your VBScript code on page 989.

   **Note:** While your project is in break mode, you can hover the mouse pointer over any VBScript variable in the code to get the current value of that variable. The value is shown in a pop-up box.

3. Repeat the previous step as desired.

4. When you have finished debugging your code, click Stop on the Debug tab.
Caution: When you run a project in Debug mode, you will not be able to stop it until the project’s Startup Script has been completely executed.

Observe the current state in the Database Spy window

Use the Database Spy window to see the current state of the project while it is in break mode.

Your project must already be in break mode, either by reaching a break point that you set earlier or by using the Break tool to manually enter break mode, before you can use these tools.

The Database Spy window has four basic tabs (DB 1 through DB 4) that can be used at any time to view and adjust project tags and to execute in-line scripts. All four of those tabs work in the same way; four of them are provided simply to give you space to organize your work. For more information about using those tabs, see Database Spy on page 82.

Tip: You may enter the names of VBScript functions (but not sub-routines) in the Database Spy window to get the returned values of those functions.

In addition to those four tabs, however, are three new tabs that only work with VBScript debugging: Locals
This tab shows all of the locally-declared VBScript variables and the current values of those variables.

**Locals tab of the Database Spy window**

**Stack Frame**

This tab shows additional information about the VBScript interfaces that are currently open for debugging. The interfaces are listed in order of mostly recently executed, so that you can see how one script calls functions in another, and the **Line** column shows the most recently executed line in each interface.

Therefore, in the screenshot below, you can see that execution was suspended at line 24 in the function **Main_02** in the Global Procedures, and that the function **Main_02** was previously called at line 97 of the Startup Script.

**Stack Frame tab of the Database Spy window**

**Tasks**

This tab shows all of the run-time tasks that being debugged and the current state of each task.

**Tasks tab of the Database Spy window**

**Note:** At this time, only Background Task (Script) can be debugged, so it is the only task that will ever be shown in this tab. Other tasks may be shown in the future, as the debugging feature is enabled for more VBScript interfaces.

Keep in mind that these tabs show information only when a project is running in Debug mode and the project is in break.
Step through your VBScript code

Use the Step Into, Step Over, and Step Out tools to step through your VBScript code one line at a time.

Your project must already be in break mode, either by reaching a break point that you set earlier or by using the Break tool to manually enter break mode, before you can use these tools.

1. To advance one step in the code, no matter what the step may be, click Step Into on the Debug tab.
This always moves the debugger a single step forward. If the next step is a function call, then the function is called and execution is suspended again at the first step of that function. That is what "step into" means: the debugger steps into the called function.
The yellow arrow is moved to show that the step has been taken.

2. To advance one step in the main script only, click **Step Over** on the **Debug** tab.

   If the next step is a function call, then the entire function is executed and the debugger continues with the main script. That is what "step over" means: the debugger steps over the called function in its entirety. The function is not skipped — it is still executed as written — but no additional time is spent stepping through it.

   **Note:** You should use this tool only when you have already debugged the function and you trust it to execute correctly.

The yellow arrow is moved to show that the step has been taken.

3. To finish executing a function and then continue with the main script, click **Step Out** on the **Debug** tab.

   That is what "step out" means: the debugger steps back out of a function that it has already stepped into. The function is not aborted — it is still executed as written — but no additional time is spent stepping through it.

   **Note:** You should use this tool only when you have already debugged the function and you trust it to execute correctly.

The yellow arrow is moved to show that the step has been taken.

After every step, the Database Spy window is updated to show the current state of the project. For more information, see **Observe the current state in the Database Spy window** on page 987.